2147130 LAB 7

Q) 1. Describe DBaaS.

ANS → database as a service is a cloud computing managed service offering that provides access to a database without requiring the setup of physical hardware, the installation of software or the need to configure the database. Most database administration and maintenance tasks are handled by the service provider, enabling users to benefit quickly from the database service.

Q) 2. List the Different Database Engines available in AWS and GCP.

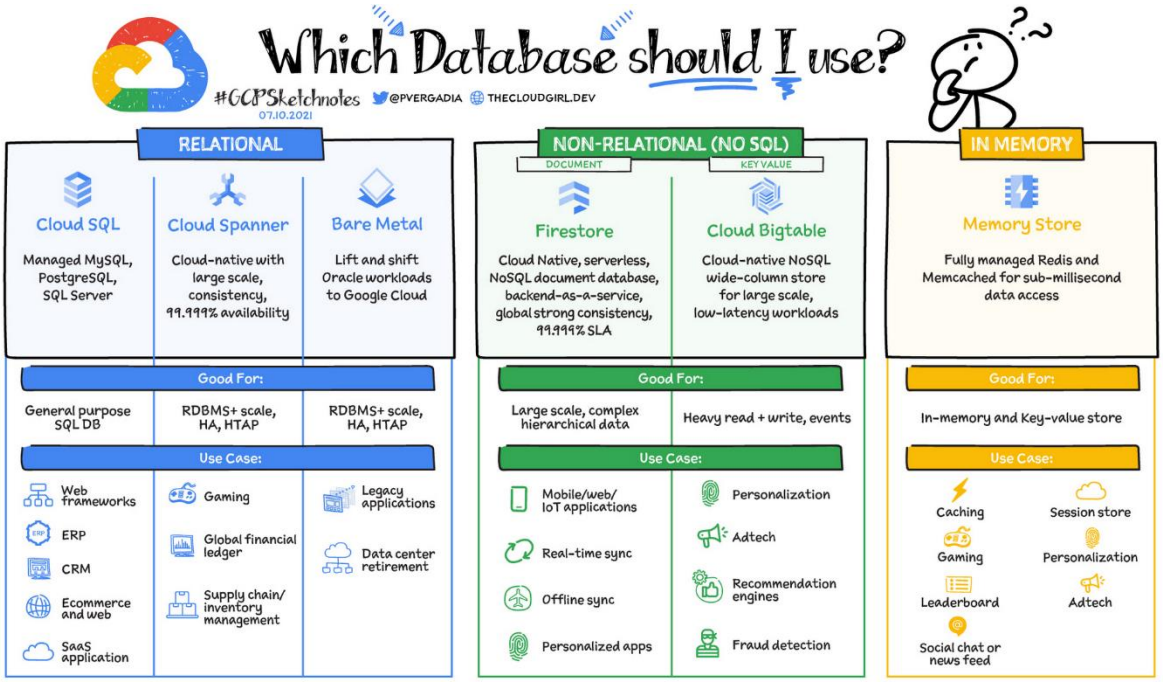
ANS →

The table below shows the main NoSQL databases services offered by AWS.

Type of Database	Use Cases	Amazon Services
Key-value Key-value databases store data as a collection of key-value pairs with the key as an ID. These databases can store various types of data, including simple and compound objects.	● Real-time bidding ● eCommerce shopping carts ● Product catalogs ● Customer preferences	● Amazon DynamoDB
Document Document databases store data in JSON or JSON-like documents. You can query data using the same document-model format used in programming applications.	● Cataloging ● Content management systems ● Customer profiles and personalization ● Mobile apps	● Amazon DocumentDB
In-memory In-memory databases store data in-memory for low-latency access. You can use these stores as a database, cache, message broker, or queue.	● Caching ● Session stores ● Gaming ● Leaderboards ● Geospatial services ● Pub/sub messaging ● Real-time streaming	● Amazon ElastiCache for Memcached ● Amazon ElastiCache for Redis
Graph Graph databases are a type of NoSQL (non-relational) database. This database type represents relationships directly. You can query data with specific graph languages.	● Fraud detection ● Social networking ● Recommendation engines ● Knowledge graphs ● Data lineage	● Amazon Neptune
Time-series Time-series databases store data in time-order and as append-only. You can query data over various time intervals.	● DevOps ● Application monitoring ● Industrial telemetry ● IoT applications	● Amazon Timestream

Ledger Ledger databases store data in an immutable, transparent, and cryptographically verifiable log. This log is owned by a trusted central authority to ensure provenance.	● Finance ● Manufacturing ● Insurance claims ● HR and payroll ● Retail inventories	●Amazon Quantum Ledger Database

In Gcp



Q) 3. Assume that you are developing a dashboard to collect and displaying the details of the "Covid19" cases reported in different states of our country on daily basis. The requirements are as follows,

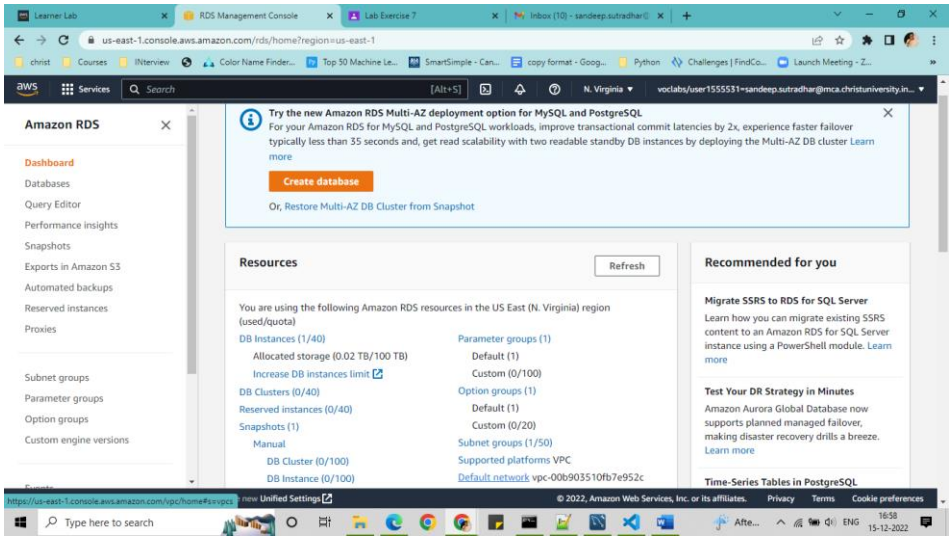
a. In order to provide seamless access to all the stakeholders, Host your application in a virtual machine instance on a Cloud Environment.

b. Add the provision in the web page to enter the details such as, Name of the state, Date of record, No samples collected, No of positive and negative cases, No of discharges and No of deaths reported from each state. Also, display the state wise positive cases in ascending order to support the decision-making process.

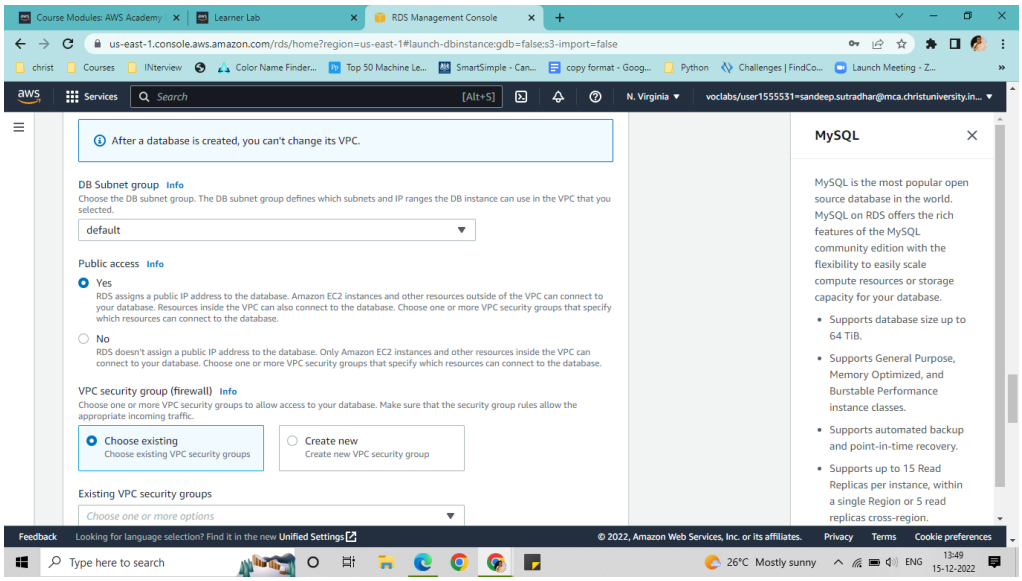
c. To store and process these details in a scalable database environment, Create a MySQL Database Server Using Amazon RDS (Relational Database Service) and Create a Database "Covid19" and a table Covid_details with the following attributes. State_Name, Date_of_Record, No_of_Samples, No_of_Deaths, No_of_Positive, No_of_Negative, No_of_Discharge.

ANS →

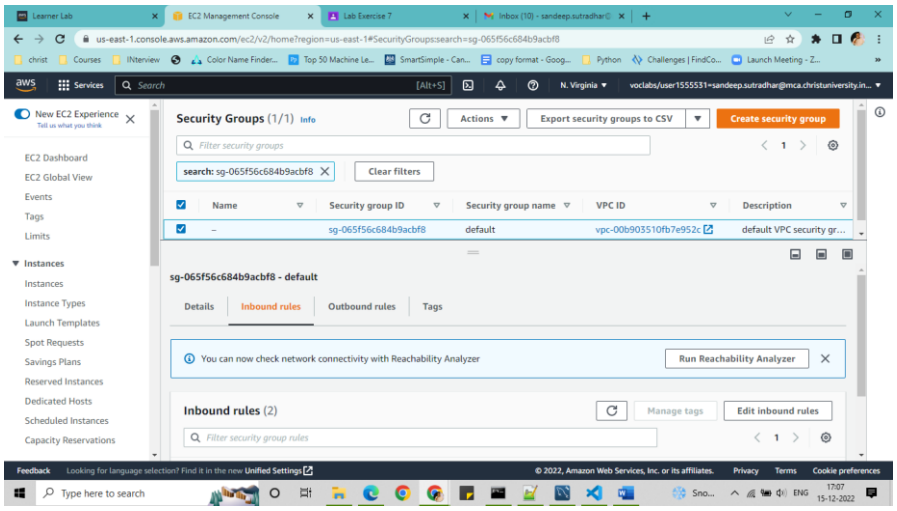
STEP 1 → Login to AWS Academy learner Lab → Search on the search bar for RDS → Click on DB Instance → and Click Create Database



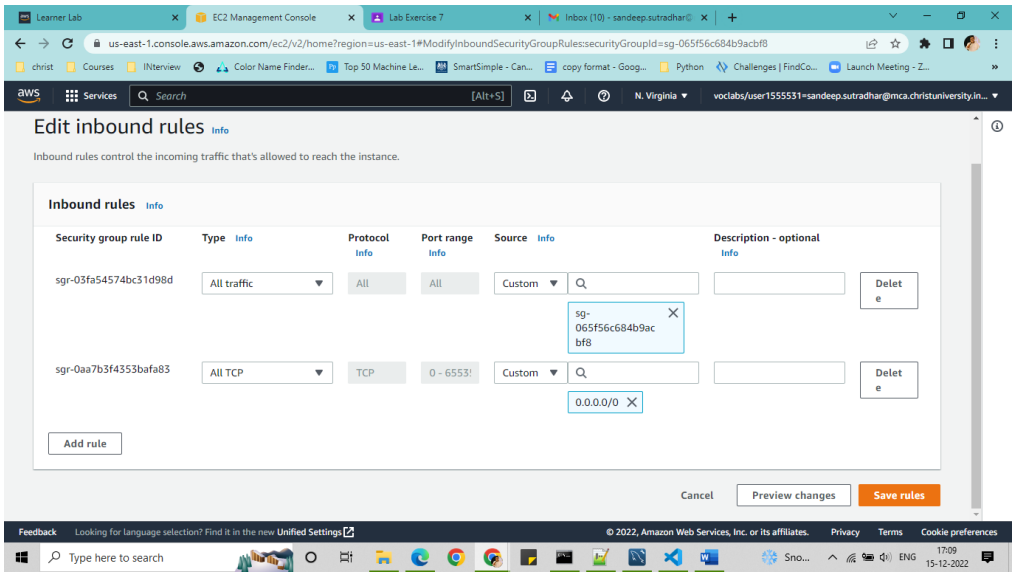
STEP 2 → In the engine option Choose “MySQL” → In templates choose Free tier → Give database cluster identifier(db name) as COVID19 → Under the Credential settings give Master username as “admin” and password “ 12345678 ” → Set Public access to “Yes” → Leave everything as same → select **Create database** → It will take some time to turn on



Step 3→ Click on the database just created (Covid9) → Go to Security group which is default →Click Inbound rules → click edit inbound rules

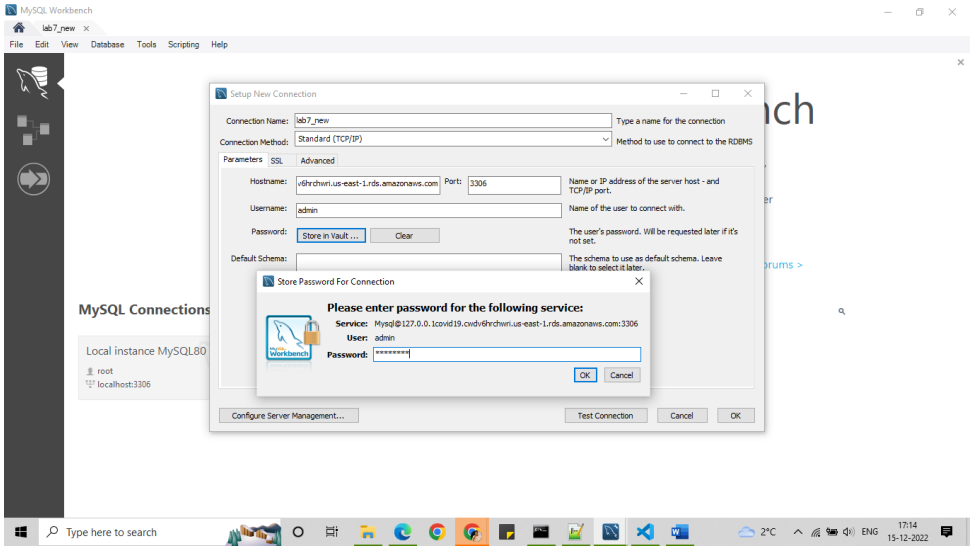


Click Add Rule → in the Type select All TCP → In the Source choose Custom anywhere from ipv4 → click save rules

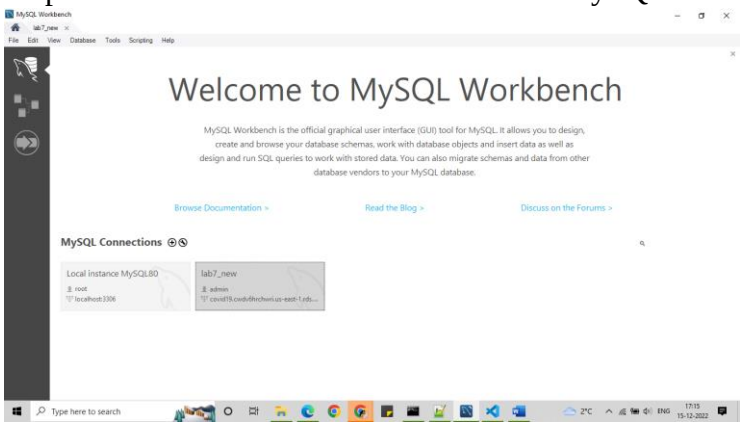


STEP 4 → CONNECT WITH MYSQL

Select the endpoint of database and copy it ("covid19.cwdv6hrchwri.us-east-1.rds.amazonaws.com")→ Open Mysql Workbench -→ Click on the + button → Give connection name ("lab7_new") → Hostname as the link we coopied → username as admin → password choose store in Vault and type "12345678" → Ok → Click OK

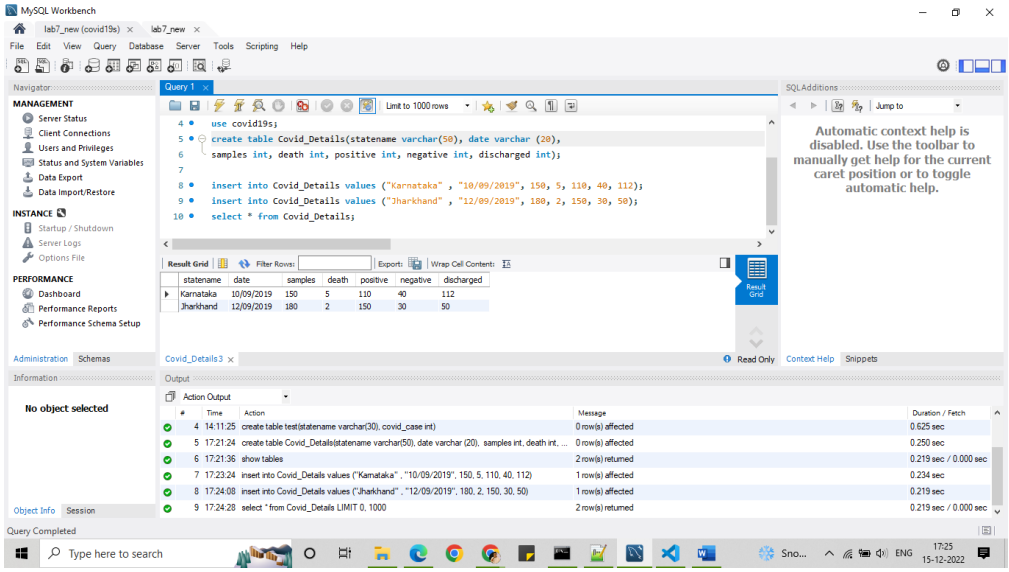


Step 5 → It will create the connection with MySQL and aws server → Click on it



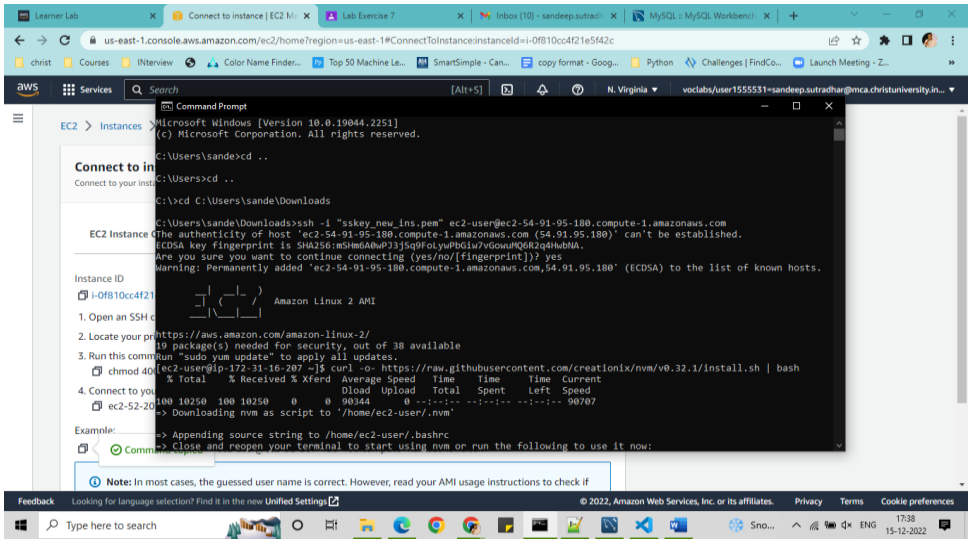
Step 6 →Create database (“covid19s”) → Make table with the following arguments

```
create database covid19s;

use covid19s;

create table Covid_Details(statename varchar(50), date varchar (20),

samples int, death int, positive int, negative int, discharged int);

insert into Covid_Details values ("Karnataka" , "10/09/2019", 150, 5, 110, 40, 112);

insert into Covid_Details values ("Jharkhand" , "12/09/2019", 180, 2, 150, 30, 50);

select * from Covid_Details;
```



STEP 7 → Go to EC2 instance → Launch Instance → Give name as ss_lab_new →In the AMI choose Amazon Linux →Key Pain (login Choose Create new (ss_key_new) and Download it → In the firewall security group choose existing one where HTTP and HTTPS traffic is allowed or you can create new according to your requirement → Leave everything as it is → Click Launch Instance

Step 8 → Click the instance then select connect → Choose SSH client → Copy the link given in example → Open terminal in your system → Open the directory where the .pem file has been downloaded → Paste the link and enter.



STEP 9 → Type the following codes –

curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.32.1/install.sh | bash

source ~/.bashrc

nvm install 7

node –version

STEP 10 → create directory to store all the files →

mkdir app1
cd app1
#nano node.js

```
const mysql=require('mysql2')
const express=require('express')
const port =7000;
const app=express();
const bodyParser=require('body-parser');
const cors=require('cors');
app.use(cors({
    orgin: "*",
}))

app.use(bodyParser.urlencoded({extended:true}));
app.use(express.static('public'));
const connection=mysql.createConnection(
    {
        host:'covid19.cwdv6hrchwri.us-east-1.rds.amazonaws.com',
        user:'admin',
        password:'12345678',
        database:'covid19s'
    }
)
connection.connect((err)=>
{
    if(!err)
    console.log('Database connection eshtablished....!')
    else
    console.log('Error')
})
app.listen(port,()=>
{
    console.log('Connection Eshtablished successfully....!!!')
}
)
app.get('/getjson',(req,res)=>
{
    connection.query("SELECT * from Covid_Details",(err,rows,fields)=>
    {
        if(!err)
        res.send(rows);
        else
        console.log('Error in Displaying')
```

```
    })
})
app.get('/',(req,res)=>
{
    res.sendFile(__dirname +"/inputform.html");
})
app.get('/display',(req,res)=>
{
    res.sendFile(__dirname +"/display.html");
})


app.post('/add', (req, res) => {
    console.log(req.body);
    var sql="INSERT INTO Covid_Details
VALUES('"+req.body.statename+"','"+req.body.date+"',"+
parseInt(req.body.samples)+","+parseInt(req.body.death)+","+parseInt(req.body.
positive)+","+parseInt(req.body.negative)+","+parseInt(req.body.discharged)+")
;"
    // var sql="INSERT INTO Covid_Details VALUES('"+req.body.statename+"','"+
parseInt(req.body.samples)+");"

    console.log(sql)
    connection.query(sql,(err,rows,fields)=>
    {
        if(!err)
        console.log('Data has been inserted successfully...!')
    })
    res.send('Data has been inserted successfully...!')
})
```

#nano display.html

```
<!DOCTYPE html>
<html lang="en" ng-app="myapp" ng-controller="ctrldetails">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></
script>
    <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular-
route.min.js"></script>
    <script src="/angularapp.js"></script>
    <title>Covid19 Details</title>
    <style>
        *{
            font-family: Arial, Helvetica, sans-serif;
            border-collapse: collapse;
```

```
            margin:10px;
        }

        td, th {
        border: 1px solid #ddd;
        padding: 8px;
        }

        tr:nth-child(even){background-color: #f2f2f2;}

        tr:hover {background-color: #ddd;}

        th {
        padding-top: 12px;
        padding-bottom: 12px;
        text-align: left;
        background-color: #000;
        color: white;

        }
    </style>

</head>

<body>
    <center>
        <h1> Covid19 Details</h1>
        <table>
            <tr>
                <th width ="10%">State Name<br>
                <th width ="10%">Date</th>
                <th width ="10%">Number of Samples</th>
                <th width ="10%">Number of deaths</th>
                <th width ="10%">Number of positive</th>
                <th width ="10%">Number of negative</th>
                <th width ="10%">Number of discharges</th>
            </tr>
            <tr ng-repeat="item in table">
                <td>{{item.State_Name}}</td>
                <td>{{item.Date_of_Record}}</td>
                <td>{{item.No_of_Samples}}</td>
                <td>{{item.No_of_Deaths}}</td>
                <td>{{item.No_of_Positive}}</td>
                <td>{{item.No_of_Negative}}</td>
                <td>{{item.No_of_Discharge}}</td>
            </tr>
        </table>
        <!-- <p><h3>The total number of participants: {{count}} </h3></p> -->
    </center>

</body>

</html>
```

#nano inputform.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form</title>
</head>
<body>
    <center>
        <h1>Add data into the database</h1>
        <form method="POST" action="/add">
            Enter State Name:<br><input type="text" name="statename"
/><br><br>
            Enter Date of Record:<br><input type="date" name="date" /><br><br>
            Enter Number of samples:<br><input type="number" name="samples"
/><br><br>
            Enter Number of death:<br><input type="number" name="death"
/><br><br>
            Enter Number of Postive:<br><input type="number" name="positive"
/><br><br>
            Enter Number of Negative:<br><input type="number" name="negative"
/><br><br>
            Enter Number of Discharged:<br><input type="number"
name="discharged" /><br><br>
            <input type="submit" value="ADD">
        </form>
        <a href="/display">Display</a>
    </center>
    <!-- <script scr="node.js"></script> -->
</body>
</html>
```

#mkdir public

#cd public
#nano angularapp.js

```
angular.module('myapp',[])
.controller('ctrldetails',function($scope,$http)
{
    $http.get('http://127.0.0.1:7000/getjson')
    .success(function(response)
    {
        $scope.table=response;
        $scope.count=$scope.table.length;
    })
})
```

#cd..

sudo iptables -t nat -I PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 7000

STEP 10 → Come to the directory app1 (where the node file is stored) → and install the dependencies → type

#npm install mysql2
#npm install express
#npm install cors

Step 11→ Run the Node file → in the app1 directory → type

#node node.js → it will give a message → “Database connection eshtablished....!” → come to instance → copy paste the public ip in the browser

NOTE -> we have to change the web address link everytime we start the instance from new → Go to angularapp.js and change the address part only and give the public ip to it