**A Project Report**

**On**

***COVID-19 DATA ANALYSIS***

Submitted by:

APOLO RANA

Under the guidance of

**Mr. Prabhat Das**

(PGT-IP)



**ARMY PUBLIC SCHOOL JORHAT**

**MES GATE ROWRIAH JORHAT**

**Assam-785004**

CERTIFICATE BY PRINCIPAL

This is to certify that this project report entitled “**Covid-19 Data Analysis**” submitted by **Apolo Rana** to Army Public School Jorhat has been examined and evaluated.

The report has been prepared as per the regulations of CBSE and qualifies to be accepted.

Date:

Place:

Mrs. Firdausi Sultana Hazarika

(Principal)

Army Public School Jorhat

CERTIFICATE BY EXAMINERS

This is to certify that this project report entitled“**Covid-19 Data Analysis**” is the bona fide work of who carried out the project work under my supervision and guidance.

To the best of my knowledge, the matter embodied in the report has not been submitted to any other institute for the award of any other degree.

Date:

Place:

Mr. Prabhat Das

(External Examiner) (Internal Examiner)

ACKNOWLEDGEMENT

I take this opportunity to extend my heart full gratitude to Army Public School Jorhat for providing me the opportunity.

I am highly grateful to my guide Mr. Prabhat Das, PGT-IP, Army Public School Jorhat for giving us the opportunity to work under him and providing us an ample guidance and support through the project.

Lastly, I would also like to thank the authors whose publications guided us regarding our project.

DECLARATION

I admit that this report is of my own work and all the sources of the information used in this report have fully acknowledged.

I hereby declare that the dissertation work entitled “**Covid-19 Data Analysis**” submitted to the Army Public School Jorhat, is prepared by me and was not submitted to any other institution for award of any other degree.

Date:

Place:

Signature

Abstract

In times of natural calamities like the COVID-19 pandemic it is crucial to have a bird’s eye view of the on ground medical condition. Especially in a country like India which is one of the most populated countries in the world, the absence of an accurate data regarding the total no of positive cases, no of active cases, etc can lead to severe problem as it would lead to loopholes in vaccine distribution and medical facilities being denied to a large section of the population resulting in a huge loss of life.

In such cases the availability of accurate data regarding the on ground medical situation can help government and health care services to plan their future actions and device new strategies to culminate the problems prevailing on ground, thus reducing the no of fatalities and ensuring that correct and timely medical treatment and adequate medication reaches each and every citizen of the country.

The existing program provides the user with the on ground medical condition of the 28 states and 9 union territories by interacting with the user through a menu based interface providing three types of graphically analysed data and nine types of numerically estimated data such as total positive cases, total number of deaths, etc it also helps the new users to register as a permanent user through a simple yet effective method of registration.

Contents

* Tools and Library 1
* Introduction 3
* Project overview 4
* Source code 13
* Commands in SQL 19
* Conclusion and Future work 20
* References 21 List Of Figures
* Fig 1. User information table 4
* Fig 2. Login options 4
* Fig 3.1. Login procedure for existing users 5
* Fig 3.2. Login procedure for new users 5
* Fig 4. Types of state/UT wise data(main menu option 1) 6
* Fig 5. Output for option 1 6
* Fig 6. Output for option 2 7
* Fig 7. Output for option 3 7
* Fig 8. Output for option 4 8
* Fig 9. Output for option 5 8
* Fig 10. Output for option 6 9
* Fig 11. Output for option 7 9
* Fig 12. Output for option 8 10
* Fig 13. Output for option 9 10
* Fig 14. Types of graphical representations( main menu option 2) 11
* Fig 15. Output for option 1(states vs cured graph) 11
* Fig 16. Output for option 2(states vs death graph) 12
* Fig 17. Output for option 3(cured-death trend graph) 12

Tools and Libraries Used

MySQL

MySQL is the world's most used relational [database management system](https://www.limswiki.org/index.php?title=Database_management_system&action=edit&redlink=1) (RDBMS) that runs as a server providing multi-user access to a number of databases. It is named after developer Michael Widenius' daughter. The phrase "SQL" stands for "structured query language."

[Open source projects](https://www.limswiki.org/index.php/Category:Open-source_software) that require a full-featured database management system often use MySQL. Applications which use MySQL databases include:

[TYPO3](https://www.limswiki.org/index.php/TYPO3), [Joomla](https://www.limswiki.org/index.php/Joomla), [WordPress](https://www.limswiki.org/index.php/WordPress), [phpBB](https://www.limswiki.org/index.php/PhpBB), [Drupal](https://www.limswiki.org/index.php/Drupal), and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale Web products, including Wikipedia, Google (though not for searches), Facebook, and Twitter.

The free open-source version of MySQL is commonly referred to as the MySQL Community Edition. For commercial use, several paid editions are also available, offering additional functionality.[1]

PyCharm

PyCharm is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), specifically for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) language. It is developed by the [Czech](https://en.wikipedia.org/wiki/Czech_Republic) company [JetBrains](https://en.wikipedia.org/wiki/JetBrains).[[5]](https://en.wikipedia.org/wiki/PyCharm#cite_note-5) It provides code analysis, a graphical debugger, an integrated unit tester, integration with [version control systems](https://en.wikipedia.org/wiki/Revision_control) (VCSes), and supports web development with [Django](https://en.wikipedia.org/wiki/Django_(web_framework)) as well as [data science](https://en.wikipedia.org/wiki/Data_science) with [Anaconda](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)).

PyCharm is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), with [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux) versions. The Community Edition is released under the [Apache License](https://en.wikipedia.org/wiki/Apache_License), and there is also Professional Edition with extra features – released under a [proprietary license](https://en.wikipedia.org/wiki/Proprietary_software). [2]

Pandas

In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), pandas is a [software library](https://en.wikipedia.org/wiki/Software_library) written for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and [time series](https://en.wikipedia.org/wiki/Time_series). It is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [three-clause BSD license](https://en.wikipedia.org/wiki/3-clause_BSD_license). The name is derived from the term "[panel data](https://en.wikipedia.org/wiki/Panel_data)", an [econometrics](https://en.wikipedia.org/wiki/Econometrics) term for data sets that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. [Wes McKinney](https://en.wikipedia.org/wiki/Wes_McKinney) started building what would become pandas at [AQR Capital](https://en.wikipedia.org/wiki/AQR_Capital) while he was a researcher there from 2007 to 2010. [3]

Numpy

NumPy (pronounced [/ˈnʌmpaɪ/](https://en.wikipedia.org/wiki/Help:IPA/English) ([NUM-py](https://en.wikipedia.org/wiki/Help:Pronunciation_respelling_key)) or sometimes [/ˈnʌmpi/](https://en.wikipedia.org/wiki/Help:IPA/English) ([NUM-pee](https://en.wikipedia.org/wiki/Help:Pronunciation_respelling_key))) is a library for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)), adding support for large, multi-dimensional [arrays](https://en.wikipedia.org/wiki/Array_data_structure) and [matrices](https://en.wikipedia.org/wiki/Matrix_(math)), along with a large collection of [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [mathematical](https://en.wikipedia.org/wiki/Mathematics) [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by [Jim Hugunin](https://en.wikipedia.org/wiki/Jim_Hugunin) with contributions from several other developers. In 2005, [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is [open-source software](https://en.wikipedia.org/wiki/Open-source_software) and has many contributors. [[4]](https://en.wikipedia.org/wiki/NumPy#cite_note-Nature-5)

Matplotlib

Matplotlib is a [plotting](https://en.wikipedia.org/wiki/Plotter) [library](https://en.wikipedia.org/wiki/Library_(computer_science)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language and its numerical mathematics extension [NumPy](https://en.wikipedia.org/wiki/NumPy). It provides an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [API](https://en.wikipedia.org/wiki/API) for embedding plots into applications using general-purpose [GUI toolkits](https://en.wikipedia.org/wiki/GUI_toolkit) like [Tkinter](https://en.wikipedia.org/wiki/Tkinter), [wxPython](https://en.wikipedia.org/wiki/WxPython), [Qt](https://en.wikipedia.org/wiki/Qt_(software)), or [GTK+](https://en.wikipedia.org/wiki/GTK%2B). There is also a [procedural](https://en.wikipedia.org/wiki/Procedural_programming) "pylab" interface based on a [state machine](https://en.wikipedia.org/wiki/State_machine) (like [OpenGL](https://en.wikipedia.org/wiki/OpenGL)), designed to closely resemble that of [MATLAB](https://en.wikipedia.org/wiki/MATLAB), though its use is discouraged. [SciPy](https://en.wikipedia.org/wiki/SciPy) makes use of Matplotlib.

Matplotlib was originally written by [John D. Hunter](https://en.wikipedia.org/wiki/John_D._Hunter). Since then it has an active development community and is distributed under a [BSD-style license](https://en.wikipedia.org/wiki/BSD_licenses). Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012 and was further joined by Thomas Caswell.[[5]](https://en.wikipedia.org/wiki/Matplotlib#cite_note-6)

Introduction

The theme of my project is ‘ZOMATO DATA ANALYSIS’. This project is fine thought to make complex procedure of Zomato data in an easy manner which is systematic, modular designed, selective menu based user display. The modular design and constructed is very much user oriented in which user can easily understand the tools and can do edit of his own choice. The system is not any though more and does not possesses many application but it is made by focusing on maintaining record employee’s action in a computerized rather than time taking and cumbersome manual system.

The project software application that can easily handle by minimum educated and simple computer knowledge person without any option of error

Project Overview

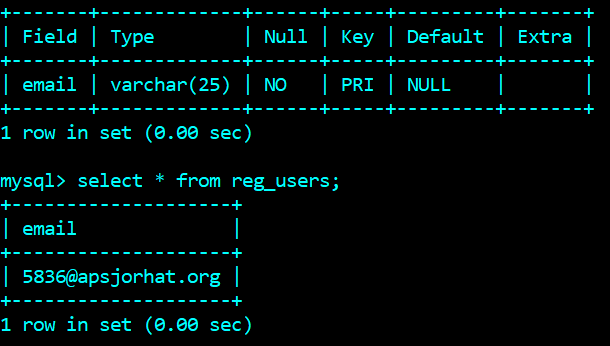


Figure 1**:**All the information regarding the users are stored in this table

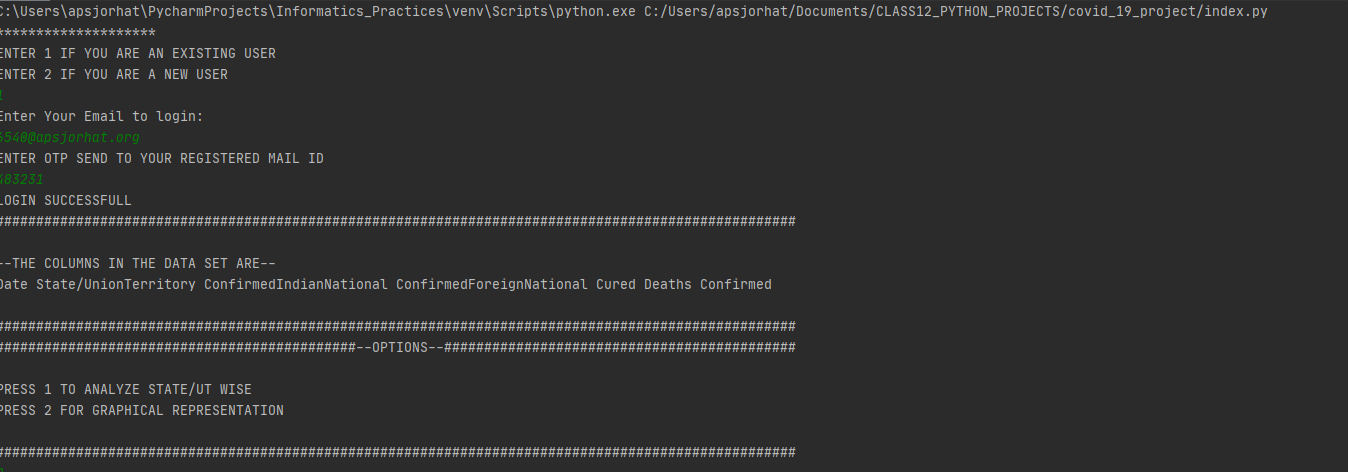


Figure 2: When the home page is opened two login options are presented to the us

.

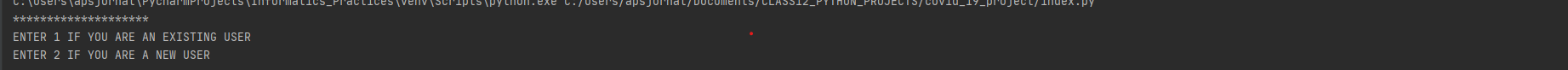


Figure 3.1: When an existing user chooses option 1 the program asks for his/her email address, on entering the email-id the user receives an OTP from the program entering which completes the login procedure and shows two options

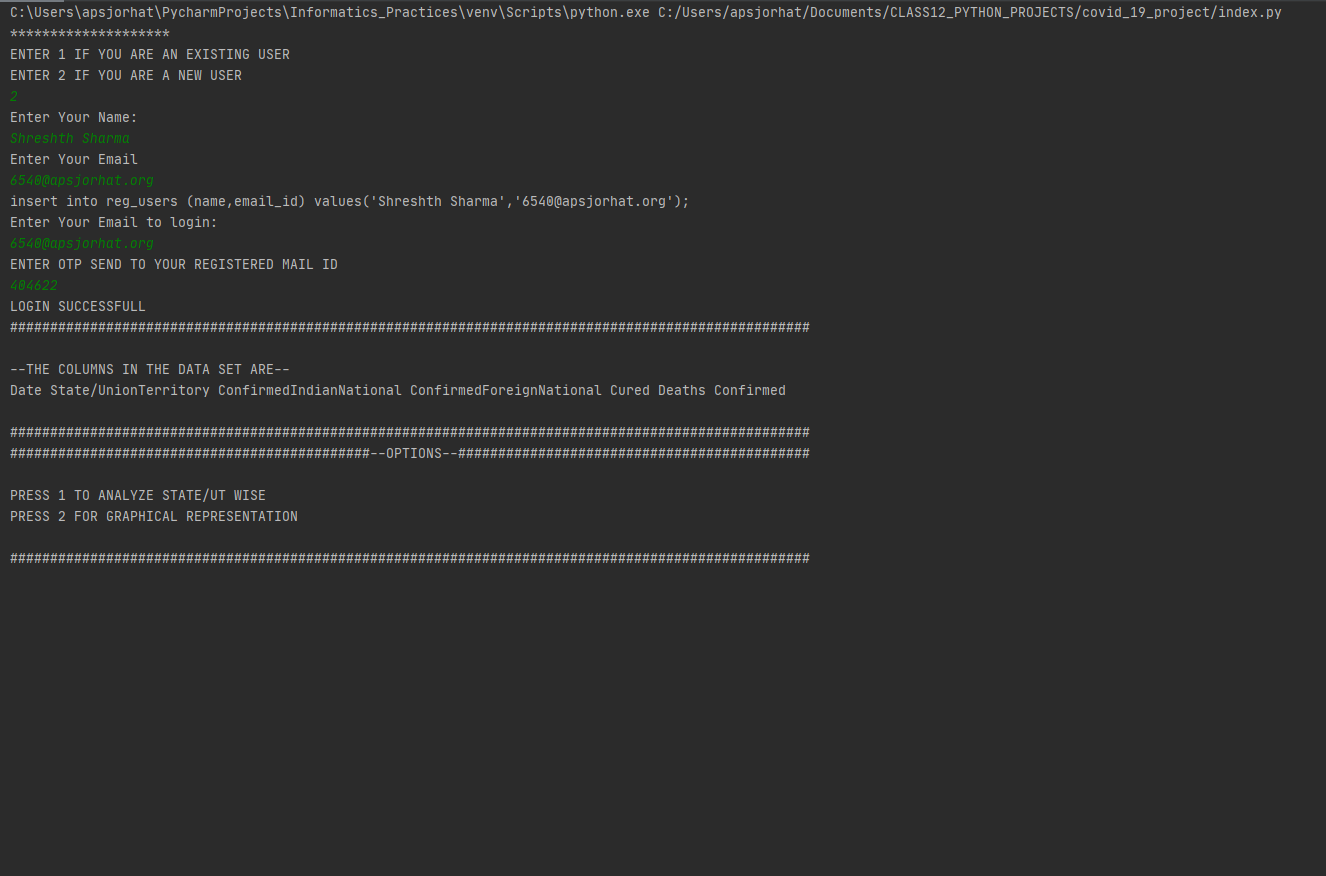


Figure 3.2: Login procedure for new user

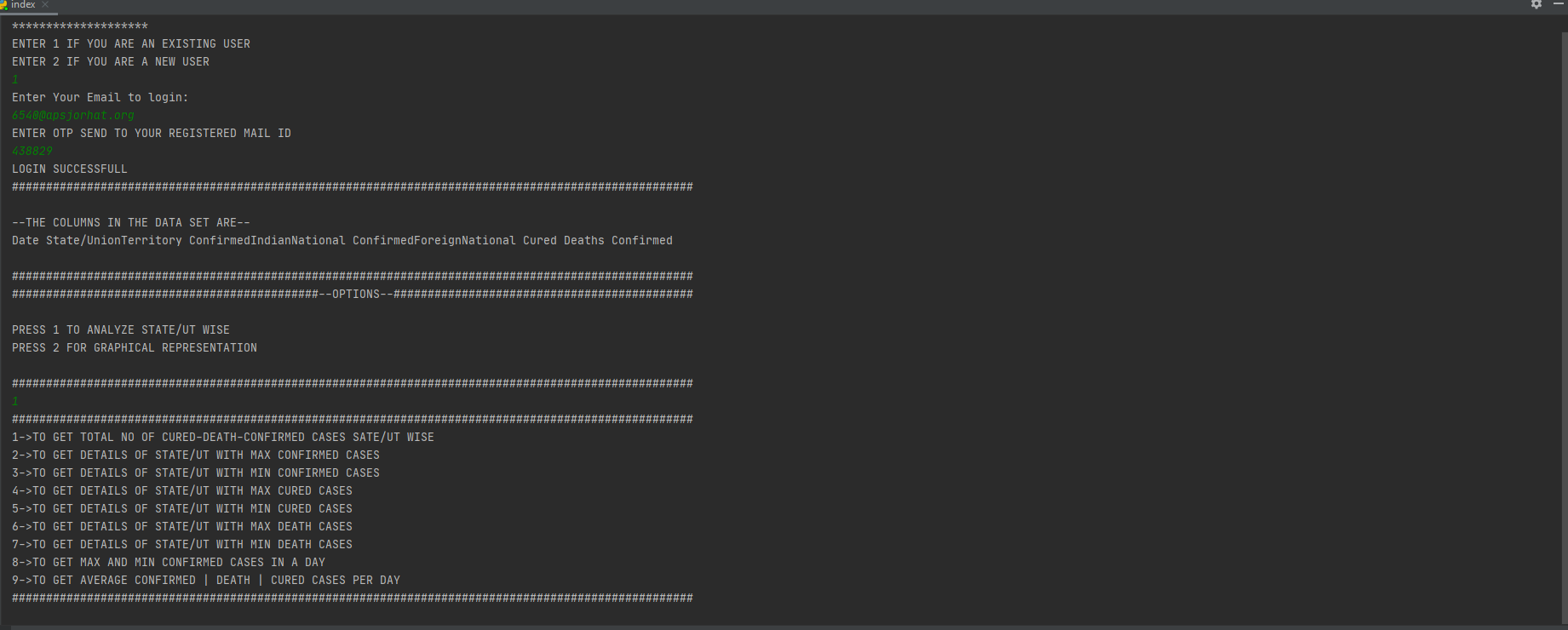


Figure 4: On entering option 1 we get 9 types of state/UT wise analysed data

The data can be retrieved by entering the options as given in the list , the output can be received as follows

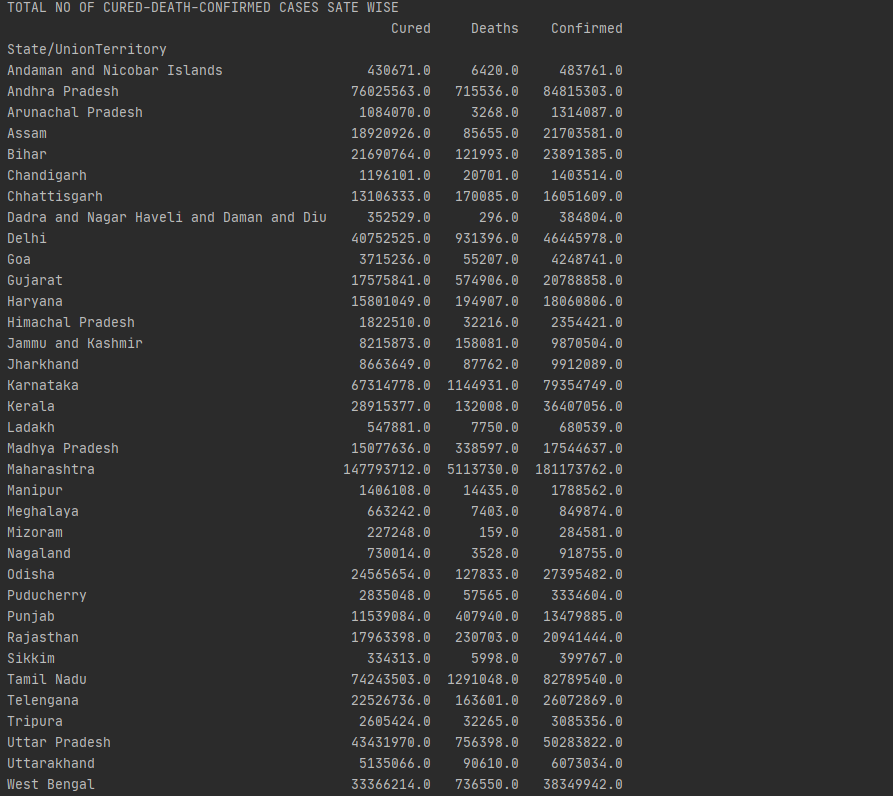


Figure 5: Output for option 1(Cured-death-confirmed cases)

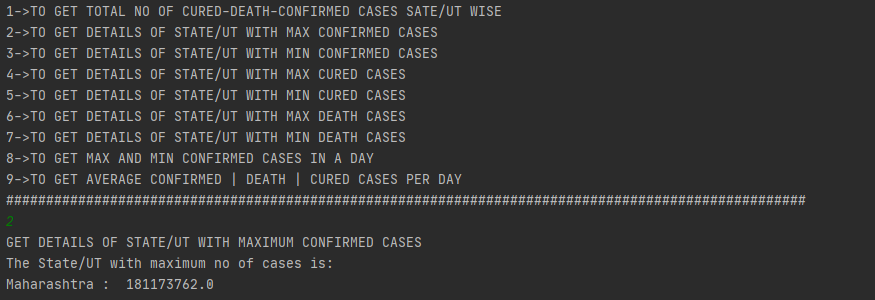


Figure 6: Output for option 2(State/ut with max confirmed cases)

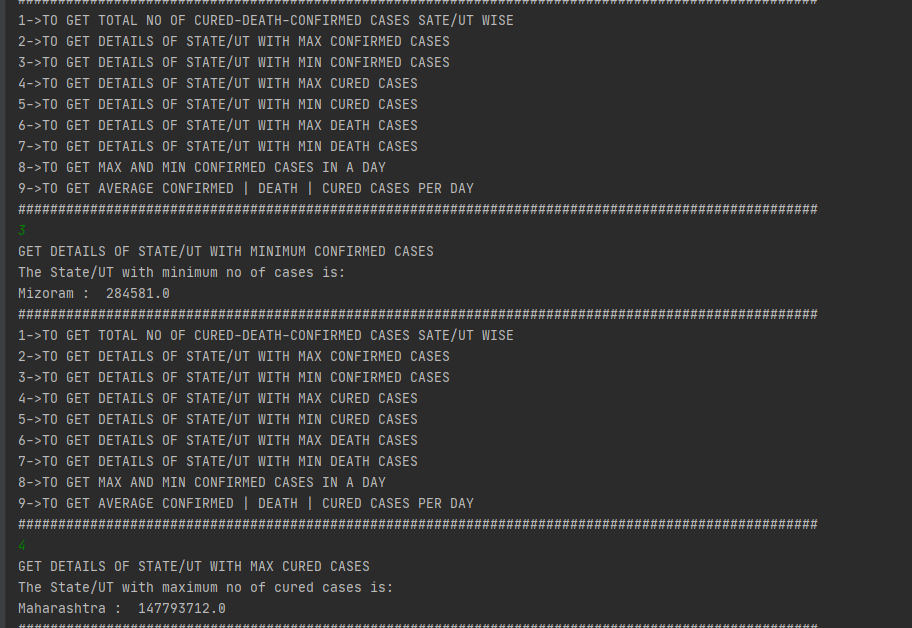


Figure 7: output for option 3((State/UT with minimum confirmed cases)

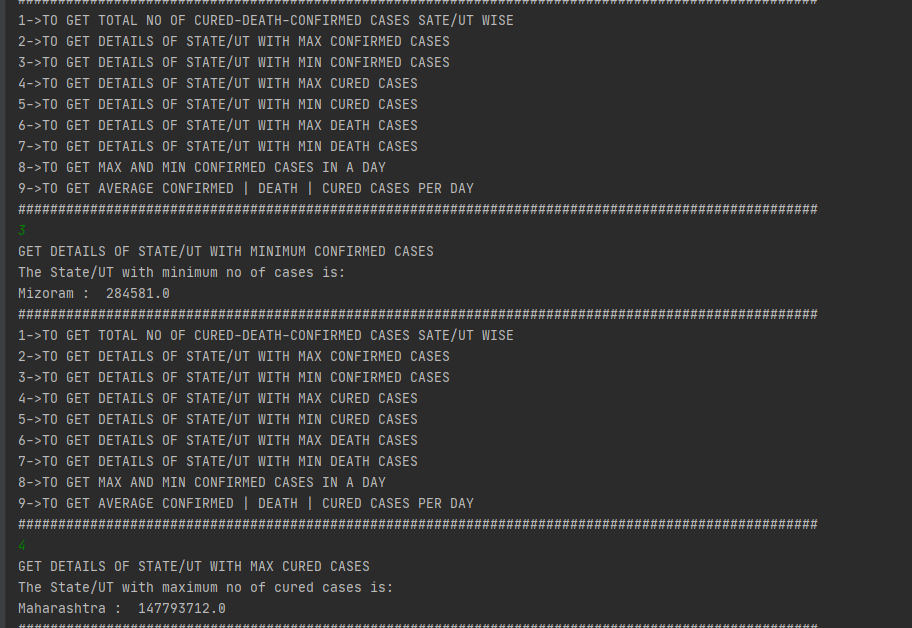


Figure 8 Output for option 4: (State/UT with maximum cured cases)

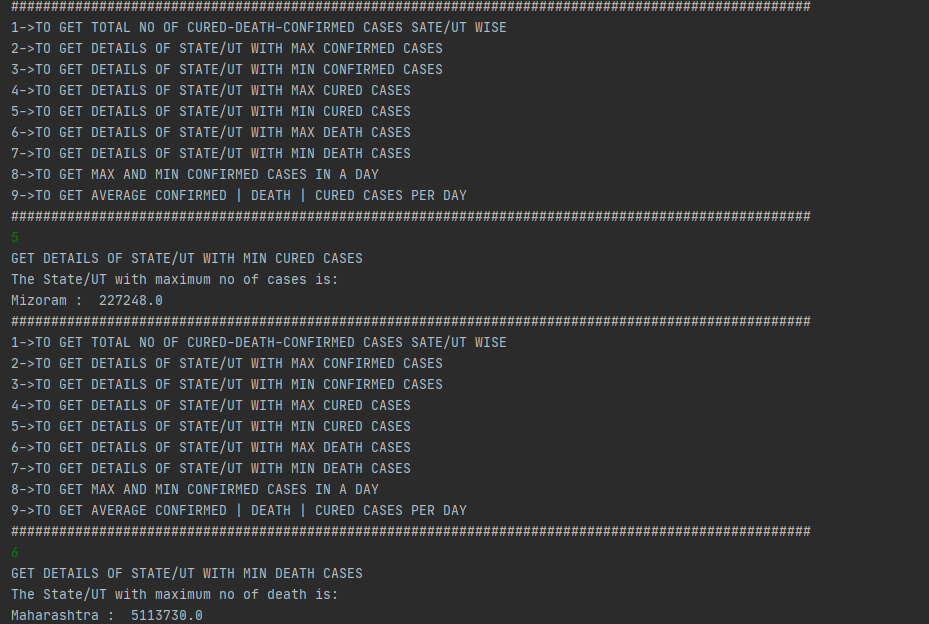


Figure 9: Output for option 5(States/UT with min)

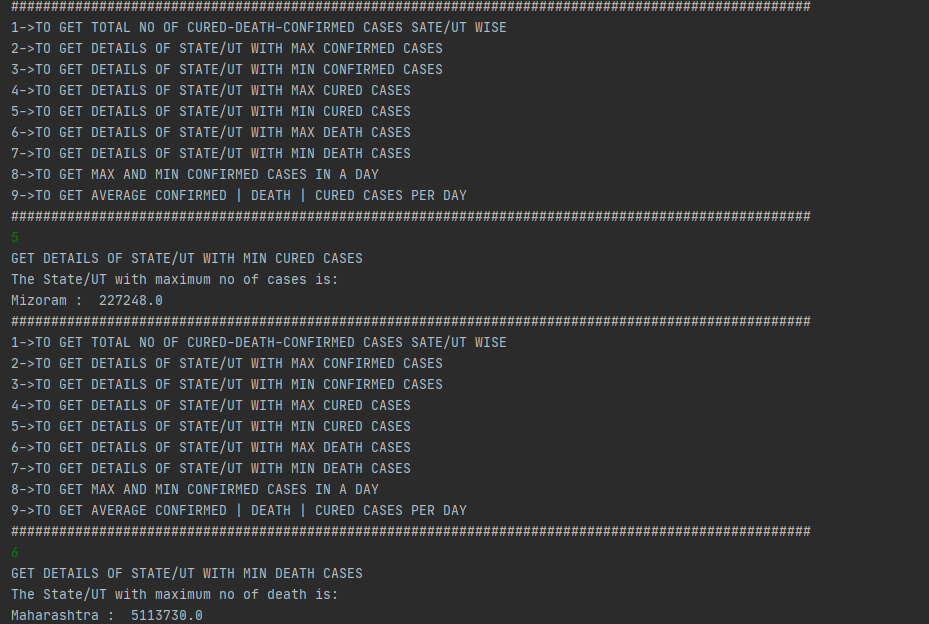


Figure 10: Output for option 6(State/ut with maximum death cases)

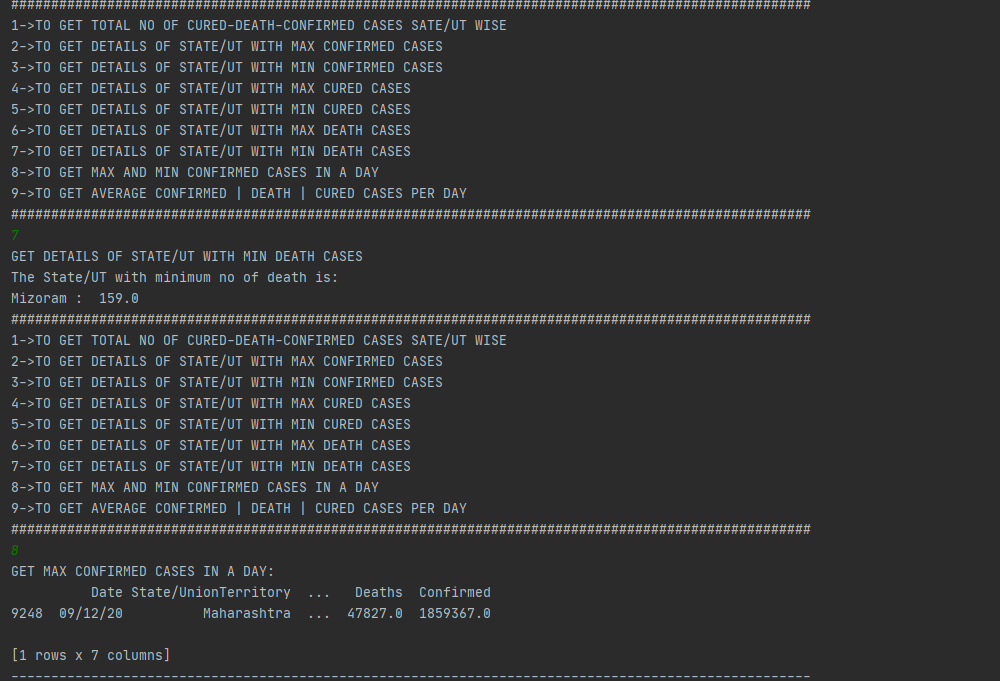


Figure 1: Output for option 7(State/UT with minimum death cases)

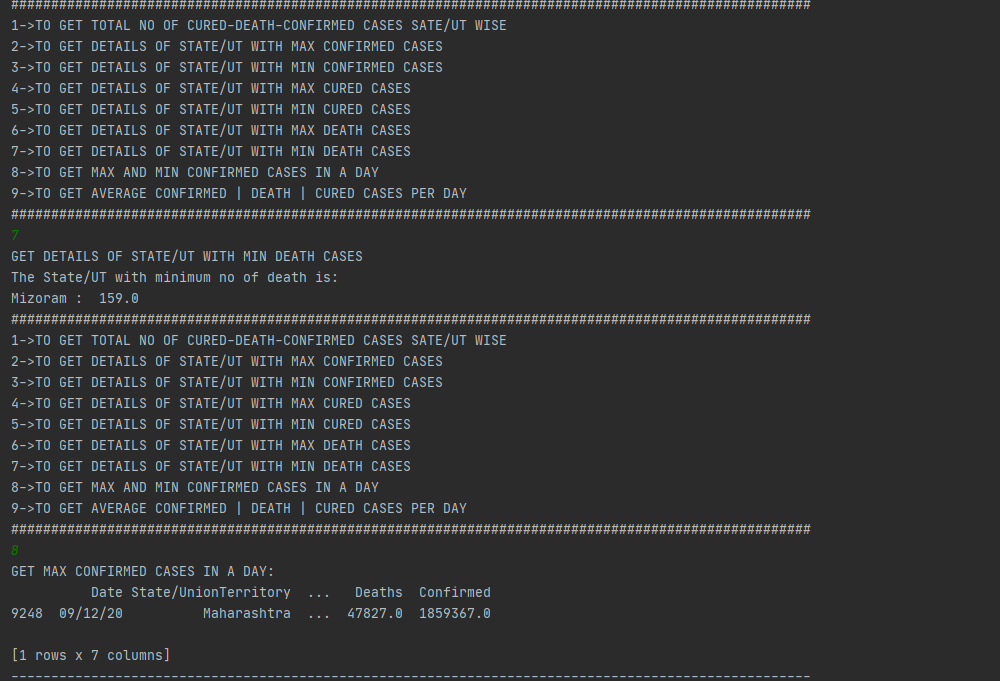


Figure 12: Output for option 8(Max and min confirmed cases)

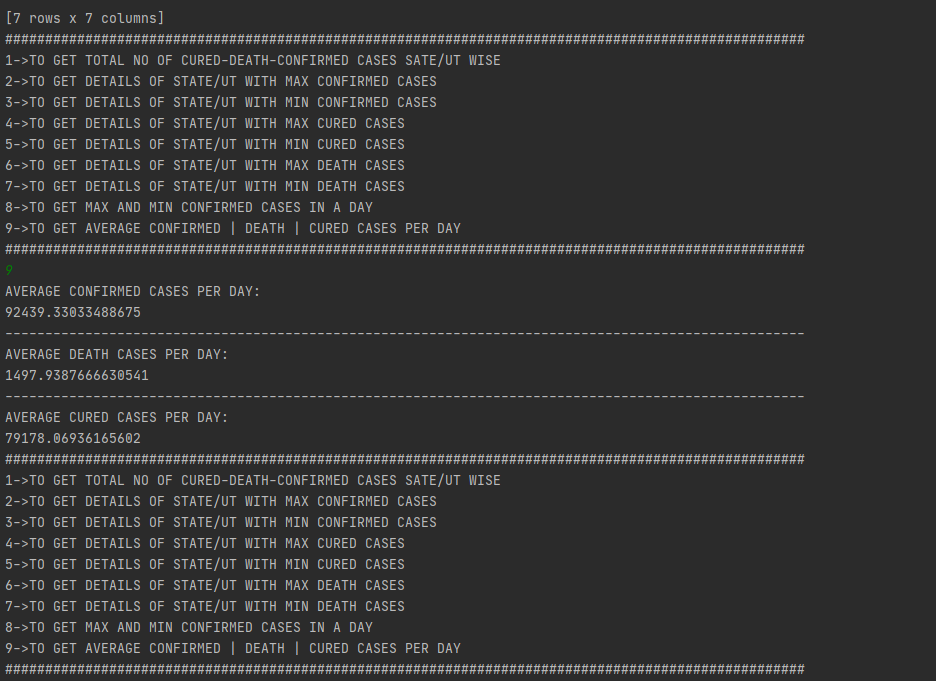


Figure 13: Output for option 9(Average confirmed-death-cured cases per day)

Entering option 2 from the before mentioned main menu we get 3 types of graphical representations as shown below:

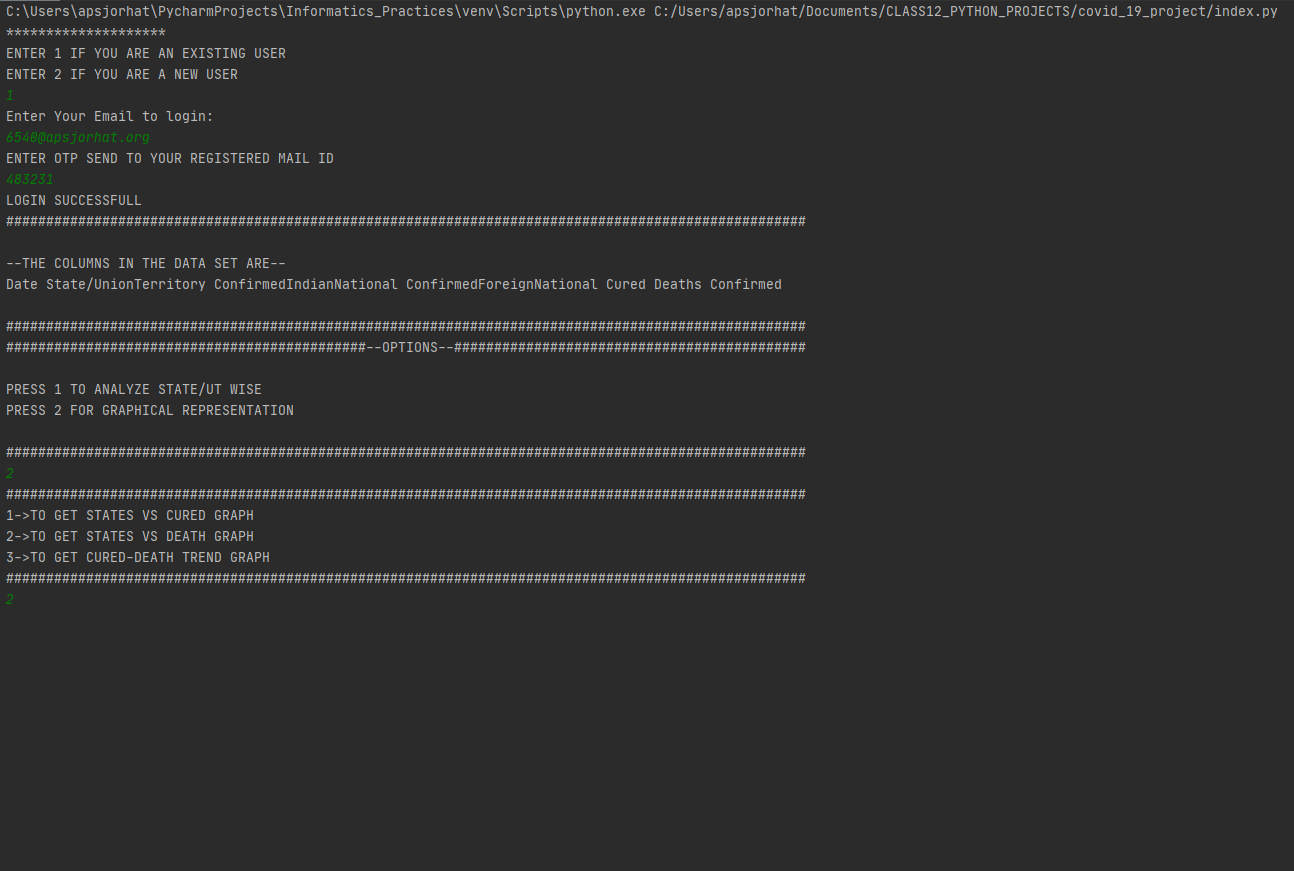


Figure 14: (Main menu option 2)

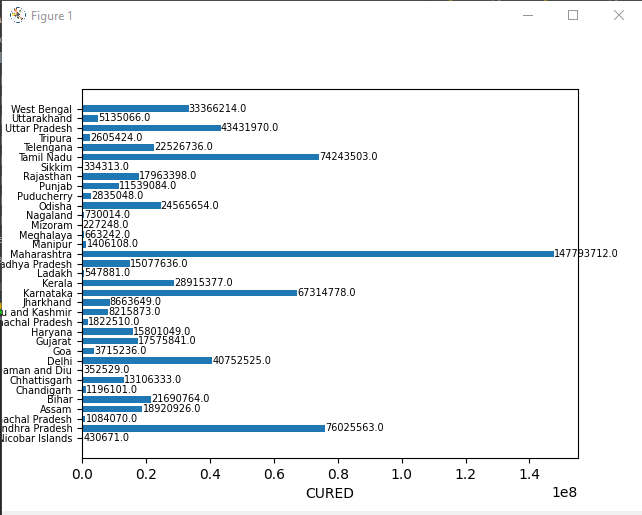


Figure 15: Output for option 1(States vs cured graph)

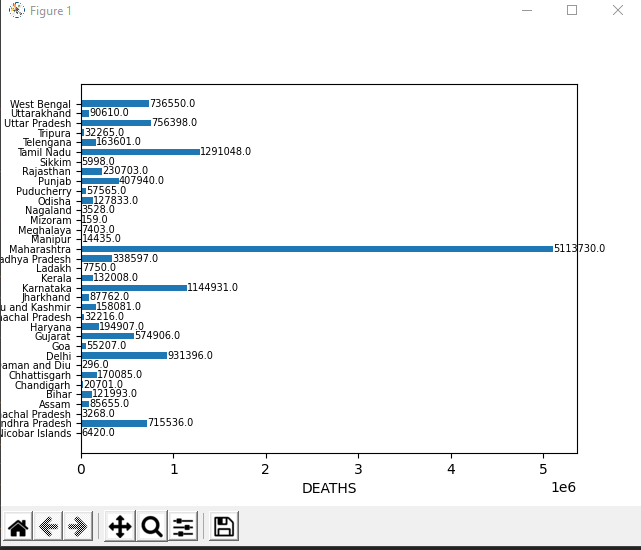


Figure 16: Output for option 2(States vs deaths)

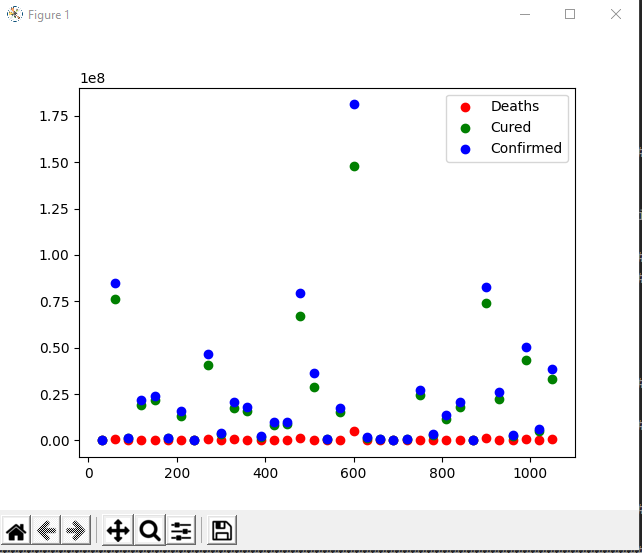


Figure 17: Output for option 3(Cured-death trend graph)

**Source Code**

**Source Code for Index Page**

import covid\_19\_project.user\_auth as u\_auth  
import covid\_19\_project.smtp as smtp  
import random  
print import covid\_19\_project.user\_reg as u\_reg  
 ("\*"\*20)  
print("ENTER 1 IF YOU ARE AN EXISTING USER")  
print("ENTER 2 IF YOU ARE A NEW USER")  
selection = int(input())  
def login():  
 print("Enter Your Email to login:")  
 email = input()  
 authentication = u\_auth.user\_auth(email, count=0)  
 if authentication == 1:  
 otp = int(random.randint(100000, 999999))  
 smtp.otp\_sender(email, otp)  
 print("ENTER OTP SEND TO YOUR REGISTERED MAIL ID")  
 input\_otp = int(input())  
 if input\_otp == otp:  
 print("LOGIN SUCCESSFULL")  
 import covid\_19\_project.covid\_data\_analysis as cda  
 else:  
 print("INVALID OTP")  
 login()  
 else:  
 print("INVALID EMAIL !")  
 login()  
  
def reg():  
 print("Enter Your Name:")  
 name = input()  
 print("Enter Your Email")  
 email = input()  
 u\_reg.user\_reg(name, email)  
 login()  
if selection==1:  
 login()  
elif selection==2:  
 reg()

**Source Code For User Authentication**

import covid\_19\_project.mysql\_connector as mc  
def user\_auth(email,count):  
 fetch\_query = "select \* from reg\_users;"  
 mc.cursor.execute(fetch\_query)  
 for i in mc.cursor:  
 if i[2] == email:  
 count = 1  
 break  
 return count

**Source Code For User Regentration**

import covid\_19\_project.mysql\_connector as mc  
def user\_reg(name,email):  
 query = "insert into reg\_users (name,email\_id) values('{}','{}');".format(name, email)  
 print(query)  
 mc.cursor.execute(query)  
 mc.dbc.commit()

**Source Code For Data Analysis**

import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
data=pd.read\_csv("covid\_19\_india.csv")  
df=pd.DataFrame(data)  
col=df.columns  
grouped\_data=df.groupby('State/UnionTerritory')  
max\_cnf = grouped\_data['Confirmed'].sum().max()  
min\_cnf = grouped\_data['Confirmed'].sum().min()  
max\_cured = grouped\_data['Cured'].sum().max()  
min\_cured = grouped\_data['Cured'].sum().min()  
max\_death = grouped\_data['Deaths'].sum().max()  
min\_death = grouped\_data['Deaths'].sum().min()  
name\_sum\_cnf = dict(zip(list(grouped\_data.groups.keys()),list(grouped\_data['Confirmed'].sum())))  
name\_sum\_cured = dict(zip(list(grouped\_data.groups.keys()),list(grouped\_data['Cured'].sum())))  
name\_sum\_death = dict(zip(list(grouped\_data.groups.keys()),list(grouped\_data['Deaths'].sum())))  
def state\_wise():  
 print("#"\*100)  
 print("1->" + "TO GET TOTAL NO OF CURED-DEATH-CONFIRMED CASES SATE/UT WISE")  
 print("2->" + "TO GET DETAILS OF STATE/UT WITH MAX CONFIRMED CASES")  
 print("3->" + "TO GET DETAILS OF STATE/UT WITH MIN CONFIRMED CASES")  
 print("4->" + "TO GET DETAILS OF STATE/UT WITH MAX CURED CASES")  
 print("5->" + "TO GET DETAILS OF STATE/UT WITH MIN CURED CASES")  
 print("6->" + "TO GET DETAILS OF STATE/UT WITH MAX DEATH CASES")  
 print("7->" + "TO GET DETAILS OF STATE/UT WITH MIN DEATH CASES")  
 print("8->" + "TO GET MAX AND MIN CONFIRMED CASES IN A DAY")  
 print("9->" + "TO GET AVERAGE CONFIRMED | DEATH | CURED CASES PER DAY")  
 print("#" \* 100)  
 input\_1 = int(input())  
 if input\_1 == 1:  
 print("TOTAL NO OF CURED-DEATH-CONFIRMED CASES SATE WISE")  
 print(grouped\_data.sum())  
 state\_wise()  
 if input\_1 == 2:  
 print("GET DETAILS OF STATE/UT WITH MAXIMUM CONFIRMED CASES")  
 for i in name\_sum\_cnf:  
 if name\_sum\_cnf[i] == max\_cnf:  
 print("The State/UT with maximum no of cases is:")  
 print(i, ": ", name\_sum\_cnf[i])  
 state\_wise()  
 if input\_1 == 3:  
 print("GET DETAILS OF STATE/UT WITH MINIMUM CONFIRMED CASES")  
 for i in name\_sum\_cnf:  
 if name\_sum\_cnf[i] == min\_cnf:  
 print("The State/UT with minimum no of cases is:")  
 print(i, ": ", name\_sum\_cnf[i])  
 state\_wise()  
 if input\_1 == 4:  
 print("GET DETAILS OF STATE/UT WITH MAX CURED CASES")  
 for i in name\_sum\_cured:  
 if name\_sum\_cured[i] == max\_cured:  
 print("The State/UT with maximum no of cured cases is:")  
 print(i, ": ", name\_sum\_cured[i])  
 state\_wise()  
 if input\_1 == 5:  
 print("GET DETAILS OF STATE/UT WITH MIN CURED CASES")  
 for i in name\_sum\_cured:  
 if name\_sum\_cured[i] == min\_cured:  
 print("The State/UT with maximum no of cases is:")  
 print(i, ": ", name\_sum\_cured[i])  
 state\_wise()  
  
 if input\_1 == 6:  
 print("GET DETAILS OF STATE/UT WITH MIN DEATH CASES")  
 for i in name\_sum\_death:  
 if name\_sum\_death[i] == max\_death:  
 print("The State/UT with maximum no of death is:")  
 print(i, ": ", name\_sum\_death[i])  
 state\_wise()  
 if input\_1 == 7:  
 print("GET DETAILS OF STATE/UT WITH MIN DEATH CASES")  
 for i in name\_sum\_death:  
 if name\_sum\_death[i] == min\_death:  
 print("The State/UT with minimum no of death is:")  
 print(i, ": ", name\_sum\_death[i])  
 state\_wise()  
 if input\_1 == 8:  
 print("GET MAX CONFIRMED CASES IN A DAY:")  
 print(df[df.Confirmed==df.Confirmed.max()])  
 print("-"\*100)  
 print("GET MIN CONFIRMED CASES IN A DAY")  
 print(df[df.Confirmed == df.Confirmed.min()])  
 state\_wise()  
 if input\_1 == 9:  
 print("AVERAGE CONFIRMED CASES PER DAY:")  
 print(df['Confirmed'].mean())  
 print("-"\*100)  
 print("AVERAGE DEATH CASES PER DAY:")  
 print(df['Deaths'].mean())  
 print("-" \* 100)  
 print("AVERAGE CURED CASES PER DAY:")  
 print(df['Cured'].mean())  
 state\_wise()  
  
def state\_wise\_graph():  
 print("#" \* 100)  
 print("1->" + "TO GET STATES VS CURED GRAPH")  
 print("2->" + "TO GET STATES VS DEATH GRAPH")  
 print("3->" + "TO GET CURED-DEATH TREND GRAPH")  
 print("#" \* 100)  
 input\_2 = int(input())  
 if input\_2==1:  
 x=list(grouped\_data['Cured'].sum())  
 y = np.arange(30,1080,30)  
 plt.barh(y,x,height=20)  
 state\_list=list(grouped\_data.groups.keys())  
 plt.yticks(y,state\_list)  
 plt.tick\_params(axis='y',labelsize=7)  
 plt.xlabel("CURED")  
 plt.ylabel("STATES")  
 for i in range(len(x)):  
 plt.annotate(str(x[i]), xy=(x[i], y[i]), ha='left', va='center',fontsize=7)  
 plt.show()  
 state\_wise\_graph()  
 if input\_2==2:  
 x=list(grouped\_data['Deaths'].sum())  
 y = np.arange(30,1080,30)  
 plt.barh(y,x,height=20)  
 state\_list=list(grouped\_data.groups.keys())  
 plt.yticks(y,state\_list)  
 plt.tick\_params(axis='y',labelsize=7)  
 plt.xlabel("DEATHS")  
 plt.ylabel("STATES")  
 for i in range(len(x)):  
 plt.annotate(str(x[i]), xy=(x[i], y[i]), ha='left', va='center',fontsize=7)  
 plt.show()  
 state\_wise\_graph()  
 if input\_2==3:  
 death\_list=list(grouped\_data['Deaths'].sum())  
 cured\_list=list(grouped\_data['Cured'].sum())  
 cnf\_list = list(grouped\_data['Confirmed'].sum())  
 x1 = np.arange(30,1080,30)  
 plt.scatter(x1,death\_list,color='r',label="Deaths")  
 plt.scatter(x1,cured\_list,color='g',label="Cured")  
 plt.scatter(x1, cnf\_list, color='b',label="Confirmed")  
 plt.legend(loc="upper right")  
 plt.show()  
 state\_wise\_graph()  
  
  
print("#"\*100)  
print()  
print("--THE COLUMNS IN THE DATA SET ARE--")  
for i in col:  
 print(i,end=" ")  
print()  
print()  
print("#" \* 100)  
print("#"\*45+"--OPTIONS--"+"#"\*44)  
print()  
print("PRESS 1 TO ANALYZE STATE/UT WISE")  
print("PRESS 2 FOR GRAPHICAL REPRESENTATION")  
print()  
print("#"\*100)  
option\_input=int(input())  
if option\_input==1:  
 state\_wise()  
if option\_input==2:  
 state\_wise\_graph()

**Source Code For SMTP**

import smtplib  
def otp\_sender(rec\_mail,otp):  
 s = smtplib.SMTP('smtp.gmail.com', 587)  
 s.starttls()  
 s.login('group3@apsjorhat.org', 'apsj#12345678')  
 message = str(otp)  
 s.sendmail("group3@apsjorhat.org",rec\_mail, message)  
 s.quit()

**Source Code For Sql Connector**

import mysql.connector  
dbc=mysql.connector.connect(host="localhost",user="root",passwd="root",database="covid")  
cursor=dbc.cursor()

**Commands Used In MySql**

**Creating database**

Create database covid**;**

**Using database**

Use covid**;**

**Creating table and inserting values**

Create table reg\_user(id int(10) primary key,name varcher(30),email\_id varchar(30))**;**

**Desc table**

Desc reg\_user**;**

**Inserting values**

insert into reg\_users values((1,”Shreshth Sharma”, “6540@apsjorhat.org”)(2,"Deepjyoti Medhi","3295@apsjorhat.org"),(3,"Apolo Rana","3656@apsjorhat.org)**;**

**To fetch all values**

Select \* from reg\_user**;**

**Conclusion and Future Work**

By making this project we have analysed the overall medical condition of the COVID-19 pandemic in India for its 28 states and 8 union territories on a daily basis , in critical natural calamities like these we need to have an accurate information of the problems prevailing in the present case scenario which has been experimented by us in this project , by doing so we have been able to device a method which can help us to analyse the on ground condition (for example: the no of positive cases, death rate of each state, etc) of every state and take better decisions and suitable preventive measures to culminate the prevailing problems.

The functions performed by this experimental project are accurate only in accordance to our assumptions for further upgradation we can also add features such as real time analysis and a Graphical User Interface to this project , which can be developed in the form of a web , desktop or mobile application. This project can also be scaled to analyse global data in real time.

**References**

[1] <https://www.limswiki.org/index.php/MySQL>

[2] <https://en.wikipedia.org/wiki/PyCharm>

[3] <https://en.wikipedia.org/wiki/Pandas_(software)>

[4] <https://en.wikipedia.org/wiki/NumPy>

[5] <https://en.wikipedia.org/wiki/Matplotlib>