A

PROJECT REPORT ON

"Manipulating Database in AWS Cloud"

SUBMITTED BY

VAGAL DEVESH KIRAN

MASTER OF COMPUTER APPLICATION

SEMESTER III, ROLL NO: 1808

VIKAS COLLEGE STUDY CENTRE, VIKHROLI (EAST) [MUMBAI]

TILAK MAHARASHTRA VIDYAPEETH, PUNE

(2019-2020)

TILAK MAHARSHTRA VIDYAPEETH, PUNE

('Deemed University' under section 3 of UGC Act 1956 vide notification NO.F 9- 19/85-U3 dated 24th April 1987 by the Government of India)
Vidyapeeth Bhavan, Gultekdi, Pune- 411 037

CERTIFICATE

This is to certify the project titled "Manipulating Database in AWS Cloud"

Is a bonafied work carried out by

VAGAL DEVESH KIRAN (ROLL NO: 1808)

By Students of **MASTER of Computer Application**

Semester III

Under Tilak Maharashtra Vidyapeeth in the Year 2019

Head of the Department Examiner Internal Examiner External

Date:
Place:

University Seal

ACKNOWLEDGEMENT

With immense please I am presenting “Manipulating Database in AWS Cloud” Project report as a part of the curriculum of ‘Master of Computer Application’. I wish to thank all the people who gave me unending support.

I express my profound thanks to our Head of Department,Project Guide and Project In-charge “**Mr. VIKAS RAUT**” and all those who have indirectly guided and helped me towards development of this project.

VAGAL DEVESH KIRAN

(ROLL NO: 1808)

INDEX

Ch No. 1 Introduction

A database management system is important because it manages data efficiently and allows users to perform multiple tasks with ease. A database management system stores, organizes and manages a large amount of information within a single software application. Use of this system increases efficiency of business operations and reduces overall costs.Database management systems are important to businesses and organizations because they provide a highly efficient method for handling multiple types of data. Some of the data that are easily managed with this type of system include: employee records, student information, payroll, accounting, project management, inventory and library books. These systems are built to be extremely versatile.

Problem Statement :-

If we compare with the current techniques that are used for database management there are generally more amount of companies that operate their database on the on–premises systems which can cost high maintenance charges. Because of on-premises database management the loss of data may occur due to system failure or if regular backups are not been taken.

With this technique, the companies that use on-premises database can migrate to the AWS Cloud and manage their database on the cloud which is more flexible,secure & at the same time the maintenance cost also will be reduced.

Ch No. 2 Requirement Specifications & Project Highlights

Requirement Specifications :-

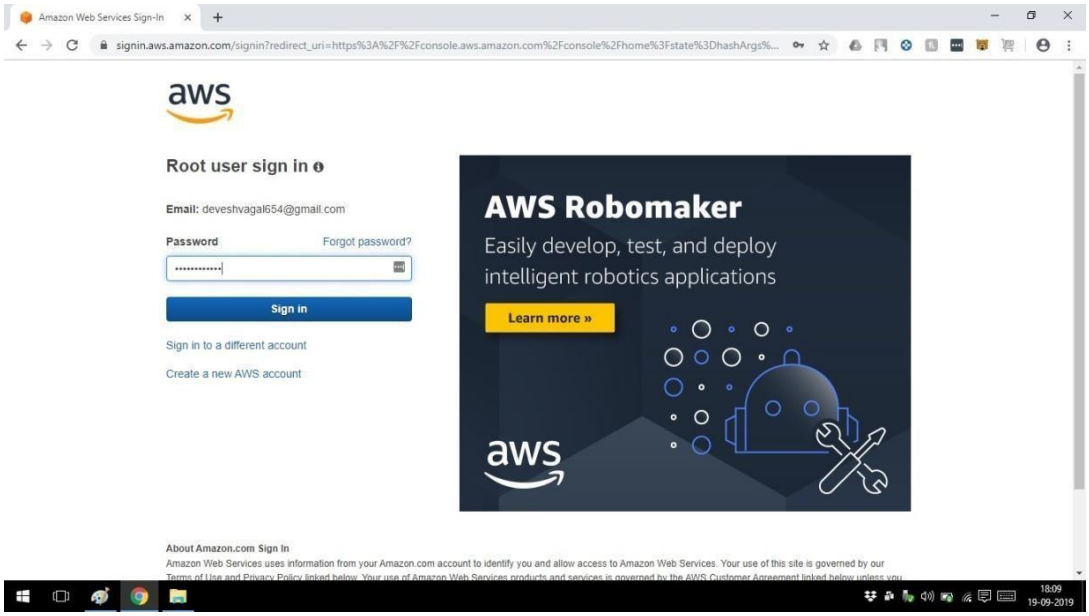
Name of Component	Specification
Operating System	Windows 7 and Above.
Languages Used	Python & SQL.
Development Platform	AWS (Amazon Web Services).
Tools Required	AWS Management Console,AWS Ec2,Anaconda Navigator,Jupyter Notebooks,MySQL connecter package,MySQL Workbench.
Processor	1.33 Gigahertz or Above.

Project Highlights :-

1) Flexible Environment.
2) Secure Cloud Infrastructure.
3) Less Maintenance Cost.

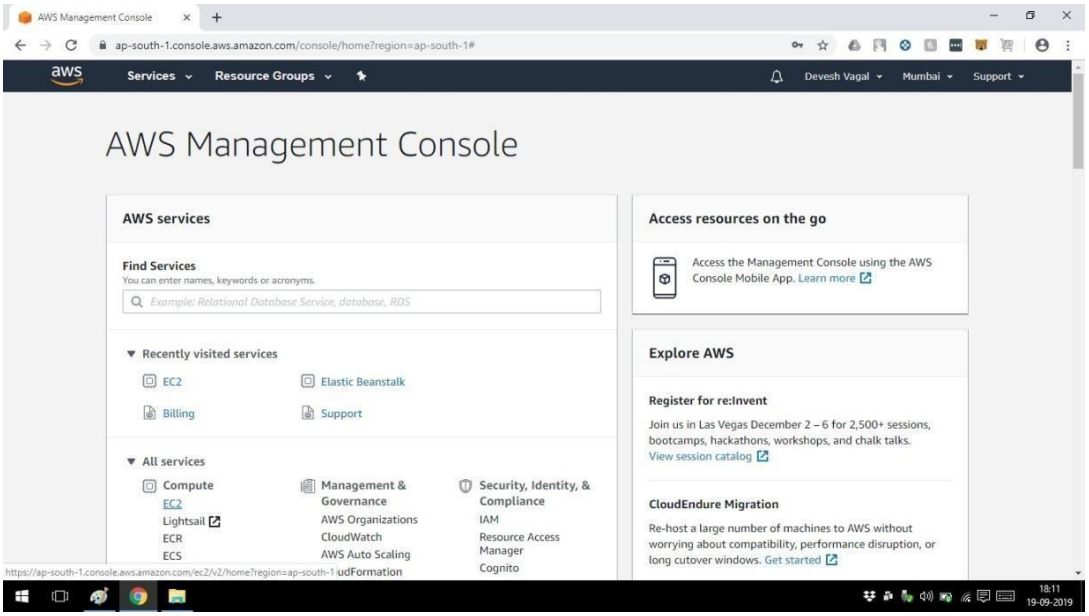
Ch No. 3 Steps to Configure AWS EC2 Cloud Environment

Step 1 :-



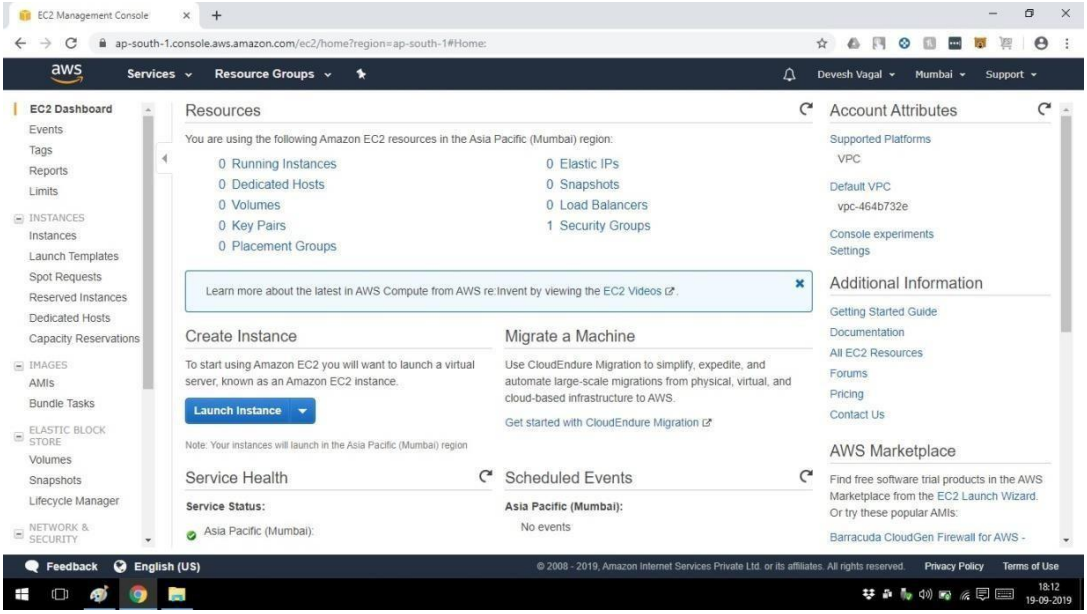
To access your EC2 services first sign in to your AWS Management Console by adding your security credentials.

Step 2 :-



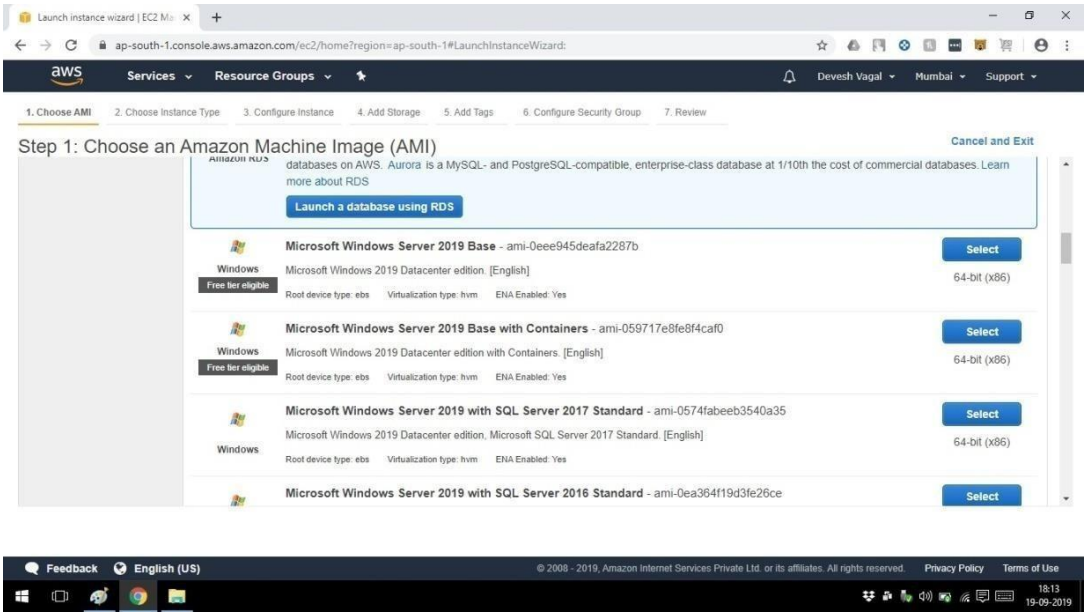
After you have signed in to your AWS Management Console , Select the "EC2" service from "Compute" Section as shown in the above image.

Step 3 :-



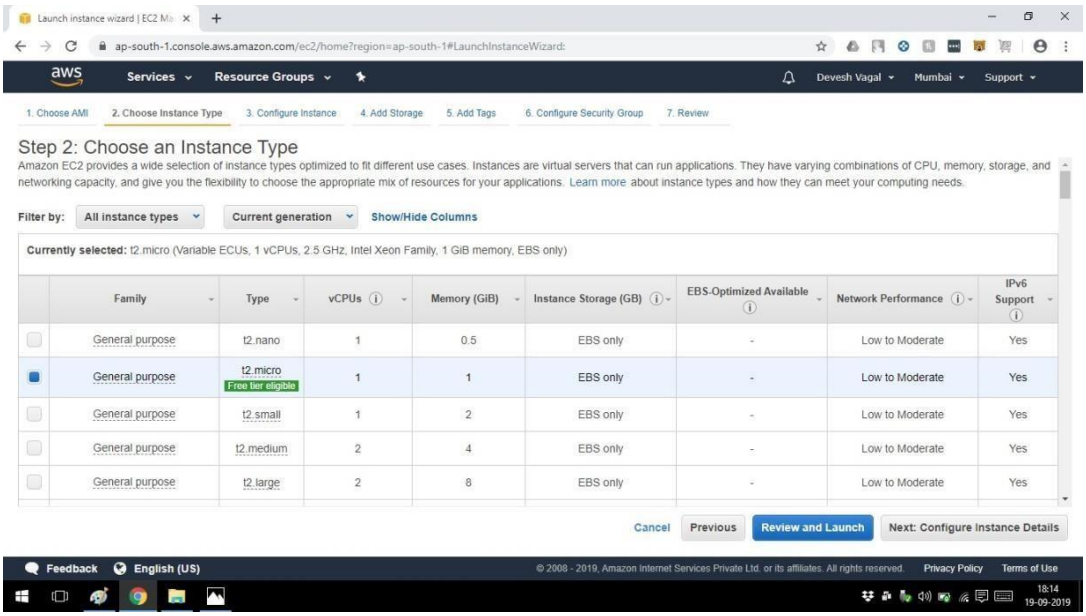
After clicking on "EC2" in AWS Management Console the EC2 Dashboard will appear as per the above image & click on "Launch Instance"

Step 4 :-



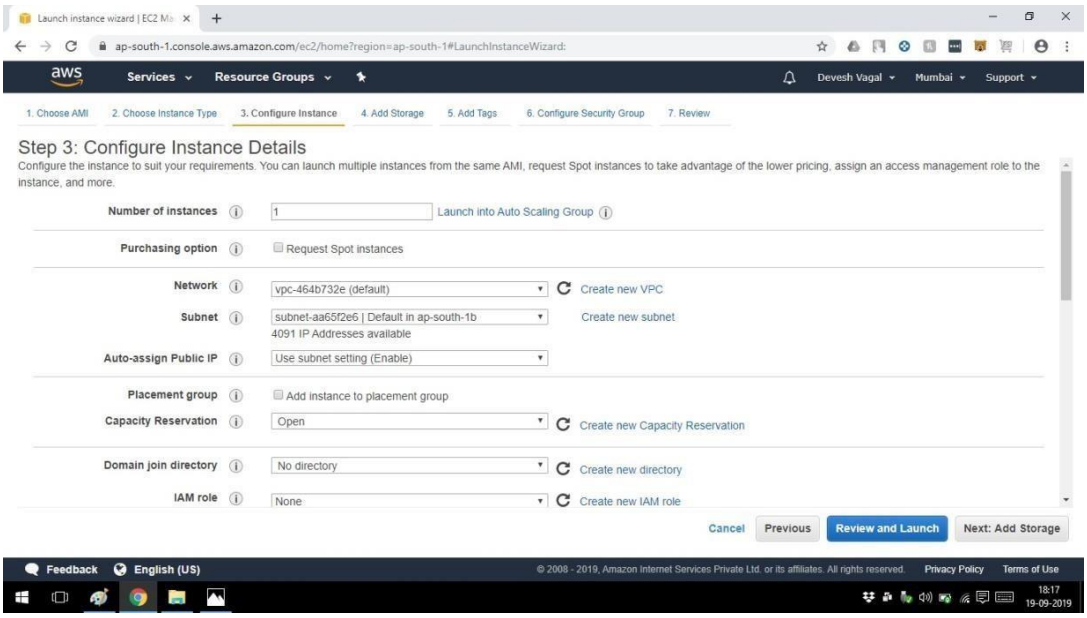
After clicking on "Launch Instance" , we have choose an AMI i.e. the OS which we want to setup for our cloud environment. In this project we have chosen "Microsoft Windows Server 2019 Base" and then clicked "Select".

Step 5 :-



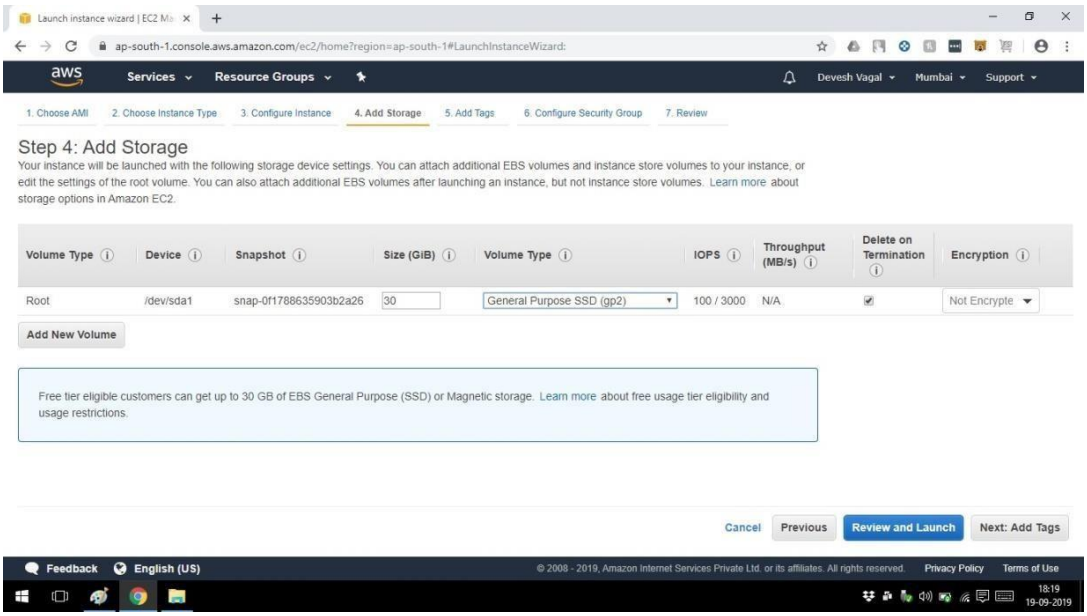
After choosing the AMI , we have to choose the Instance type for our AMI. Click on "Configure Instance Details".InstanceType is basically a Virtual CPU for our cloud environment.

Step 6 :-



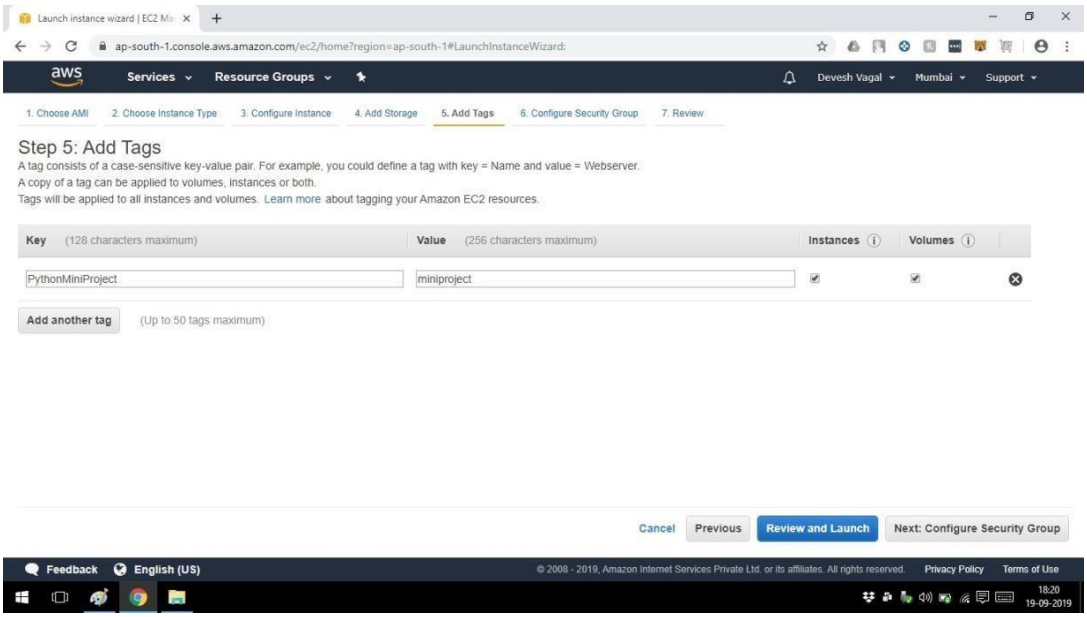
After choosing the Instance type we have to configure the Instance that we have chosen. In this, we have to configure No of instances we want for this project we have used only 1 instance and the VPC used for this project is the default VPC but we can create our custom VPC as well. After completing click on "Add Storage".

Step 7 :-



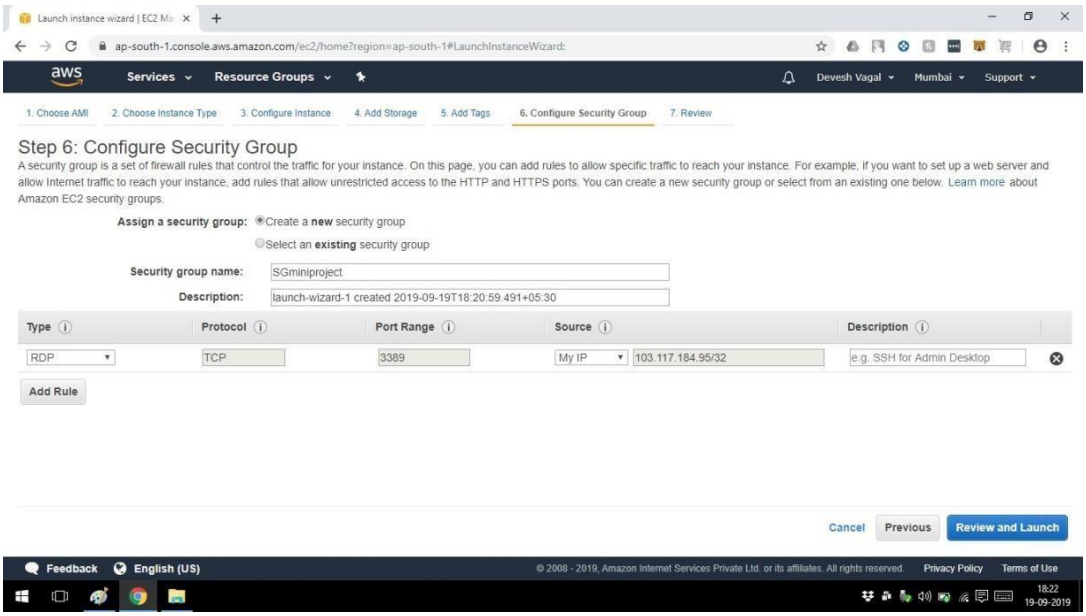
After configuring the Instance we have added storage to our windows system,So in the root row in the size column add the amount of storage as you need for this project we have added "30 GB" as our storage.

Step 8 :-



After adding the storage we have to add tags to our EC2 Instance so that we can easily recognize our instance if there are more than one instances in our instance list. After adding tags click on "Configure Security Group".

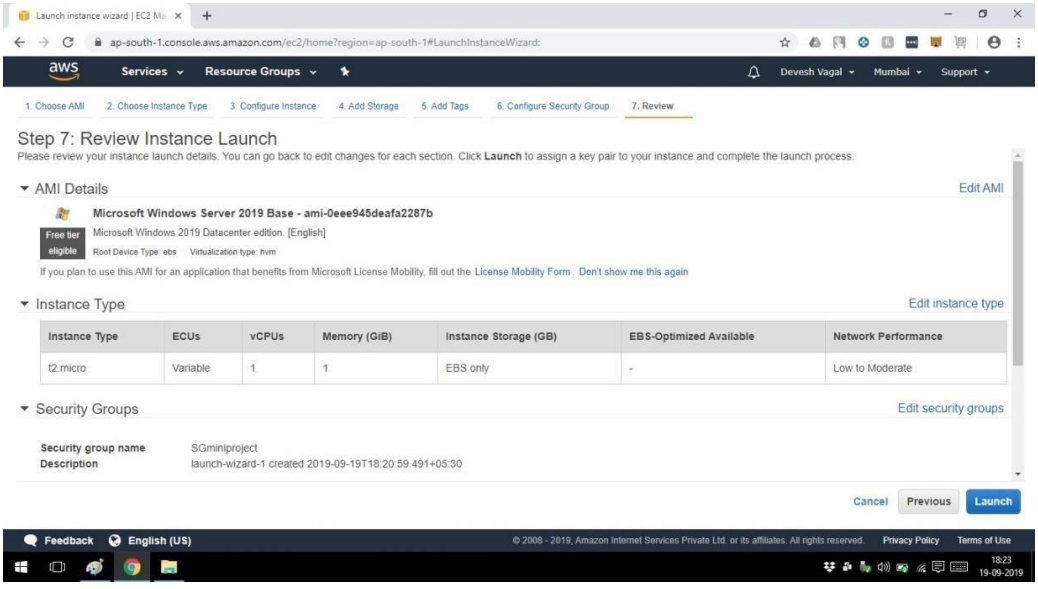
Step 9 :-



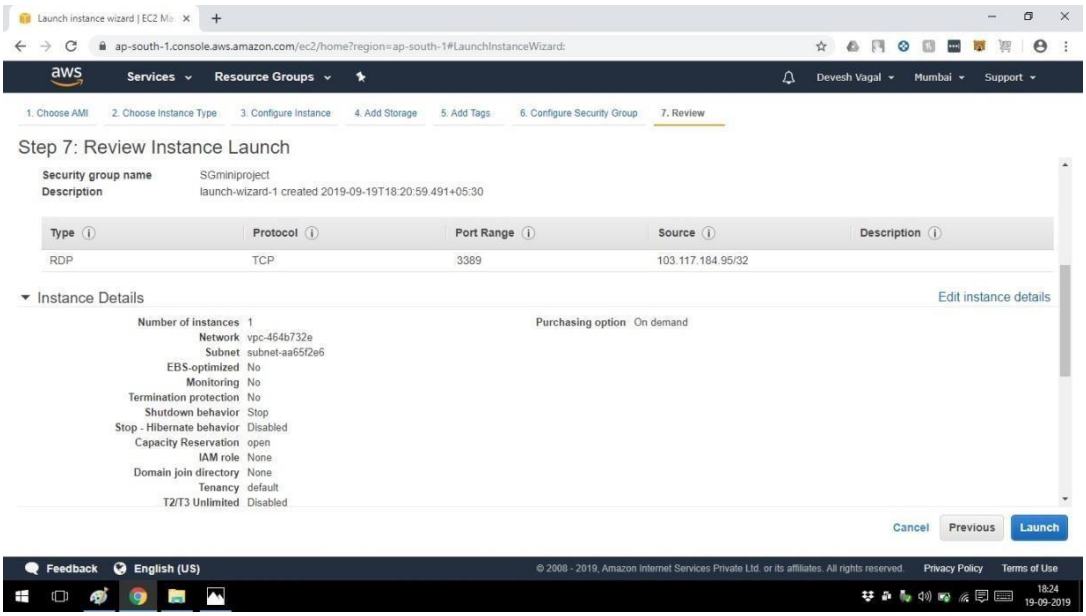
After adding the tags we have to give the security groups. Security groups are basically the firewalls that will control the traffic for our instance. As we have chosen windows OS for our project we have chosen "RDP" as our security group & assigned the IP address for the same. After assigning the security group(s) , click on "Review and lunch".

Step 10 :-

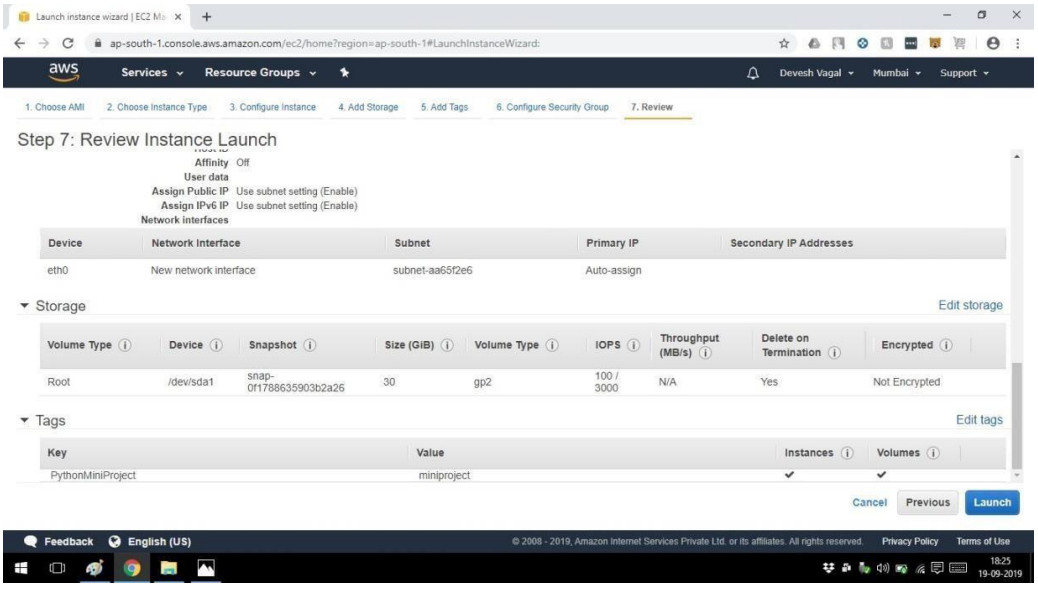
(a)



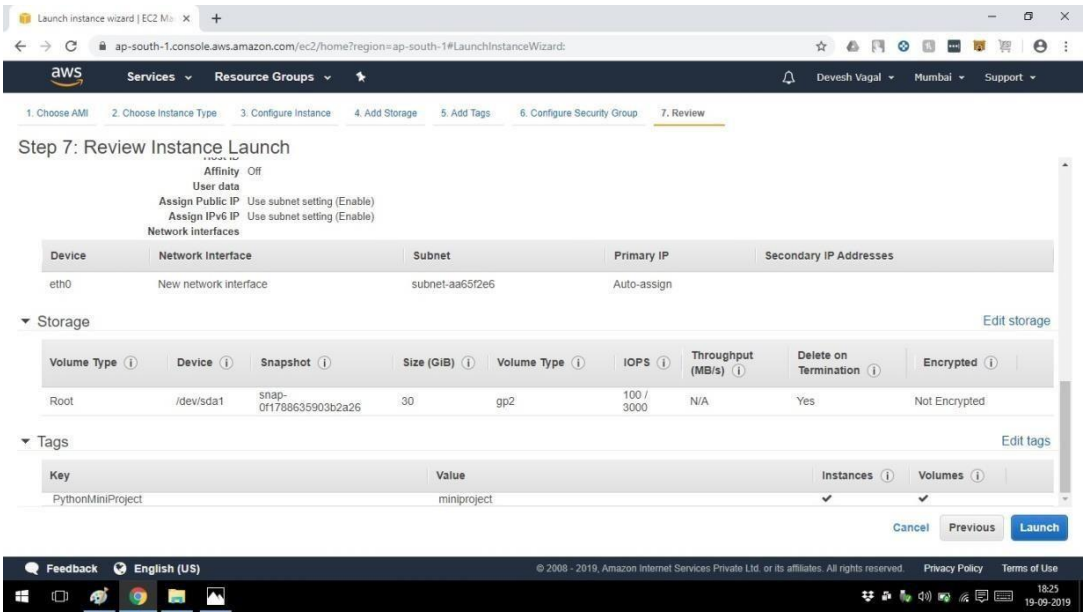
(b)


Launch instance wizard | EC2 Ma
ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstanceWizard:
aws
Services
Resource Groups
Devesh Vagal
Mumbai
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 7: Review Instance Launch
Security group name SGminiproject
Description launch-wizard-1 created 2019-09-19T18:20:59.491+05:30
Type
Protocol
Port Range
Source
Description
RDP
TCP
3389
103.117.184.95/32
Instance Details
Edit instance details
Number of instances 1
Purchasing option On demand
Network vpc-464b732e
Subnet subnet-aa65f2e6
EBS-optimized No
Monitoring No
Termination protection No
Shutdown behavior Stop
Stop - Hibernate behavior Disabled
Capacity Reservation open
IAM role None
Domain join directory None
Tenancy default
T2/T3 Unlimited Disabled
Cancel
Previous
Launch
Feedback
English (US)
© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
18:24
19-09-2019

(c)


Launch instance wizard | EC2 Ma
ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstanceWizard:
aws
Services
Resource Groups
Devesh Vagal
Mumbai
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 7: Review Instance Launch
Affinity Off
User data
Assign Public IP Use subnet setting (Enable)
Assign IPv6 IP Use subnet setting (Enable)
Network interfaces
Device
Network Interface
Subnet
Primary IP
Secondary IP Addresses
eth0
New network interface
subnet-aa65f2e6
Auto-assign
Storage
Edit storage
Volume Type
Device
Snapshot
Size (GiB)
Volume Type
IOPS
Throughput (MB/s)
Delete on Termination
Encrypted
Root
/dev/sda1
snap-0f1788635903b2a26
30
gp2
100 / 3000
N/A
Yes
Not Encrypted
Tags
Edit tags
Key
Value
Instances
Volumes
PythonMiniProject
miniproject
Cancel
Previous
Launch
Feedback
English (US)
© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
18:25
19-09-2019

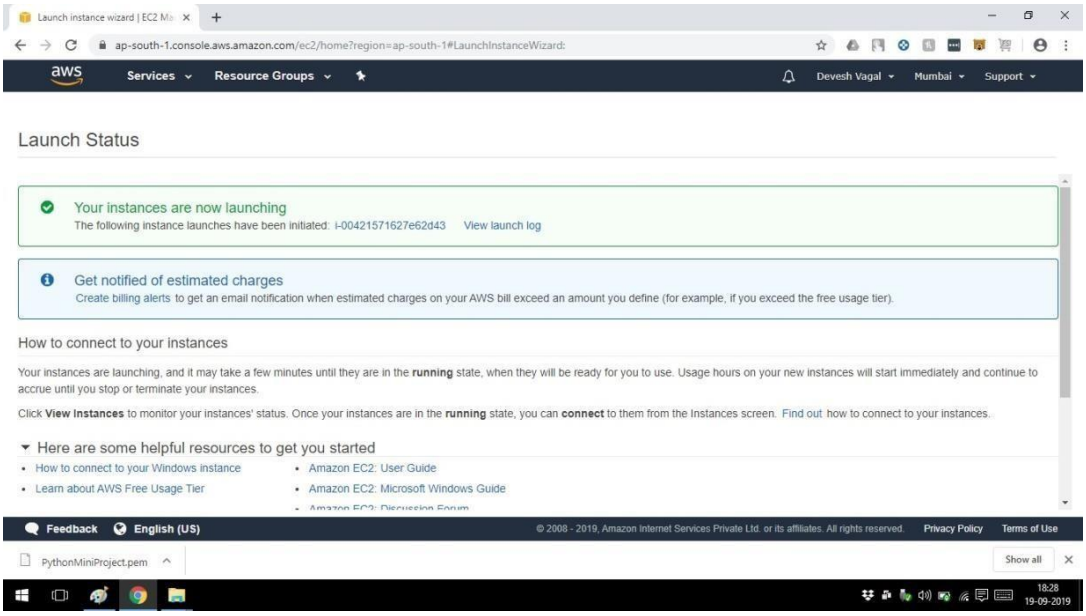
(d)



In this step we will review the whole configuration from start to end to confirm if everything is configured as per the requirements or not. After reviewing all the steps click on the "Launch".

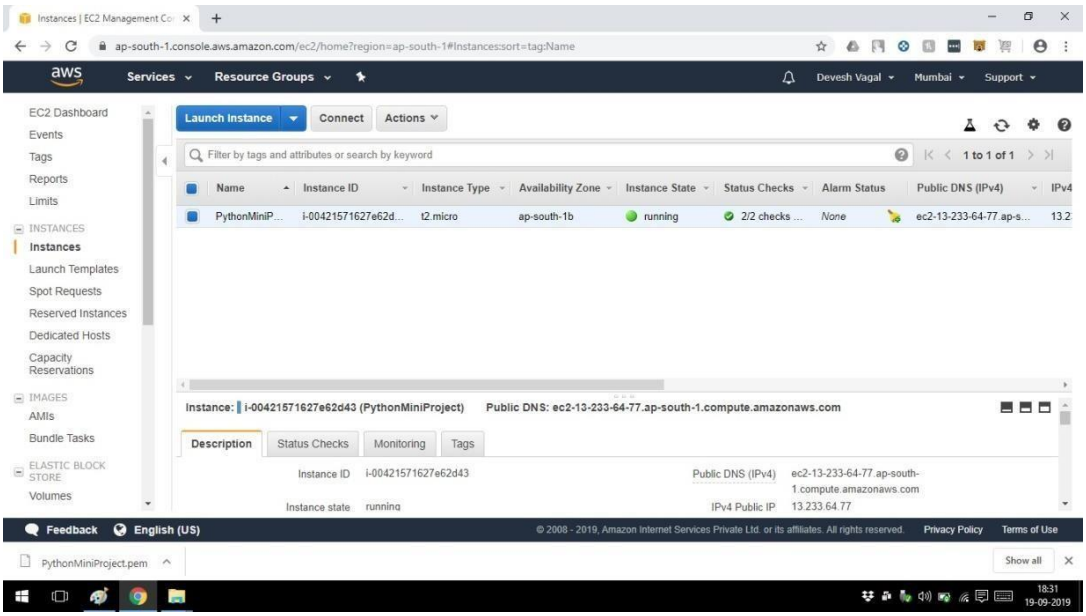
Step 11 :-

(a)



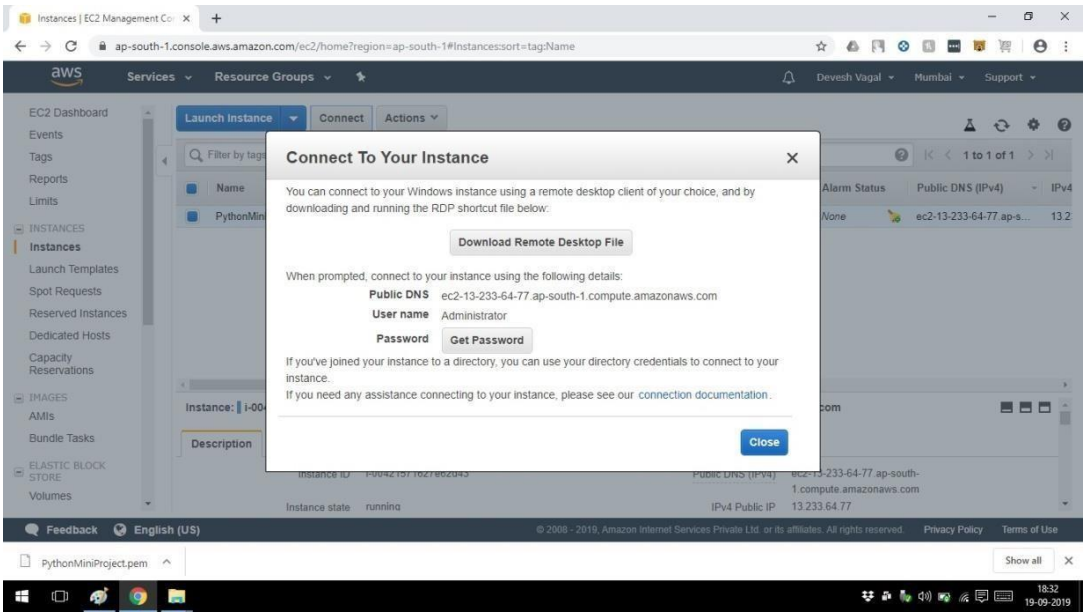
After clicking on "Launch" if all configurations are done properly the instance will be launched successfully. To view the instance click on "View Launch Log".

(b)



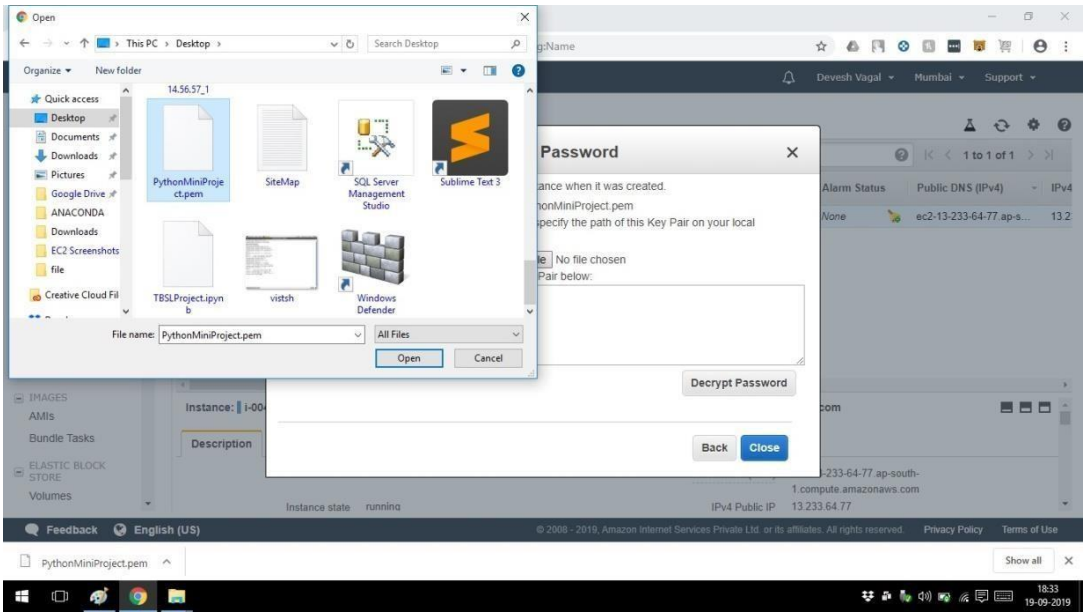
After clicking on "View Launch Log" you will be taken to the list of instances and you will be seeing that the instance we have created is up and running successfully. Click on "Connect".

Step 12 :-



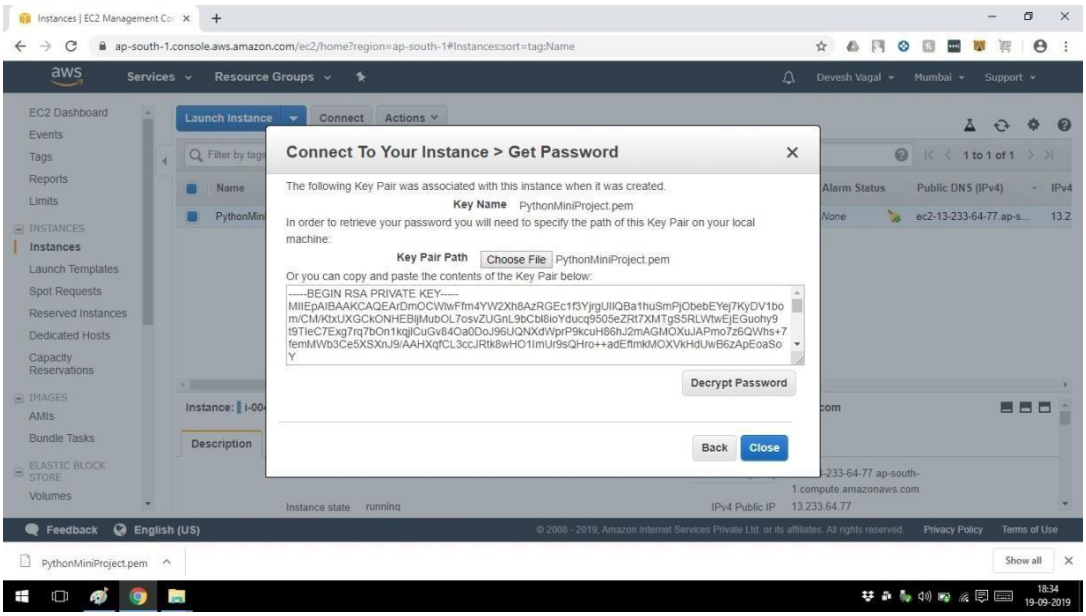
After clicking on "connect" the window will open called "Connect To Your Instance".In this click on "Get password",an ".pem" file will be downloaded as shown in the image above , which is the key to launch the virtual machine from the instance.

Step 13 :-



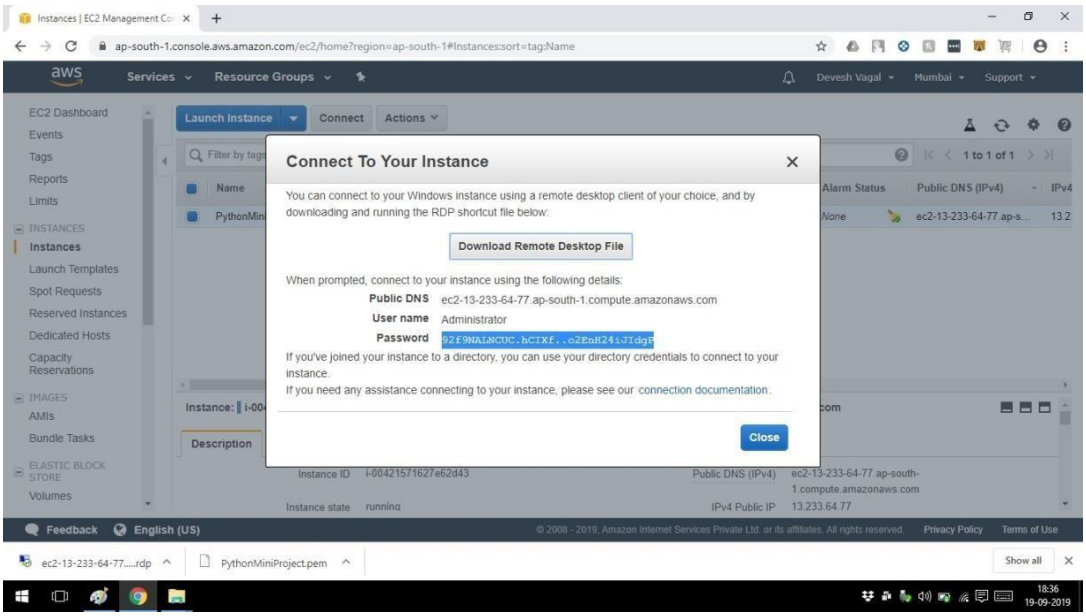
After clicking on “Get Password” an window will open called “Decrypt Password”. In that click on choose file and choose the “.pem” file(key) that was been downloaded and click “Open”.

Step 14 :-



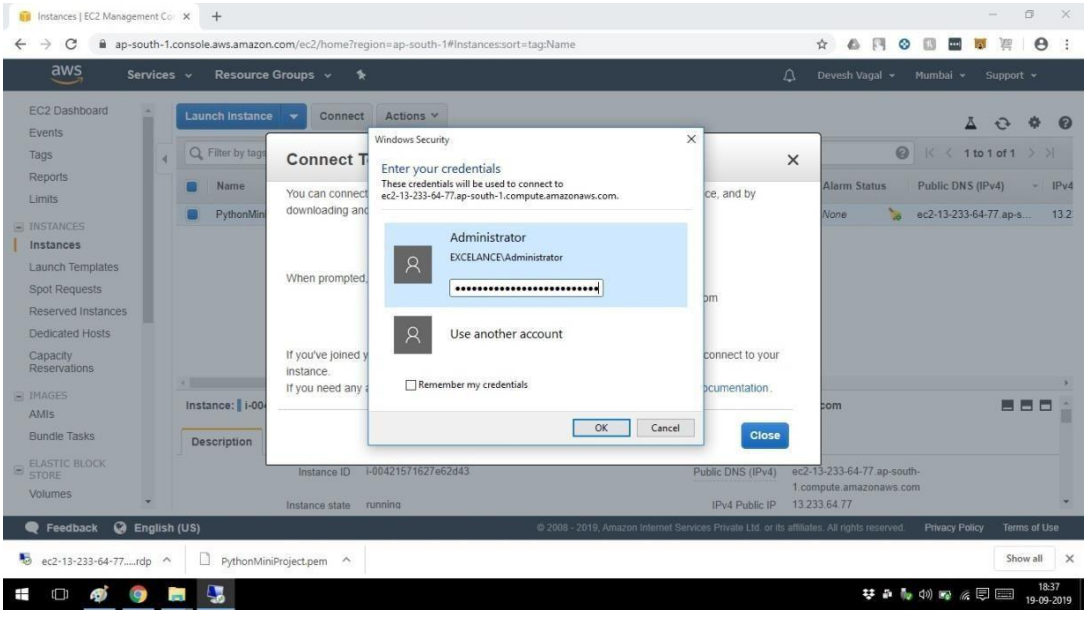
After clicking on “Open” the password pair characters will be displayed in the box. To view the encrypted password click on “Decrypt Password”.

Step 15 :-



After clicking on "Decrypt Password" the encrypted password will be displayed. Copy the password and click on "Download Remote Desktop File" , a remote file will be downloaded as per shown in the above image.

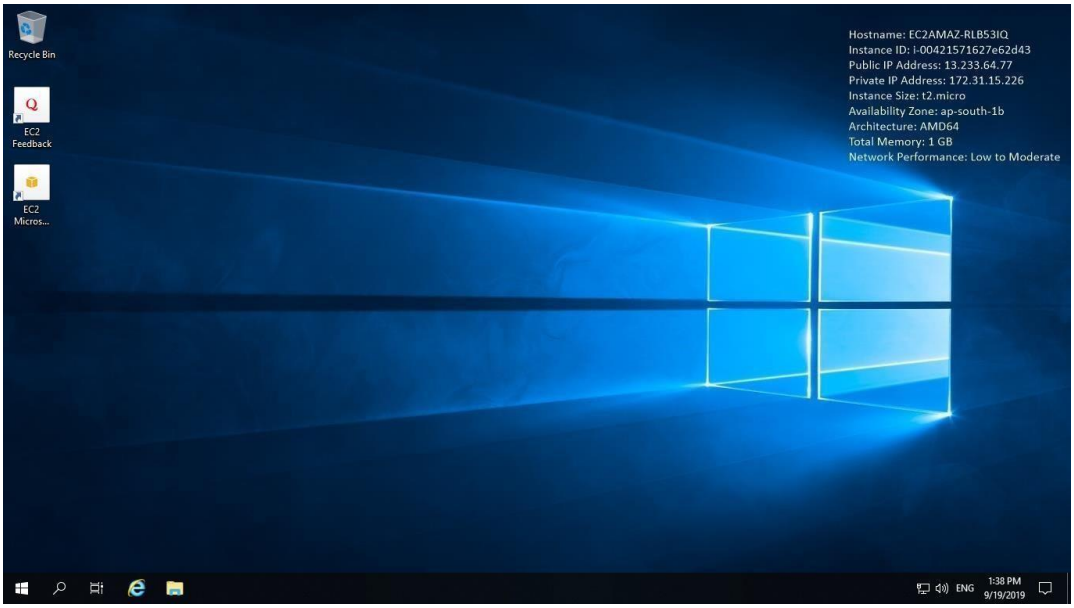
Step 16 :-



Now, open the downloaded remote file,After opening the remote file the window called "Windows Security" will appear paste the password as shown in the above image & click "OK".

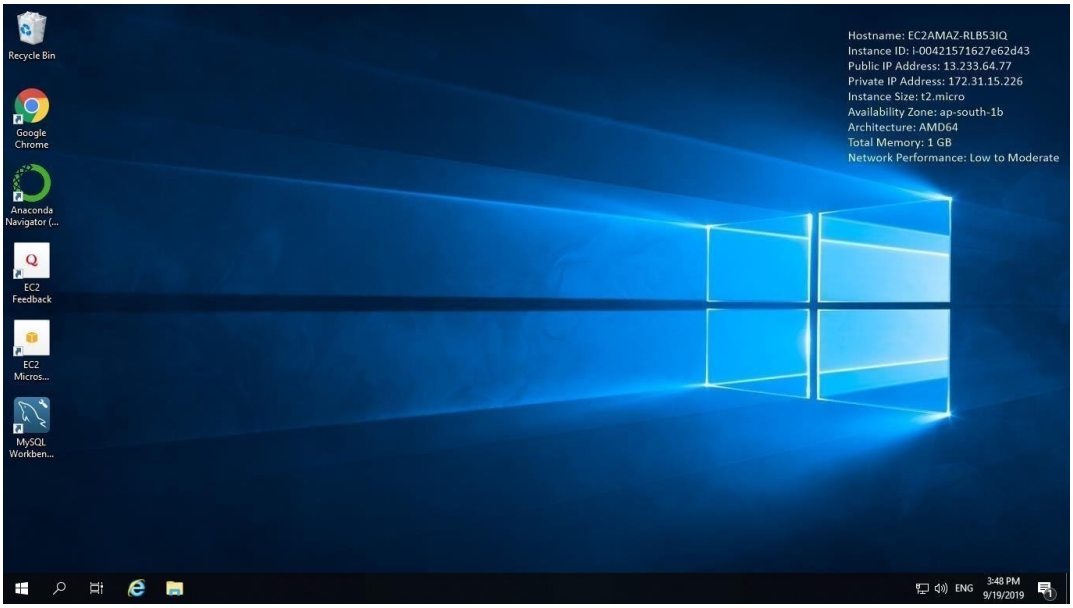
Step 17 :-

(a)



After clicking "OK" the Virtual Machine (VM) will be launched on the Cloud using the AWS EC2 Service.

(b)



After the Virtual Machine(VM) is launched on the cloud , we have installed necessary applications to manage the database i.e. Anaconda Navigator to run python code , MySQL Workbench (To view the Database) & MYSQL Connector (To connect the database with Python).

Ch No. 4 Python Code for Querying the Database

1 IMPORTING NECESSARY LIBRORIES

[1]:
```
import mysql.connector as mysql
```

2 CONNECTING TO THE ROOT INSTANCE

[2]:
```
## connecting to the database using .connect() method
## it takes 3 required parameters .host, .user, .passwd
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator"
)

print(db) # it will print a connection object if everything is fine
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x000001BD9B402B70>
```

3 CREATING DATABASE IN THE ROOT INSTANCE

[3]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator"
)

## creating an instance of .cursor class which is used to execute the .SQL
 ↪statements in .Python
cursor =db.cursor()

## creating a databse called .miniproject
## .execute() method is used to compile a .SQL statement
## below statement is used to create tha .miniproject database
cursor.execute("CREATE DATABASE miniproject")
```

4 DISPLAYING DATABASES IN THE ROOT INSTANCE

[4]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator"
)

cursor = db.cursor()

## executing the statement using 'execute()' method
cursor.execute("SHOW DATABASES")

## 'fetchall()' method fetches all the rows from the last executed statement
## it returns a list of all databases present
databases = cursor.fetchall()

## showing one by one databases
for database in databases:
    print(database)
```

```
('information_schema',)
('miniproject',)
('mysql',)
('performance_schema',)
('sakila',)
('sys',)
('world',)
```

5 SELECTING THE DATABASE IN THE ROOT INSTANCE

[5]:
```
#Selecting Database

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)
print('Database Selected Successfully')
```

```
Database Selected Successfully
```

6 CREATING TABLE IN SELECTED DATABASE

a) TABLE :- "Team" (PARENT TABLE) with PRIMARY KEY

[6]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## creating a table called 'Team' in the 'miniproject' database
cursor.execute("CREATE TABLE Team (Teamid INT(11) NOT NULL AUTO_INCREMENT
 ↪PRIMARY KEY, Date DATE, HomeTeam VARCHAR(25), AwayTeam VARCHAR(25), HomeScore
 ↪INT(11), AwayScore INT(11))")
```

b) TABLE :- "Tournament" (CHILD TABLE)

[7]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## creating a table called 'Tournament' in the 'miniproject' database
cursor.execute("CREATE TABLE Tournament (Tournamentid INT(11) NOT NULL
 ↪AUTO_INCREMENT PRIMARY KEY, Teamid INT(11), Tournament VARCHAR(25), City
 ↪VARCHAR(25), Country VARCHAR(25))")
```

7 DISPLAYING SELECTED TABLES IN THE DATABASE

[8]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()
```

```
## getting all the tables which are present in 'datacamp' database
cursor.execute("SHOW TABLES")

tables = cursor.fetchall() ## it returns list of tables present in the database

## showing all the tables one by one
for table in tables:
    print(table)
```

('team',)
('tournament',)

8 DISPLAY THE COLUMNS FROM THE SELECTED TABLES

a) TABLE :- "Team" (PARENT TABLE)

[9]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## 'DESC table_name' is used to get all columns information
cursor.execute("DESC team")

## it will print all the columns as 'tuples' in a list
print(cursor.fetchall())
```

[('Teamid', 'int(11)', 'NO', 'PRI', None, 'auto_increment'), ('Date', 'date', 'YES', '', None, ''), ('HomeTeam', 'varchar(25)', 'YES', '', None, ''), ('AwayTeam', 'varchar(25)', 'YES', '', None, ''), ('HomeScore', 'int(11)', 'YES', '', None, ''), ('AwayScore', 'int(11)', 'YES', '', None, '')]

b) TABLE :- "Tournament" (CHILD TABLE)

[10]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()
```

```
## DESC table_name  is used to get all columns information
cursor.execute("DESC tournament")

## it will print all the columns as tuples  in a list
print(cursor.fetchall())
```

```
[('Tournamentid', 'int(11)', 'NO', 'PRI', None, 'auto_increment'), ('Teamid',
'int(11)', 'YES', '', None, ''), ('Tournament', 'varchar(25)', 'YES', '', None,
''), ('City', 'varchar(25)', 'YES', '', None, ''), ('Country', 'varchar(25)',
'YES', '', None, '')]
```

9 ADDING FOREIGN KEY TO TABLE "Tournament" & CHANGING THE DEFINATION USING ALTER

a) ON DELETE CASCADE

[11]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

   ##ON DELETE CASCADE is used on the foreign key that allows you to delete data
↪from child tables automatically when you delete the data from the parent table.
cursor.execute("ALTER TABLE Tournament ADD FOREIGN KEY (Teamid) REFERENCES
 ↪Team(Teamid) ON DELETE CASCADE")
```

b) ON UPDATE CASCADE

[12]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

   ##When foreign key is created using on UPDATE CASCADE the referencing rows are
    ↪updated in the child table when the referenced row is updated in the parent
 ↪table which has a primary key.
```

```
cursor.execute("ALTER TABLE Tournament ADD FOREIGN KEY (Teamid) REFERENCES
 ↪Team(Teamid) ON UPDATE CASCADE")
```

10 INSERTING RECORDS IN THE TABLES

a) INSERTING RECORDS IN THE PARENT TABLE

[13]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = """
        INSERT INTO team (date, HomeTeam, AwayTeam, HomeScore, AwayScore) VALUES
 ↪(' 1872-11-30' , ' Scotland' ,' England' ,0,0),

 ↪(' 1873-03-08' , ' England' ,' Scotland' ,4,2),

 ↪(' 1874-03-07' , ' Wales' ,' Denmark' ,2,1),

 ↪(' 1875-03-06' , ' Denmark' ,' Wales' ,2,2),

 ↪(' 1876-03-04' , ' Scotland' ,' Ireland' ,3,0),

 ↪(' 1876-03-25' , ' Ireland' ,' Scotland' ,4,0),

 ↪(' 1877-03-03' , ' England' ,' Wales' ,1,3),

 ↪(' 1877-03-05' , ' Ireland' ,' Scotland' ,0,2),

 ↪(' 1878-03-02' , ' Scotland' ,' Wales' ,7,2),

 ↪(' 1878-03-23' , ' NewZealand' ,' Fiji' ,9,0),

 ↪(' 1879-01-18' , ' Tongo' ,' Australia' ,1,0),

 ↪(' 1879-04-05' , ' NewZealand' ,' Tongo' ,3,0),

   ( 1879-04-07 , South Africa , Zimbabwe  ,2,5),
```

```
 ↪('1880-03-13' , 'India' ,'Pakistan' ,2,1),

 ↪('1880-03-15' , 'SriLanka' ,'Bangladesh' ,4,2),

 ↪('1880-03-27' , 'India' ,'SriLanka' ,5,2),

 ↪('1881-02-26' , 'Morocco' ,'Senegal' ,4,2),

 ↪('1881-03-12' , 'Egypt' ,'Cameroon' ,1,2),

 ↪('1881-03-14' , 'Morocco' ,'Cameroon' ,3,2),

 ↪('1881-03-16' , 'Argentina' ,'Mexico' ,3,1);
"""
## executing the query with values
cursor.execute(query)

 ## to make final output we have to run the 'commit()' method of the database
 ↪object
db.commit()

print(cursor.rowcount, "Records Inserted")
```

```
20 Records Inserted
```

b) INSERTING RECORDS IN THE CHILD TABLE

[14]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query ="""
            INSERT INTO tournament (Teamid, Tournament, City, Country) VALUES
 ↪(1, 'British Championship' ,'Glasgow' ,'Scotland' ),

 ↪(2, 'British Championship' ,'London' ,'England' ),

 ↪(3, 'British Championship' ,'Cardiff' ,'Wales' ),
```

```
 ↪(4, 'British Championship','Copenhagen','Denmark'),

 ↪(5, 'British Championship','Glasgow','Scotland'),

 ↪(6, 'British Championship','Dublin','Ireland'),

 ↪(7, 'British Championship','London','England'),

 ↪(8, 'British Championship','Dublin','Ireland'),

 ↪(9, 'British Championship','Glasgow','Scotland'),

 ↪(10, 'OFC Nations Cup','Wellington','New Zealand'),

 ↪(11, 'OFC Nations Cup','Nukualofa','Tongo'),

 ↪(12, 'OFC Nations Cup','Wellington','New Zealand'),

 ↪(13, 'Friendly','CapeTown','South Africa'),

 ↪(14, 'AsiaCup','Mumbai','India'),

 ↪(15, 'AsiaCup','Colombo','SriLanka'),

 ↪(16, 'AsiaCup','Mumbai','India'),

 ↪(17, 'Africa Nations Cup','Marrakesh','Morocco'),

 ↪(18, 'Africa Nations Cup','Cairo','Egypt'),

 ↪(19, 'Africa Nations Cup','Marrakesh','Morocco'),

 ↪(20, 'Friendly','Buenos Aires','Argentina');


"""


## executing the query with values
cursor.execute(query)

 ## to make final output we have to run the 'commit()' method of the database
 ↪object
db.commit()
```

```
print(cursor.rowcount, "Records Inserted")
```

```
20 Records Inserted
```

11 WHERE CLAUSE

[15]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM team WHERE Teamid = 16"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
(16, datetime.date(1880, 3, 27), 'India', 'SriLanka', 5, 2)
```

12 UPDATING THE RECORD IN THE TABLE

[16]:
```
#Let's update the AwayTeam Name of the 13th record from Zimbabwe to Ghana.

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "UPDATE team SET AwayTeam = 'Ghana' WHERE Teamid = 13"
```

```
## executing the query
cursor.execute(query)

## final step to tell the database that we have changed the table data
db.commit()

print(cursor.rowcount,"Record(s) Affected")
```

```
1 Record(s) Affected
```

13 DELETING THE RECORD FROM THE TABLE

[17]:
```
#Let's delete 20th record from table.

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "DELETE FROM team WHERE Teamid = 20"

## executing the query
cursor.execute(query)

## final step to tell the database that we have changed the table data
db.commit()

print(cursor.rowcount,"Record(s) Deleted")
```

```
1 Record(s) Deleted
```

14 SELECTING THE RECORDS FROM TABLES

[18]:
```
#Now,if you'll notice as we have deleted the 20th record from the table "team"
→which is the parent table in the database the table "tournament" i.e. the
 ↪child table in the database which is associated to the table "team" will also
↪be affected as the 20th record from the child table also will be deleted
↪automatically due to the foreign key that we had assigned to the table
↪"tournament"
```

a) DISPLAYING THE RECORDS IN THE PARENT TABLE i.e.TABLE "Team"

[18]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM team"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
(1, datetime.date(1872, 11, 30), 'Scotland', 'England', 0, 0)
(2, datetime.date(1873, 3, 8), 'England', 'Scotland', 4, 2)
(3, datetime.date(1874, 3, 7), 'Wales', 'Denmark', 2, 1)
(4, datetime.date(1875, 3, 6), 'Denmark', 'Wales', 2, 2)
(5, datetime.date(1876, 3, 4), 'Scotland', 'Ireland', 3, 0)
(6, datetime.date(1876, 3, 25), 'Ireland', 'Scotland', 4, 0)
(7, datetime.date(1877, 3, 3), 'England', 'Wales', 1, 3)
(8, datetime.date(1877, 3, 5), 'Ireland', 'Scotland', 0, 2)
(9, datetime.date(1878, 3, 2), 'Scotland', 'Wales', 7, 2)
(10, datetime.date(1878, 3, 23), 'NewZealand', 'Fiji', 9, 0)
(11, datetime.date(1879, 1, 18), 'Tongo', 'Australia', 1, 0)
(12, datetime.date(1879, 4, 5), 'NewZealand', 'Tongo', 3, 0)
(13, datetime.date(1879, 4, 7), 'South Africa', 'Ghana', 2, 5)
(14, datetime.date(1880, 3, 13), 'India', 'Pakistan', 2, 1)
(15, datetime.date(1880, 3, 15), 'SriLanka', 'Bangladesh', 4, 2)
(16, datetime.date(1880, 3, 27), 'India', 'SriLanka', 5, 2)
(17, datetime.date(1881, 2, 26), 'Morocco', 'Senegal', 4, 2)
(18, datetime.date(1881, 3, 12), 'Egypt', 'Cameroon', 1, 2)
(19, datetime.date(1881, 3, 14), 'Morocco', 'Cameroon', 3, 2)
```

b) DISPLAYING THE RECORDS IN THE CHILD TABLE i.e. TABLE "Tournament"

[19]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM tournament"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
(1, 1, 'British Championship', 'Glasgow', 'Scotland')
(2, 2, 'British Championship', 'London', 'England')
(3, 3, 'British Championship', 'Cardiff', 'Wales')
(4, 4, 'British Championship', 'Copenhagen', 'Denmark')
(5, 5, 'British Championship', 'Glasgow', 'Scotland')
(6, 6, 'British Championship', 'Dublin', 'Ireland')
(7, 7, 'British Championship', 'London', 'England')
(8, 8, 'British Championship', 'Dublin', 'Ireland')
(9, 9, 'British Championship', 'Glasgow', 'Scotland')
(10, 10, 'OFC Nations Cup', 'Wellington', 'New Zealand')
(11, 11, 'OFC Nations Cup', 'Nukualofa', 'Tongo')
(12, 12, 'OFC Nations Cup', 'Wellington', 'New Zealand')
(13, 13, 'Friendly', 'CapeTown', 'South Africa')
(14, 14, 'AsiaCup', 'Mumbai', 'India')
(15, 15, 'AsiaCup', 'Colombo', 'SriLanka')
(16, 16, 'AsiaCup', 'Mumbai', 'India')
(17, 17, 'Africa Nations Cup', 'Marrakesh', 'Morocco')
(18, 18, 'Africa Nations Cup', 'Cairo', 'Egypt')
(19, 19, 'Africa Nations Cup', 'Marrakesh', 'Morocco')
```

15 SORTING THE DATA IN THE TABLES

a) ORDER BY CLAUSE

(i) SORTING THE DATA IN ASCENDING ORDER USING ORDER BY CALUSE & THE COLUMN "Date" FROM THE PARENT TABLE

[20]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM team ORDER BY HomeScore"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
(1, datetime.date(1872, 11, 30), 'Scotland', 'England', 0, 0)
(8, datetime.date(1877, 3, 5), 'Ireland', 'Scotland', 0, 2)
(7, datetime.date(1877, 3, 3), 'England', 'Wales', 1, 3)
(11, datetime.date(1879, 1, 18), 'Tongo', 'Australia', 1, 0)
(18, datetime.date(1881, 3, 12), 'Egypt', 'Cameroon', 1, 2)
(3, datetime.date(1874, 3, 7), 'Wales', 'Denmark', 2, 1)
(4, datetime.date(1875, 3, 6), 'Denmark', 'Wales', 2, 2)
(13, datetime.date(1879, 4, 7), 'South Africa', 'Ghana', 2, 5)
(14, datetime.date(1880, 3, 13), 'India', 'Pakistan', 2, 1)
(5, datetime.date(1876, 3, 4), 'Scotland', 'Ireland', 3, 0)
(12, datetime.date(1879, 4, 5), 'NewZealand', 'Tongo', 3, 0)
(19, datetime.date(1881, 3, 14), 'Morocco', 'Cameroon', 3, 2)
(2, datetime.date(1873, 3, 8), 'England', 'Scotland', 4, 2)
(6, datetime.date(1876, 3, 25), 'Ireland', 'Scotland', 4, 0)
(15, datetime.date(1880, 3, 15), 'SriLanka', 'Bangladesh', 4, 2)
(17, datetime.date(1881, 2, 26), 'Morocco', 'Senegal', 4, 2)
```

(16, datetime.date(1880, 3, 27), 'India', 'SriLanka', 5, 2)
(9, datetime.date(1878, 3, 2), 'Scotland', 'Wales', 7, 2)
(10, datetime.date(1878, 3, 23), 'NewZealand', 'Fiji', 9, 0)

(ii) SORTING THE DATA IN DESCENDING ORDER USING ORDER BY CLAUSE & THE COLUMN "Tournamentid" FROM THE CHILD TABLE

[21]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM tournament ORDER BY Tournamentid DESC"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

(19, 19, 'Africa Nations Cup', 'Marrakesh', 'Morocco')
(18, 18, 'Africa Nations Cup', 'Cairo', 'Egypt')
(17, 17, 'Africa Nations Cup', 'Marrakesh', 'Morocco')
(16, 16, 'AsiaCup', 'Mumbai', 'India')
(15, 15, 'AsiaCup', 'Colombo', 'SriLanka')
(14, 14, 'AsiaCup', 'Mumbai', 'India')
(13, 13, 'Friendly', 'CapeTown', 'South Africa')
(12, 12, 'OFC Nations Cup', 'Wellington', 'New Zealand')
(11, 11, 'OFC Nations Cup', 'Nukualofa', 'Tongo')
(10, 10, 'OFC Nations Cup', 'Wellington', 'New Zealand')
(9, 9, 'British Championship', 'Glasgow', 'Scotland')
(8, 8, 'British Championship', 'Dublin', 'Ireland')
(7, 7, 'British Championship', 'London', 'England')
(6, 6, 'British Championship', 'Dublin', 'Ireland')
(5, 5, 'British Championship', 'Glasgow', 'Scotland')
(4, 4, 'British Championship', 'Copenhagen', 'Denmark')

```
(3, 3, 'British Championship', 'Cardiff', 'Wales')
(2, 2, 'British Championship', 'London', 'England')
(1, 1, 'British Championship', 'Glasgow', 'Scotland')
```

b) GROUPING THE DATA IN THE CHILD TABLEUSING GROUP BY CLAUSE

[22]:
```
##Let's group the table as per the city and country that were selected for the
 ↪tournaments

db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "Select Tournament,City,Country from tournament group by Tournamentid"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
('British Championship', 'Glasgow', 'Scotland')
('British Championship', 'London', 'England')
('British Championship', 'Cardiff', 'Wales')
('British Championship', 'Copenhagen', 'Denmark')
('British Championship', 'Glasgow', 'Scotland')
('British Championship', 'Dublin', 'Ireland')
('British Championship', 'London', 'England')
('British Championship', 'Dublin', 'Ireland')
('British Championship', 'Glasgow', 'Scotland')
('OFC Nations Cup', 'Wellington', 'New Zealand')
('OFC Nations Cup', 'Nukualofa', 'Tongo')
('OFC Nations Cup', 'Wellington', 'New Zealand')
('Friendly', 'CapeTown', 'South Africa')
('AsiaCup', 'Mumbai', 'India')
```

('AsiaCup', 'Colombo', 'SriLanka')
('AsiaCup', 'Mumbai', 'India')
('Africa Nations Cup', 'Marrakesh', 'Morocco')
('Africa Nations Cup', 'Cairo', 'Egypt')
('Africa Nations Cup', 'Marrakesh', 'Morocco')

c) HAVING CLAUSE

[23]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM team HAVING HomeScore > 5 and AwayScore < 5"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

(9, datetime.date(1878, 3, 2), 'Scotland', 'Wales', 7, 2)
(10, datetime.date(1878, 3, 23), 'NewZealand', 'Fiji', 9, 0)

16 JOINS

a) INNER JOIN

[24]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()
```

```
## defining the Query
query = """

             select team.Teamid,
             team.Date,
             team.HomeTeam,
             team.AwayTeam,
             tournament.Tournamentid,
             tournament.Tournament
             from team
             inner join tournament
             on team.Teamid=tournament.Teamid
             where HomeTeam=- Morocco-

        """

## getting records from the table
cursor.execute(query)

## fetching all records from the - cursor- object
records =cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

(17, datetime.date(1881, 2, 26), - Morocco- , - Senegal- , 17, - Africa Nations Cup-)
(19, datetime.date(1881, 3, 14), - Morocco- , - Cameroon- , 19, - Africa Nations Cup-)

b) LEFT JOIN

[25]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = """

             select team.Teamid,
             team.Date,
```

```
            team.HomeTeam,
            team.AwayTeam,
            tournament.Tournamentid,
            tournament.Tournament
            from team
            left join tournament
            on team.Teamid = tournament.Teamid
            where Date between CAST('1875-03-06' AS DATE) and CAST('1881-02-26'
 ↪AS DATE)

        """

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
(4, datetime.date(1875, 3, 6), 'Denmark', 'Wales', 4, 'British Championship')
(5, datetime.date(1876, 3, 4), 'Scotland', 'Ireland', 5, 'British Championship')
(6, datetime.date(1876, 3, 25), 'Ireland', 'Scotland', 6, 'British
Championship')
(7, datetime.date(1877, 3, 3), 'England', 'Wales', 7, 'British Championship')
(8, datetime.date(1877, 3, 5), 'Ireland', 'Scotland', 8, 'British Championship')
(9, datetime.date(1878, 3, 2), 'Scotland', 'Wales', 9, 'British Championship')
(10, datetime.date(1878, 3, 23), 'NewZealand', 'Fiji', 10, 'OFC Nations Cup')
(11, datetime.date(1879, 1, 18), 'Tongo', 'Australia', 11, 'OFC Nations Cup')
(12, datetime.date(1879, 4, 5), 'NewZealand', 'Tongo', 12, 'OFC Nations Cup')
(13, datetime.date(1879, 4, 7), 'South Africa', 'Ghana', 13, 'Friendly')
(14, datetime.date(1880, 3, 13), 'India', 'Pakistan', 14, 'AsiaCup')  (15,
datetime.date(1880, 3, 15), 'SriLanka', 'Bangladesh', 15, 'AsiaCup')  (16,
datetime.date(1880, 3, 27), 'India', 'SriLanka', 16, 'AsiaCup')
(17, datetime.date(1881, 2, 26), 'Morocco', 'Senegal', 17, 'Africa Nations Cup')
```

c) RIGHT JOIN

[26]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)
```

```
cursor = db.cursor()

## defining the Query
query = """

            select team.Teamid,
            team.Date,
            team.HomeTeam,
            team.AwayTeam,
            tournament.Tournamentid,
            tournament.Tournament
            from team
            right join tournament
            on team.Teamid = tournament.Teamid
            where tournament.Tournament='British Championship'

        """

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
(1, datetime.date(1872, 11, 30), 'Scotland', 'England', 1, 'British
Championship')
(2, datetime.date(1873, 3, 8), 'England', 'Scotland', 2, 'British Championship')
(3, datetime.date(1874, 3, 7), 'Wales', 'Denmark', 3, 'British Championship')
(4, datetime.date(1875, 3, 6), 'Denmark', 'Wales', 4, 'British Championship')
(5, datetime.date(1876, 3, 4), 'Scotland', 'Ireland', 5, 'British Championship')
(6, datetime.date(1876, 3, 25), 'Ireland', 'Scotland', 6, 'British
Championship')
 (7, datetime.date(1877, 3, 3), 'England', 'Wales', 7, 'British Championship')
 (8, datetime.date(1877, 3, 5), 'Ireland', 'Scotland', 8, 'British Championship')
 (9, datetime.date(1878, 3, 2), 'Scotland', 'Wales', 9, 'British Championship')
```

d) JOIN

[15]:
```
##In MySQL instead of 'Full Join', 'join' is used. which performs same as full
↪join
```

[27]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = """

             select team.Teamid,
             team.Date,
             team.HomeTeam,
             team.AwayTeam,
             team.HomeScore,
             team.AwayScore,
             tournament.Tournamentid,
             tournament.teamid,
             tournament.tournament,
             tournament.city,
             tournament.country
             from team
             join tournament
             on team.Teamid = tournament.Teamid;

         """

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
(1, datetime.date(1872, 11, 30), 'Scotland', 'England', 0, 0, 1, 1, 'British Championship', 'Glasgow', 'Scotland')
(2, datetime.date(1873, 3, 8), 'England', 'Scotland',  4, 2, 2, 2, 'British Championship', 'London', 'England')
(3, datetime.date(1874, 3, 7), 'Wales', 'Denmark', 2,  1, 3, 3, 'British Championship', 'Cardiff', 'Wales')
(4, datetime.date(1875, 3, 6), 'Denmark', 'Wales', 2,  2, 4, 4, 'British Championship', 'Copenhagen', 'Denmark')
```

```
(5, datetime.date(1876, 3, 4), 'Scotland', 'Ireland', 3, 0, 5, 5, 'British
Championship', 'Glasgow', 'Scotland')
(6, datetime.date(1876, 3, 25), 'Ireland', 'Scotland', 4, 0, 6, 6, 'British
Championship', 'Dublin', 'Ireland')
(7, datetime.date(1877, 3, 3), 'England', 'Wales', 1, 3, 7, 7, 'British
Championship', 'London', 'England')
(8, datetime.date(1877, 3, 5), 'Ireland', 'Scotland', 0, 2, 8, 8, 'British
Championship', 'Dublin', 'Ireland')
(9, datetime.date(1878, 3, 2), 'Scotland', 'Wales', 7, 2, 9, 9, 'British
Championship', 'Glasgow', 'Scotland')
(10, datetime.date(1878, 3, 23), 'NewZealand', 'Fiji',  9, 0, 10, 10, 'OFC
Nations Cup', 'Wellington', 'New Zealand')
(11, datetime.date(1879, 1, 18), 'Tongo', 'Australia',  1, 0, 11, 11, 'OFC
Nations Cup', 'Nukualofa', 'Tongo')
(12, datetime.date(1879, 4, 5), 'NewZealand', 'Tongo', 3, 0, 12, 12, 'OFC
Nations Cup', 'Wellington', 'New Zealand')
(13, datetime.date(1879, 4, 7), South Africa ,  Ghana , 2, 5, 13, 13,
'Friendly', 'CapeTown', 'South Africa')
(14, datetime.date(1880, 3, 13), 'India', 'Pakistan', 2, 1, 14, 14, 'AsiaCup',
'Mumbai', 'India')
(15, datetime.date(1880, 3, 15), 'SriLanka', 'Bangladesh', 4, 2, 15, 15,
'AsiaCup', 'Colombo', 'SriLanka')
(16, datetime.date(1880, 3, 27), 'India', 'SriLanka', 5, 2, 16, 16, 'AsiaCup',
'Mumbai', 'India')
(17, datetime.date(1881, 2, 26), 'Morocco', 'Senegal', 4, 2, 17, 17, 'Africa
Nations Cup', 'Marrakesh', 'Morocco')
(18, datetime.date(1881, 3, 12), 'Egypt', 'Cameroon', 1, 2, 18, 18, 'Africa
Nations Cup', 'Cairo', 'Egypt')
(19, datetime.date(1881, 3, 14), 'Morocco', 'Cameroon', 3, 2, 19, 19, 'Africa
Nations Cup', 'Marrakesh', 'Morocco')
```

17 VIEWS

[20]:
```
##Views are virtual tables; they do not contain the data that is returned. The
↪data is stored in the tables referenced in the SELECT statement. Views improve
↪security of the database by showing only intended data to authorized users.
↪They hide sensitive data.
```

[25]:
```
##We cannot add Duplicate columns while creating a view.
```

a) CREATING A VIEW IN THE DATABASE

[28]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
```

```
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = """
    CREATE VIEW RelationalTable
    AS
              select team.Teamid,
              team.Date,
              team.HomeTeam,
              team.AwayTeam,
              team.HomeScore,
              team.AwayScore,
              tournament.Tournamentid,
              tournament.tournament,
              tournament.city,
              tournament.country
              from team
              join tournament
              on team.Teamid = tournament.Teamid;

        """

## getting records from the table
cursor.execute(query)

print("View Created Successfully")
```

```
View Created Successfully
```

b) EXECUTING THE VIEW

[29]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = "SELECT * FROM RelationalTable"
```

```
## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
(1, datetime.date(1872, 11, 30), 'Scotland', 'England', 0, 0, 1, 'British Championship', 'Glasgow', 'Scotland')
(2, datetime.date(1873, 3, 8), 'England', 'Scotland',  4, 2, 2, 'British Championship', 'London', 'England')
(3, datetime.date(1874, 3, 7), 'Wales', 'Denmark', 2,  1, 3, 'British Championship', 'Cardiff', 'Wales')
(4, datetime.date(1875, 3, 6), 'Denmark', 'Wales', 2,  2, 4, 'British Championship', 'Copenhagen', 'Denmark')
(5, datetime.date(1876, 3, 4), 'Scotland', 'Ireland',  3, 0, 5, 'British Championship', 'Glasgow', 'Scotland')
(6, datetime.date(1876, 3, 25), 'Ireland', 'Scotland', 4, 0, 6, 'British Championship', 'Dublin', 'Ireland')
(7, datetime.date(1877, 3, 3), 'England', 'Wales', 1, 3, 7, 'British Championship', 'London', 'England')
(8, datetime.date(1877, 3, 5), 'Ireland', 'Scotland', 0, 2, 8, 'British Championship', 'Dublin', 'Ireland')
(9, datetime.date(1878, 3, 2), 'Scotland', 'Wales', 7, 2, 9, 'British Championship', 'Glasgow', 'Scotland')
(10, datetime.date(1878, 3, 23), 'NewZealand', 'Fiji',  9, 0, 10, 'OFC Nations Cup', 'Wellington', 'New Zealand')
(11, datetime.date(1879, 1, 18), 'Tongo', 'Australia',  1, 0, 11, 'OFC Nations Cup', 'Nukualofa', 'Tongo')
(12, datetime.date(1879, 4, 5), 'NewZealand', 'Tongo', 3, 0, 12, 'OFC Nations Cup', 'Wellington', 'New Zealand')
(13, datetime.date(1879, 4, 7), 'South Africa', 'Ghana', 2, 5, 13, 'Friendly', 'CapeTown', 'South Africa')
(14, datetime.date(1880, 3, 13), 'India', 'Pakistan', 2, 1, 14, 'AsiaCup', 'Mumbai', 'India')
(15, datetime.date(1880, 3, 15), 'SriLanka', 'Bangladesh', 4, 2, 15, 'AsiaCup', 'Colombo', 'SriLanka')
(16, datetime.date(1880, 3, 27), 'India', 'SriLanka', 5, 2, 16, 'AsiaCup', 'Mumbai', 'India')
(17, datetime.date(1881, 2, 26), 'Morocco', 'Senegal', 4, 2, 17, 'Africa Nations Cup', 'Marrakesh', 'Morocco')
(18, datetime.date(1881, 3, 12), 'Egypt', 'Cameroon', 1, 2, 18, 'Africa Nations Cup', 'Cairo', 'Egypt')
(19, datetime.date(1881, 3, 14), 'Morocco', 'Cameroon', 3, 2, 19, 'Africa
```

Nations Cup' , 'Marrakesh' , 'Morocco')

18 STORED PROCEDURE

[8]:
```
##Stored procedures help reduce the traffic between application and database
↪server because instead of sending multiple lengthy SQL statements. Stored
↪procedures are reusable and transparent to any applications.
```

a) CREATING A STORED PROCEDURE IN THE DATABASE

[9]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)

cursor = db.cursor()

## defining the Query
query = """
    create procedure NumberofGoals()
    BEGIN
    select sum(HomeScore) as HomeScore,
    sum(AwayScore) as AwayScore
    from team;
    END

        """

## getting records from the table
cursor.execute(query)

print('Store Procedure Created Successfully')
```

```
Store Procedure Created Successfully
```

b) EXECUTING THE STORED PROCEDURE

[10]:
```
db = mysql.connect(
    host = "localhost",
    user = "root",
    passwd = "Dvadministrator",
    database = "miniproject"
)
```

```
cursor = db.cursor()

## defining the Query
query = "CALL NumberofGoals()"

## getting records from the table
cursor.execute(query)

## fetching all records from the 'cursor' object
records = cursor.fetchall()

## Showing the  data
for record in records:
    print(record)
```

```
(Decimal('57'), Decimal('28'))
```

[ ]:

Ch No. 5 Conclusion

This Project "Manipulating Database in AWS Cloud" is prepared to meet the requirements of Small as well as Large Organisations to manage their databases & their data. Implementing this technique the data of the organisations can be maintained in a very proper manner.

As shown in the project the AWS platform can be highly secure and flexible for any organisation to implement it in their day-to-day jobs. Tools such as MySQL workbench, Anaconda,etc can be easily installed in the AWS cloud environment as shown in the project.

Thus, this technique will by far be the complete solution for the organisations who are willing to implement it.