**COMPUTER SCIENCE**

**PROJECT FILE**

Project prepared by : Harsh Vishwakarma

XII-A

Roll No. – 11100011001

Session: 2020-2021

Yuvashakti Model School Sector-3, Rohini, Delhi-85

TABLE OF CONTENTS

* Certificate
* Acknowledgement
* Modules Used
* Brief Overview of Project
* Coding
* Output Screens
* Database and Tables used in MYSQL
* Bibiliography

**CERTIFICATE**

This is to certify that HARSH VISHWKARMA

student of Class XII A , Yuvashakti Model

School has completed solely and satisfactorily

regarding the project titled **"** **Sudo Tech**

**Incorportations "** during the academic

year 2020-2021 towards fulfillment of credit for

the Computer Science practical conducted by

CBSE under my supervision.

Ms. Swati Narang

**Department of Computer Science**

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to my computer science mentor Ms. Swati Narang, for her vital support, guidance and encouragement - without which this project would not have come forth. Her unflagging patience, creativity and immense knowledge that she shared with us have proved highly beneficial to us and have made our Project File both possible and successful.

**HARSH VISHWAKARMA**

**XII-A**

MODULES USED

* *colorama==0.4.1*

Colorama is a module used for printing colors to windows command prompt/powershell as cmd/powershell doesn’t officially support **ANSI escape sequence**. To print colors to windows terminal **init() method** from this moule is used in this project .

* *easygui==0.98.1*

Easygui module is a module used for Graphical User Interface (GUI) related tasks. Its **fileopenbox()** method is used for taking input of files from the user .

* *mysql-connector-python==8.0.22*

This module is used for connecting mysql-database to our python program for insertion, deletion,updation,searching and modifcation related tasks. Its **fetchone(),fetchall(),execute()** methods have been used . Connection object and cusor object OF THIS MODULE had also been used in this project. This module is basically **for interacting with our mysql datatbase** through python.

* *prettytable==1.0.1*

This module is used for displaying the data fetched from our mysql database through our python script for making this project more user-friendly and simple for end user to understand. This moduledisplays the fetched data in the form of table as displayed in mysql command-line-client.

* *pyfiglet==0.* *8.post1*

This module is used for printing terminal art (designing on terminal) or ASCII art onto terminal/cmd prompt .Its figlet\_format() method has been used for converting text to ASCII art .

* *PyPDF2==1.26.0*

This module has been used for handling **PDF** files in this project and perform some operations like extracting text using **extractText()** method from **PDF** file , getting number of pages in pdf file using **numPage** method , selecting a particular page in pdf using **getPage()** method and performing above operations and some other methods imported from this module has been used.

* *termcolor==1.1.0*

This module is used for designing menu options like changing the font color and filling colors onto the terminal through **colored(),cprint()** methods of this module.

* *tqdm==4.54.1*

This module is used for **loading interface** onto the terminal. It is cross platform and supereasy to use.Its **trange()** method has been used in this project to make **progress bar** while program is launching.

* *plyer*

This module is used for sending **push notifications** in this project by using its notify() method . It is cross-platform which means you can use it on linux,windows,macos.

* *requests*

This module is used for making get requests to website through the python program only by using its **get()** method.

* json

This module is used to convert the fetched data from website into json format using **json()** method **. JSON (JavaScriptObjectNotation)** format is widely used for information exchange also it is both **machine and human-readable.**

* *Beepy*

This module is used for push notification sound .It plays audio after every push notification.

* *pyttsx3==2.7*

This module is basically **a TTS(Text-to-Speech)** converter module . It allows us to read the text with the help of ***Windows OS inbuilt voice*** and using windows inbuilt voice **API *,SAPI (SPEECH APPLICATION PROGRAMMING INTERFACE).***

Brief Overview of Project

**This project is aimed at solving real world problems and providing end user a quality service from our company . This project consists of a employee management system , a file reader and a real time corona virus notification system .**

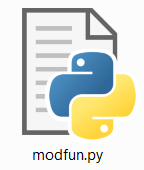
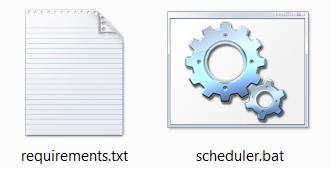
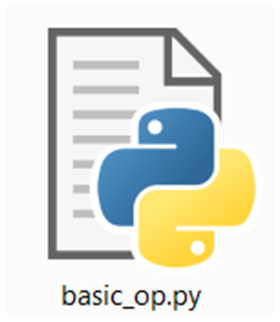
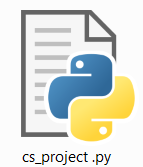
* **Employee Management System:** This system can be used for **management** of employee details in business and organisations .Employee such as name,employeeID,salary,designation etc can be added , modified , deleted

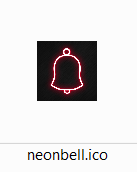
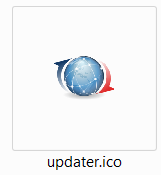
and displayed.One can also search details of any particular Employee . There is also a feature using which we can **send** **e-mails** to any of the employees.

It makes managing details of Employee very **convinient** for business.Features like **push notifications** and **notification sound** is also added.

* **File Reader:** This application can be used for reading texts from any *pdf/text* files.The **system generated voice (computer-voice)** is used to read in this program to read the content from the files This application is also beneficial for the *visually-impaired people*. The **number of pages** in **PDF** file can also be seen during the running of application.
* **Covid-Notification-System:** This system can be used for getting regular updates regarding the **coronavirus pandemic** in India . Real time data will be displayed through the **push notifications** with a short-term buzzing sound . One can also ***schedule notification*** for every 15 minutes to be displayed on their system.

Structure of my project

****



CODING (cs\_project.py)

import os

from termcolor import colored,cprint

from colorama import init

import pyfiglet

from basic\_op import addrec,delrec,searchrec,disprec,mailfun

from modfun import modrec

from tqdm import tqdm,trange

from time import sleep

from filereader import freader

from prettytable import PrettyTable

from covidinfo import coronainfo

from plyer import notification as nt # Notification system

from beepy import beep #Notification sound

x=PrettyTable()

init() #WINDOWS PLATFORM FOR PRINIRING COLOrs on cmd/powershell prompt!

x.field\_names=['EmpName','Gender','EmpId','Designation','Age','Emp\_Mail','Salary']

# ATTRIBUTES OF TABLE

def progressbar():

for i in trange(100):

sleep(0.02) #LOADING INTERFACE FUNCTION

def menu():

cprint(colored('[1] Enter 1 to use Employee Management System ','white','on\_red'))

print()

cprint(colored('[2] Enter 2 to use File Reader ','magenta','on\_white'))

print()

cprint(colored('[3] Enter 3 to use Covid Notification System

','grey','on\_green'))

print()

cprint(colored(' SUDO TECH INCORPORATION ','white','on\_red'))

print()

a=pyfiglet.figlet\_format('MENU')

print()

print('=================================================')

print(a)

menu()

choice=int(input('Enter your choice:'))

print()

print('-------------------------------------------------')

if choice==1:

def sqlmenu():

print(

'''

[1]. ADD RECORD

[2]. DELETE RECORD

[3]. MODIFY RECORD

[4]. DISPLAY ALL RECORDS

[5]. SEARCH RECORDS

[6]. SEND A MAIL TO EMPLOYEE

[7]. EXIT THE PROGRAM''')

print() #FORMATTING PURPOSE

print()#fORMATTING

print("[!] Connecting to Database !")

sleep(1.0)

print("[!] Initializing Database !")

sleep(1)

print('[!] Loading database records !')

progressbar()

sleep(1)

print('DATABASE LOADED SUCCESSULLY !')

cprint(colored('\*\*\*\*\*\*EMPLOYEE MANAGEMENTSYSTEM\*\*\*\*\*\*','white','on\_red'))

banner=pyfiglet.figlet\_format('EMS')

print(banner)

cprint(colored("options:",'blue','on\_white'))

print()

ans='y'

while ans=='y' or ans=='Y':

sqlmenu()

menu=int(input("Type your option : "))

if menu==1:

addrec()

print()# For some gap b/w records and y/n input

elif menu==2:

delrec()

print()# For some gap b/w records and y/n input

elif menu==3:

modrec()

print()# For some gap b/w records and y/n input

elif menu==4:

try:

L=disprec() #Store nested list of records

for data in L:

x.add\_row(data) #

print(x)

nt.notify(title='Success',message='RECORDS FOUND',app\_icon='gtick.ico',timeout=3)

beep(sound='ping')

x.clear\_rows()

except Exception as e: #runtime error if no recs

cprint(colored('ERROR!!','white','on\_red'))

nt.notify(title='Database

Empty',message='No records to display ‘,app\_icon='error.ico',timeout=3)

beep(sound='error')

print()

# For some gap b/w records and y/n input

elif menu==5:

try: #if not none

L=searchrec() #L WILL STORE LIST

x.add\_row(L)

print(x)

nt.notify(title='Success',message='RECORD FOUND',app\_icon='gtick.ico',timeout=3)

beep(sound='ping')

x.clear\_rows()

except Exception as e:

cprint(colored('ERROR!!','white','on\_red'))

print()# For some gap b/w records and y/n input

nt.notify(title='Failure',message='Error',app\_icon='error.ico',timeout=3)

beep(sound='error')

elif menu==6:

try:

mailfun()

except Exception as e:

cprint(colored("MAIL NOT SENT , PLEASE CONNECT TO INTERNET!",'white','on\_blue'))

nt.notify(title='Failure',message='Please connect to Internet',app\_icon='error.ico',timeout=3)

beep(sound="error")# Notifiication sound

elif menu==7:

cprint(colored('BYE USER !!','white','on\_red'))

nt.notify(title='Exit',message='BYE USER',app\_icon='redbell.ico',timeout=3)

beep(sound='success')

sleep(4)

exit()

else:

print('Invalid option !!')

print()# For some gap b/w records and y/n input

nt.notify(title='Invalid ',message='Invalid option ',app\_icon='error.ico',timeout=3)

beep(sound='error')

ans=input("Want to continue type[y], else[n] :")

elif choice==2:

freader() # FILE READER

elif choice==3:

print()

cprint(colored('COVID\_\_INFO\_\_NOTIFIER\_\_SYSTEM','white','on\_red'))

print()

coronainfo()

print('''After typing y you can close this program and

notification will be sent to you every 15 minutes ''')

print()

ch3=input("If you want notification every 15 minute type[y],else[n]:")

if ch3=='y' or 'Y':

os.system('scheduler.bat')

cprint(colored('Notification Scheduled for every 15 minutes !','white','on\_red'))

else:

cprint(colored('Bye User','white','on\_red'))

else:

cprint(colored('Error , Invalid option!','white','on\_red'))

nt.notify(title='Invalid ',message='Invalid option !',app\_icon='error.ico',timeout=3)

beep(sound='error')

Coding (basic\_op.py)

**from email.mime.text import MIMEText**

**from email.mime.multipart import MIMEMultipart**

**import smtplib**

**from termcolor import cprint,colored**

**from colorama import init # WINDOWA PLATFORM**

**import mysql.connector as ms# for line 15**

**from time import sleep # FOR LINE 19**

**from plyer import notification as nt # Notification system**

**from beepy import beep #Notification sound**

**init() #WNDOWS PLATFORM**

**try:**

**mycon=ms.connect(host='localhost',user='root',passwd='Kali',database='Emplmgs',autocommit=True) #,autocommit=True**

**cursor=mycon.cursor()**

**except Exception as e:**

**cprint(colored("Error connecting to databse!!",'white','on\_yellow'))**

**sleep(4)# TIME FOR USER TO READ ERROR MESSAGE**

**exit() #TERMINATING THE PROGRAM**

**def addrec():**

**EmpName=input('Enter Employee name :').upper()**

**Gender=input("Enter gender [M]/[F] :").upper()**

**EmpId=input('Enter unique 5 Digit Employee ID:').upper()**

**Designation=input("Enter Designation :").upper()**

**Age=int(input('Enter age of Employee:'))**

**Emp\_Mail=input("Enter E-Mail id of Employee:").lower()**

**Sal=int(input("Enter Salary of Employee :"))**

**add\_query="insert into Emplrec values('{}','{}','{}','{}',{},'{}',{})".format(EmpName,Gender,EmpId,Designation,Age,Emp\_Mail,Sal)**

**cursor.execute(add\_query)**

**mycon.commit()**

**cprint(colored("RECORD ADDED SUCCESSULLY !",'white','on\_red'))**

**nt.notify(title='Success',message='RECORDS ADDED**

**SUCCESSULLY',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**#Notification SYSTEM**

**def delrec():**

**EmpId=input("Enter unique 5 Digit Employee ID:").upper()**

**CHK\_QUERY='Select \* from Emplrec where EmpId="{}"'.format(EmpId)**

**cursor.execute(CHK\_QUERY)**

**datachk=cursor.fetchone() #NONE IF IT IS EMPTY**

**if datachk!=None:**

**EmpName,Gender,EmpId,Designation,Age,Emp\_Mail,Salary=datachk**

**del\_query='Delete from Emplrec where EmpId="{}"'.format(EmpId)**

**cursor.execute(del\_query)**

**mycon.commit()**

**cprint(colored("RECORD DELETED SUCCESSULLY !",'white','on\_red'))**

**nt.notify(title='Success',message='Record deleted**

**successfully',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**else:**

**cprint(colored("INVALID EMPID , ID DOES NOT EXIST IN DATABASE**

**!",'green','on\_white'))**

**nt.notify(title='Error!',message='EMPID DOES NOT EXIST IN DATABASE**

**!',app\_icon='error.ico',timeout=3)**

**beep(sound='error')**

**def searchrec():**

**Id=input("Enter unique 5 Digit Employee ID:").upper()**

**search\_q="Select \* from Emplrec where EmpId='{}'".format(Id)**

**cursor.execute(search\_q)**

**data=cursor.fetchone() #iT RETURNS NONE IF IT IS EMPTY**

**if data!=None:**

**EmpName,Gender,EmpId,Designation,Age,Emp\_Mail,Salary=data**

**L=[EmpName,Gender,EmpId,Designation,Age,Emp\_Mail,Salary]**

**return L**

**else:**

**cprint(colored('INVALID ID !!, ID DOES NOT EXIST','grey','on\_white'))**

**def disprec():**

**l=[]#TO HOLD ALL RECORDS**

**dis\_q='Select \* from Emplrec'**

**cursor.execute(dis\_q)**

**data=cursor.fetchall() # LIST TYPE ELEMENT**

**if data!=[]: #IF RECORDS ARE PRESENT**

**for row in data: # iTERATING THROUGH THE RESULT SET**

**# print(row) NOT PRINTING TUPLE , SHOWING TABLE..**

**l.append(list(row)) #CONVERTING TUPLE TO LIST(NESTED) FOR ADDING IN PRETTYTABLE**

**return l**

**else: # IF NO RECORDS IN DATABASE**

**cprint(colored('NO RECORDS EXIST!, DATABASE EMPTY!','white','on\_blue'))**

**def mailfun():**

**EmpId=input("Enter EmpId of Employee:")**

**EmpId=EmpId.upper()**

**q="Select \* from Emplrec where EmpId='{}'".format(EmpId)**

**cursor.execute(q)**

**r=cursor.fetchone()**

**if r!=None:**

**EmpName,Gender,EmpId,Designation,Age,Emp\_Mail,Salary=r # UNPACKING .....**

**Msg=MIMEMultipart()**

**BOT\_ID='sudocorporations@gmail.com'**

**BOT\_PWD='sudo\_tech@#'**

**server=smtplib.SMTP('smtp.gmail.com',587) #HOST AND PORT**

**# server.ehlo()**

**server.starttls()#Transport layer secuirity encryption**

**server.login(BOT\_ID,BOT\_PWD)**

**Subject=input('Enter Subject of mail:')**

**body=input('Enter body of mail:')**

**body=body+'\n'+ '\n' + "Thank you"+ '\n' +"Sudo Corporations"**

**Msg["From"]=BOT\_ID**

**Msg['To']=Emp\_Mail**

**Msg["Subject"]=Subject**

**Msg.attach(MIMEText(body,'plain')) #ATTACHING BODY**

**content=Msg.as\_string()**

**server.sendmail(BOT\_ID,Emp\_Mail,content)**

**print()**

**o=f"MAIL SENT SUCCESSFULLY TO {EmpName} whose EmpID is: {EmpId}"**

**cprint(colored(o,'white','on\_red'))**

**server.quit()**

**nt.notify(title='Success',message='Mail sent successfully',app\_icon='gtick.ico',timeout=3)**

**beep(sound="ping") # Notification sound**

**else:**

**cprint(colored("INVALID EmpId !!",'white','on\_green'))**

**nt.notify(title='INVALID ID',message='Id does not exist in**

**database!',app\_icon='error.ico',timeout=3)**

**beep(sound="error") # Notifiication sound**

CODING(modfun.py)

**import mysql.connector as ms**

**from colorama import init**

**from time import sleep**

**from termcolor import cprint,colored**

**from plyer import notification as nt # Notification system**

**from beepy import beep #Notification sound**

**init()**

**try:**

**mycon=ms.connect(host='localhost',user='root',passwd='Kali',database='Emplmgs',autocommit=True) #,autocommit=True**

**cursor=mycon.cursor()**

**except Exception as e:**

**cprint(colored("Error connecting to databse!!",'white','on\_red'))**

**sleep(4)# TIME FOR USER TO READ ERROR MESSAGE**

**exit() #TERMINATING THE PROGRAM**

**def modrec():**

**#EMPID HAS ALREADY UNIQUE CONSTRAINT CHECK IN MYSQL**

**print(**

**'''**

**[1]. EmpName**

**[2]. Gender**

**[3]. Designation**

**[4]. Age**

**[5]. Emp\_Mail**

**[6]. Salary**

**[7]. Increase Salary by %**

**[8]. Decrease Salary by %**

**'''**

**)**

**ch=int(input('Enter your choice:'))**

**EID=input("Enter unique 5 Digit Employee ID:").upper() #to match casing**

**chk\_q="Select \* from Emplrec where EMPID='{}'".format(EID)**

**cursor.execute(chk\_q)**

**r=cursor.fetchone()**

**if r!=None:**

**EmpName,Gender,EmpId,Designation,Age,Emp\_Mail,Salary=r**

**if ch==1:**

**field="Name" # For sending in mail**

**New\_Name=input('Enter new name :').upper()**

**query='update Emplrec set EmpName="{}" where**

**EmpId="{}"'.format(New\_Name,EID)**

**cursor.execute(query)**

**mycon.commit()**

**o=f'NAME UPDATED SUCCESSULLY TO : {New\_Name} '**

**cprint(colored(o,'white','on\_red'))**

**nt.notify(title='Updation Successful',message='NAME UPDATED**

**SUCCESSULLY',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**elif ch==2:**

**field="Gender"**

**Gen=input("Enter new gender [M]/[F]:").upper()**

**query='update Emplrec set Gender="{}" where EmpId="{}"'.format(Gen,EID)**

**cursor.execute(query)**

**mycon.commit()**

**o=f"GENDER UPDATED SUCCESSULLY TO: {Gen}"**

**cprint(colored(o,'white','on\_red'))**

**nt.notify(title='Updation Successful',message='GENDER UPDATED**

**SUCCESSULLY',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**elif ch==3:**

**field="Designation"**

**Des=input("Enter New Designation:").upper()**

**query='update Emplrec set Designation="{}" where EmpId="{}"'.format(Des,EID)**

**cursor.execute(query)**

**mycon.commit()**

**o=f"DESIGNATION UPDATED SUCCESSULLY TO : {Des}"**

**cprint(colored(o,'white','on\_red'))**

**nt.notify(title='Updation Successful',message='DESIGNATION UPDATED**

**SUCCESSULLY',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**elif ch==4:**

**field='Age'**

**age=int(input("Enter new age:"))**

**query='update Emplrec set Age={} where EmpId="{}"'.format(age,EID)**

**cursor.execute(query)**

**mycon.commit()**

**o=f"AGE UPDATED SUCCESSULLY TO :{age}"**

**cprint(colored(o,'white','on\_red'))**

**nt.notify(title='Updation Successful',message='AGE UPDATED**

**SUCCESSULLY',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**elif ch==5:**

**field="MAIL ID"**

**mail\_id=input("Enter new mail id :").lower()**

**query='update Emplrec set Emp\_Mail="{}" where**

**EmpId="{}"'.format(mail\_id,EID)**

**cursor.execute(query)**

**mycon.commit()**

**o=f'MAIL ID UPDATED SUCCESSULLY TO:{mail\_id}'**

**cprint(colored(o,'white','on\_red'))**

**nt.notify(title='Updation Successful',message='MAIL ID UPDATED**

**SUCCESSULLY',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**elif ch==6:**

**field="Salary"**

**sal=int(input("Enter updated Salary :"))**

**sal\_q="update Emplrec set Salary={} where EmpId='{}'".format(sal,EID)**

**cursor.execute(sal\_q)**

**mycon.commit()**

**o=f"SALARY UPDATED SUCCESSULLY TO: {sal}"**

**cprint(colored(o,'white','on\_red'))**

**nt.notify(title='Updation Successful',message='SALARY UPDATED**

**SUCCESSULLY',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**elif ch==7:**

**per=int(input("Enter the % by which salary to be Increased:"))**

**r\_per=per/100**

**in\_sal=int((Salary)+(Salary\*r\_per))**

**q="Update Emplrec set Salary={} where EmpId='{}'".format(in\_sal,EID)**

**cursor.execute(q)**

**mycon.commit()**

**o=f"SALARY HAS BEEN INCREASED BY {per}%"**

**cprint(colored(o,'white','on\_red'))**

**nt.notify(title='Updation Successful',message='Salary**

**Increased!',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**elif ch==8:**

**per=int(input("Enter the % by which salary to be Decreased:"))**

**r\_per=per/100**

**de\_sal=int((Salary)-(Salary\*r\_per))**

**# print(de\_sal)**

**q="Update Emplrec set Salary={} where EmpId='{}'".format(de\_sal,EID)**

**cursor.execute(q)**

**mycon.commit()**

**o=f"SALARY HAS BEEN DECREASED BY {per}%"**

**cprint(colored(o,'white','on\_red'))**

**nt.notify(title='Updation Successful',message='Salary**

**Decreased!',app\_icon='gtick.ico',timeout=3)**

**beep(sound='ping')**

**else:**

**cprint(colored("INVALID OPTION!!",'green','on\_white'))**

**nt.notify(title='Error',message='Invalid option!',app\_icon='error.ico',timeout=3)**

**beep(sound='error')**

**else:**

**cprint(colored("INVALID EMPLOYEE ID , ID DOES NOT EXIST IN**

**DATABASE",'white','on\_red'))**

**nt.notify(title='INVALID EMPLOYEE ID',message='ID DOES NOT EXIST IN DATABASE',app\_icon='error.ico',timeout=3)**

**beep(sound='error')**

CODING(filereader.py)

**import pyttsx3 #speech ENGINE**

**from colorama import init #WINDOWS COLOR FORMATTING**

**from termcolor import cprint,colored # COLOR FORMATTING TEXT**

**from time import sleep**

**import PyPDF2 #pdf file handling**

**import pyfiglet #ascii art**

**from tqdm import trange # loading interface module**

**import easygui # GUI FILE INPUT**

**init()**

**engine=pyttsx3.init('sapi5') #SAPI5 IS THE WINDOWS API FOR IN-BUILT VOICES OF WINDOWS**

**voices=engine.getProperty('voices')**

**engine.setProperty('voice',voices[0].id)**

**def speak(text):**

**engine.say(text)**

**engine.runAndWait()**

**def progressbar():**

**for i in trange(100):**

**pass**

**sleep(0.02) #LOADING INTERFACE FUNCTION**

**def freader():**

**cprint(colored('[!] INITIALISING PROGRAM ...','grey','on\_white'))**

**speak('initialsing program in your system')**

**print()**

**cprint(colored('[!] CONFIGURING SETTINGS ...','white','on\_blue'))**

**speak("configuring settings")**

**print()**

**cprint(colored('[!] LOADING.......','grey','on\_yellow'))**

**speak("loading program in your system")**

**print()**

**progressbar()**

**print()**

**cprint(colored('[!] LOADING SUCCESSFULLY'))**

**speak('PROGRAM LOADED SUCCESSFULLY IN YOUR SYSTEM')**

**print()#FORMATTING PURPOSE**

**speak("Please enter your name user")**

**u\_name=input("Please enter your name user: ")**

**print()**

**cprint(colored('\*\*\*\*FILE READER\*\*\*\*','white','on\_red'))**

**print()#FORMATTING PURPOSE**

**a=pyfiglet.figlet\_format('FILE READER')**

**speak("FILE Reader")**

**print(a)**

**filetype=''' Supported File Types**

**[1]. Text**

**[2]. PDF**

**'''**

**cprint(colored(filetype,'grey','on\_white'))**

**speak("Supported file types are of only text and pdf format ")**

**ans='y'**

**while ans=='y'or ans=="Y":**

**speak(f"Hey {u\_name},, please select your file ")**

**path=easygui.fileopenbox() # RETURNS THE PATH IN DOUBLE QUOTES**

**if path!=None: # if user has selected file**

**speak("The file has been selected")**

**if path.endswith('.pdf'):**

**cprint(colored("PDF File**

**Selected",'grey','on\_yellow'))**

**speak(f"You have selected a pdf file,,**

**{u\_name} ")**

**speak(f'I am going ,,,,to read,, it ,,,for ,,,,you.**

**,,{u\_name}')**

**with open(path,'rb') as f:**

**pdfr=PyPDF2.PdfFileReader(f)**

**page=pdfr.numPages**

**cprint(colored(f'No of pages in**

**selected pdf**

**:{page}','grey','on\_yellow'))**

**speak(f'No of pages in selected**

**pdf are :,{page}')**

**speak(f"Reading it for you**

**{u\_name}")**

**print('Reading.....')**

**for i in range(page):**

**t=pdfr.getPage(i)**

**text=t.extractText()**

**speak(text)**

**elif path.endswith('.txt'):**

**cprint(colored("Text File**

**Selected",'white','on\_blue'))**

**speak(f"You have selected a text file {u\_name}")**

**speak(f"Reading it for you {u\_name}")**

**with open(path) as f:**

**con=f.read()**

**speak(con)**

**else:**

**speak("You have selected Unsupported file**

**type ")**

**speak("please select only pdf file or text file")**

**cprint(colored("Unsupported file**

**type",'white','on\_red'))**

**speak('Please select only .TEXT/PDF format**

**file!!.')**

**print("Only TEXT/PDF file type Supported!")**

**ans=input('To continue type[y] , otherwise type[n] to**

**exit: ')**

**else: #IF NO FILE SELECTED BY USER .**

**cprint(colored("You have not selected any file**

**",'white','on\_red'))**

CODING (covidinfo.py)



CODING

from time import sleep

from covidinfo import coronainfo

while True:

coronainfo()

sleep(15\*60)

**covidnotscr.py**

START pythonw.exe covidnotscr.py

exit

**scheduler.bat**

**colorama==0.4.1**

**easygui==0.98.1**

**mysql-connector-python==8.0.22**

**prettytable==1.0.1**

**pyfiglet==0.8.post1**

**PyPDF2==1.26.0**

**pyttsx3==2.7**

**termcolor==1.1.0**

**tqdm==4.54.1**

**pywin32**

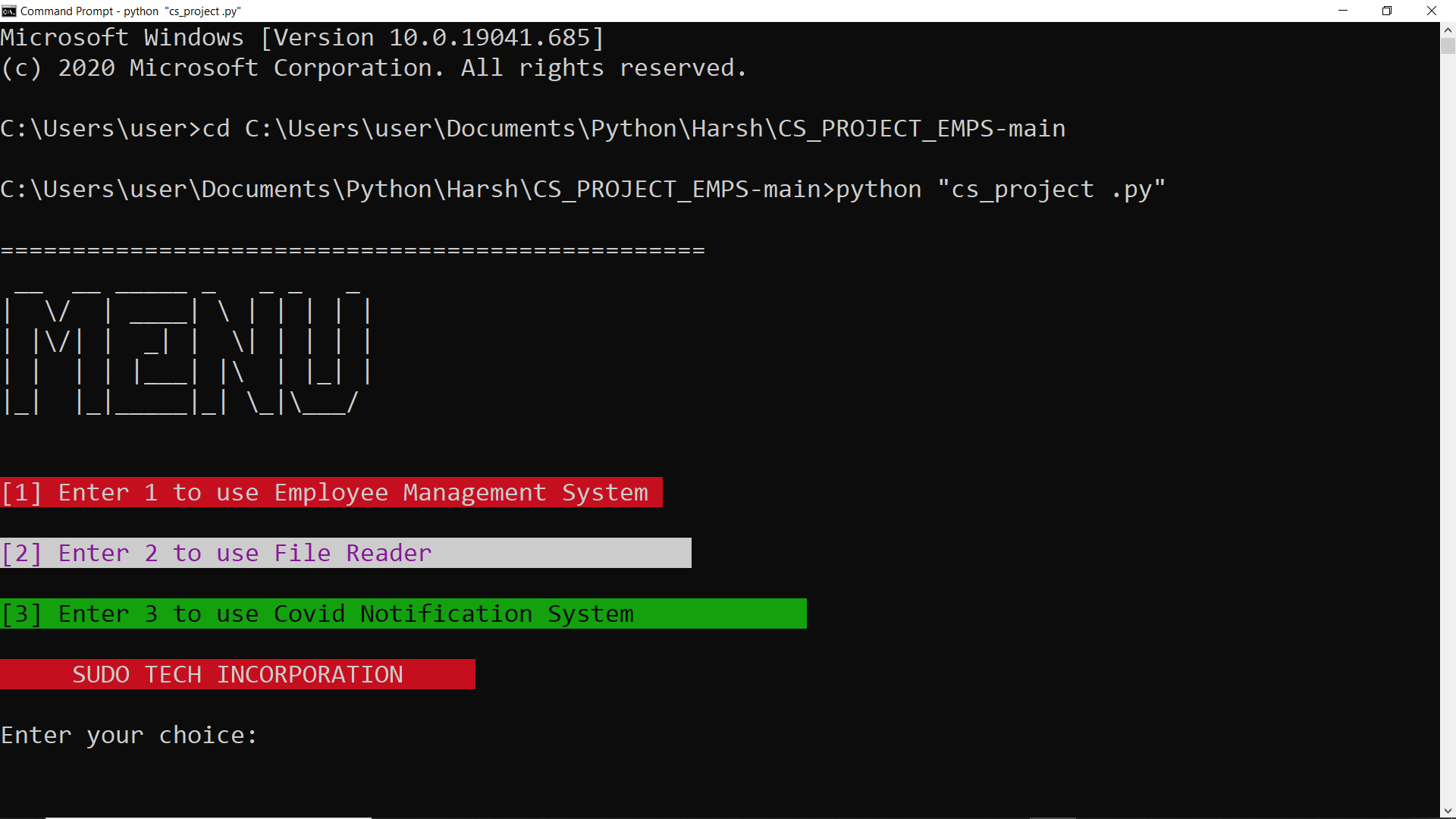
**beepy==1.0.7**

**plyer**

**requests**

**requirements.txt**

**Output Screens**

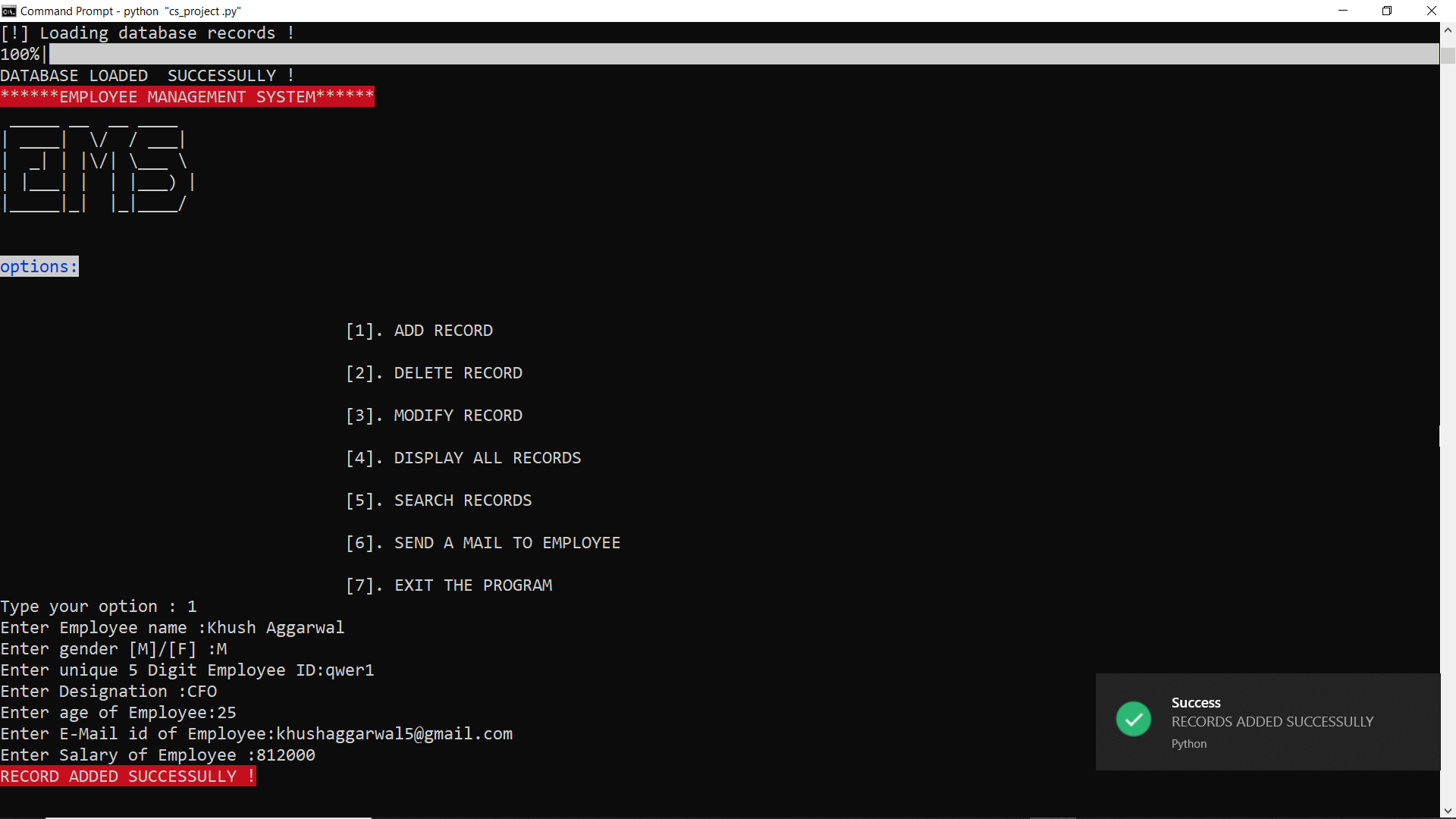


MENU



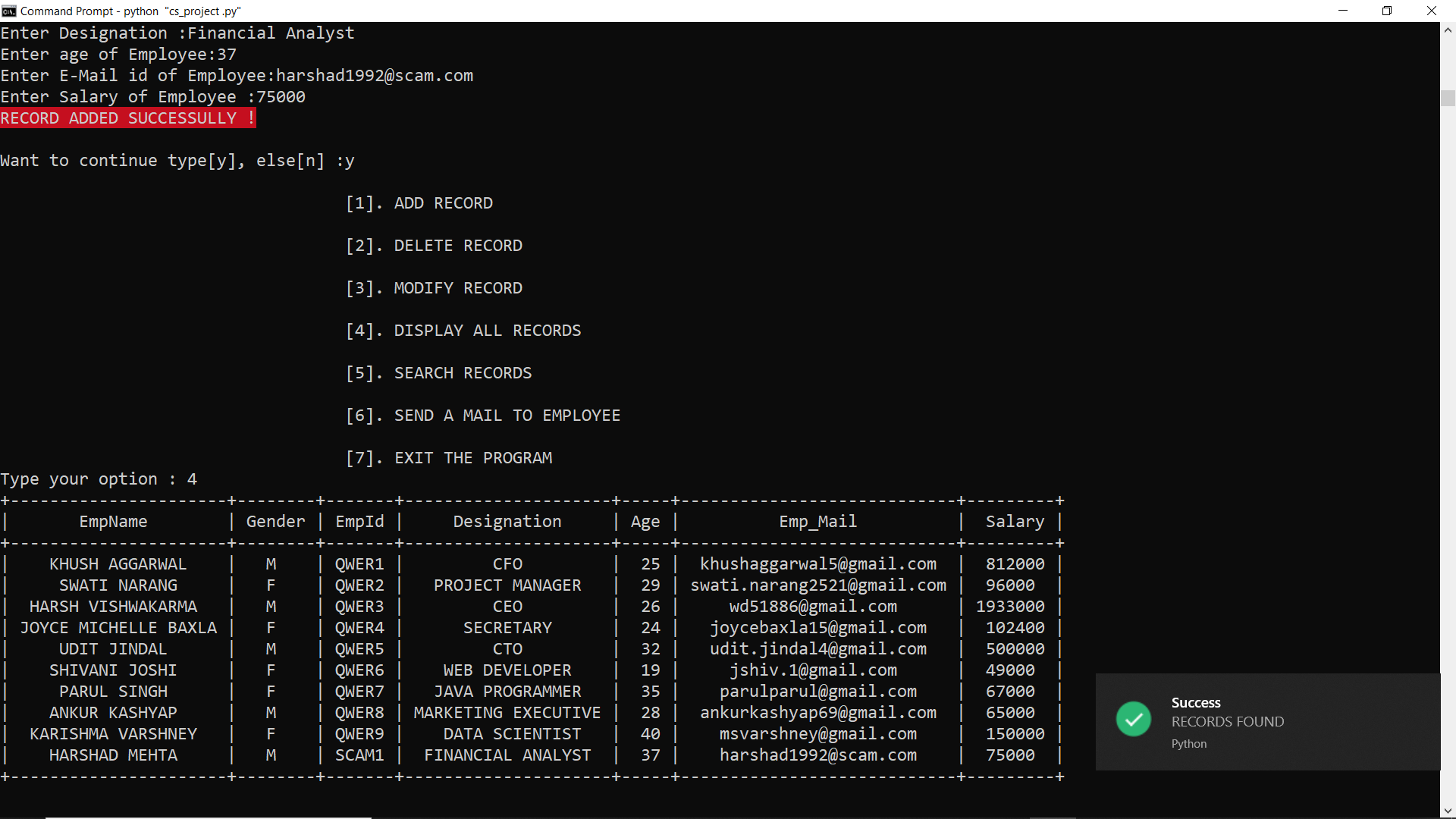
**Employee Management system**

**(Selected)**

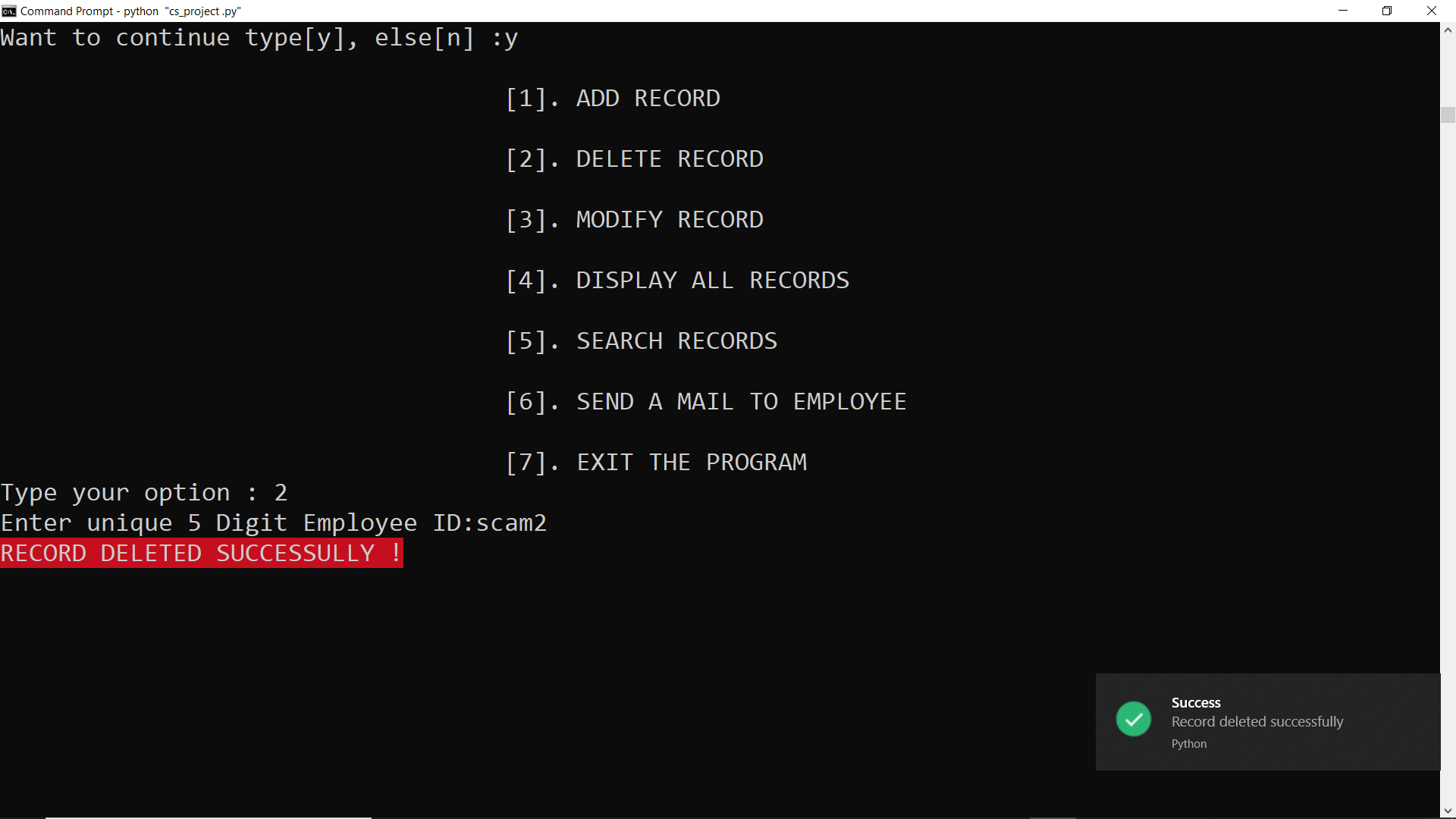


**Addition of record in database**

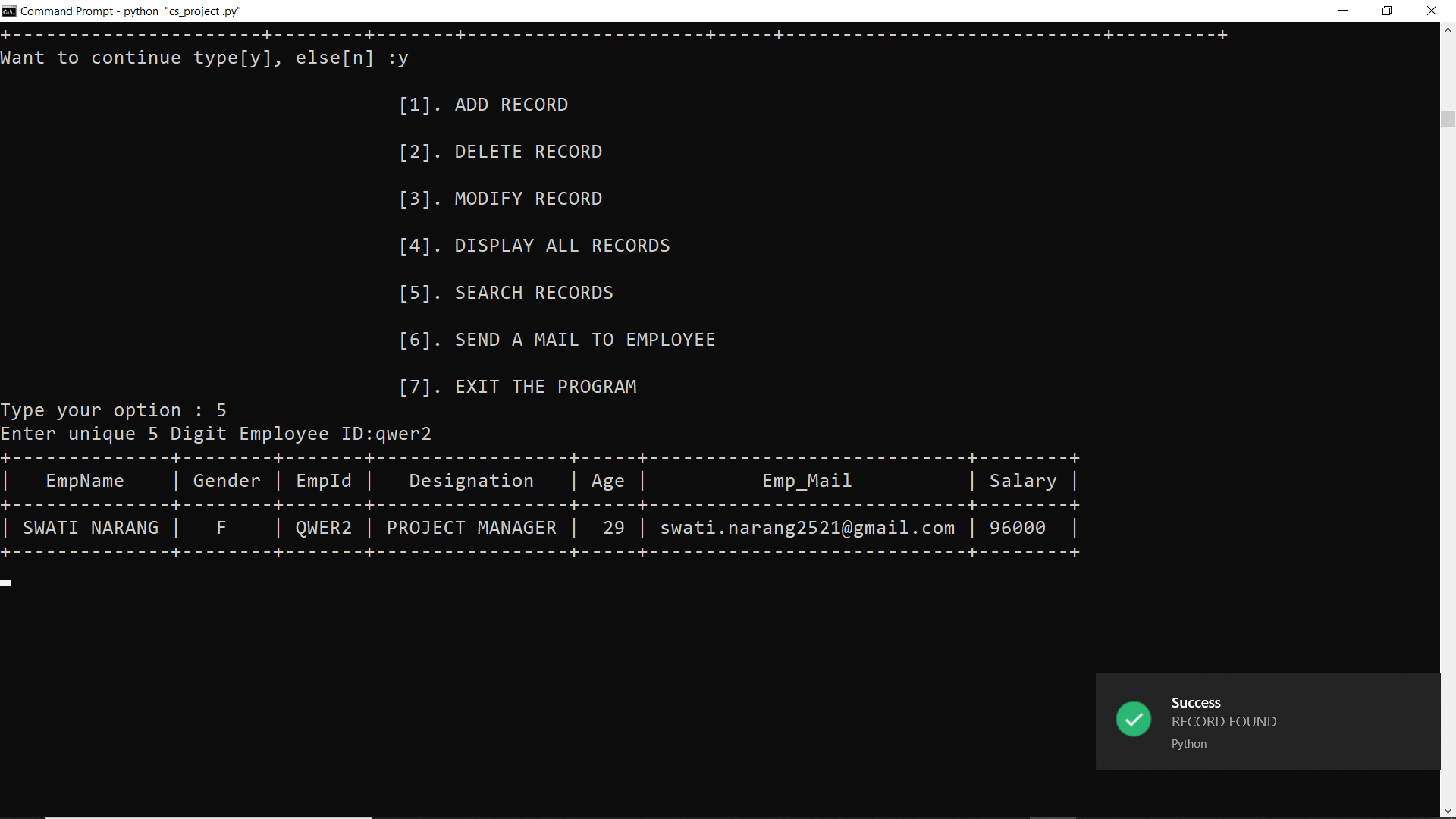
**Notification was also displayed with a sound.**



**Displaying all records in database.**

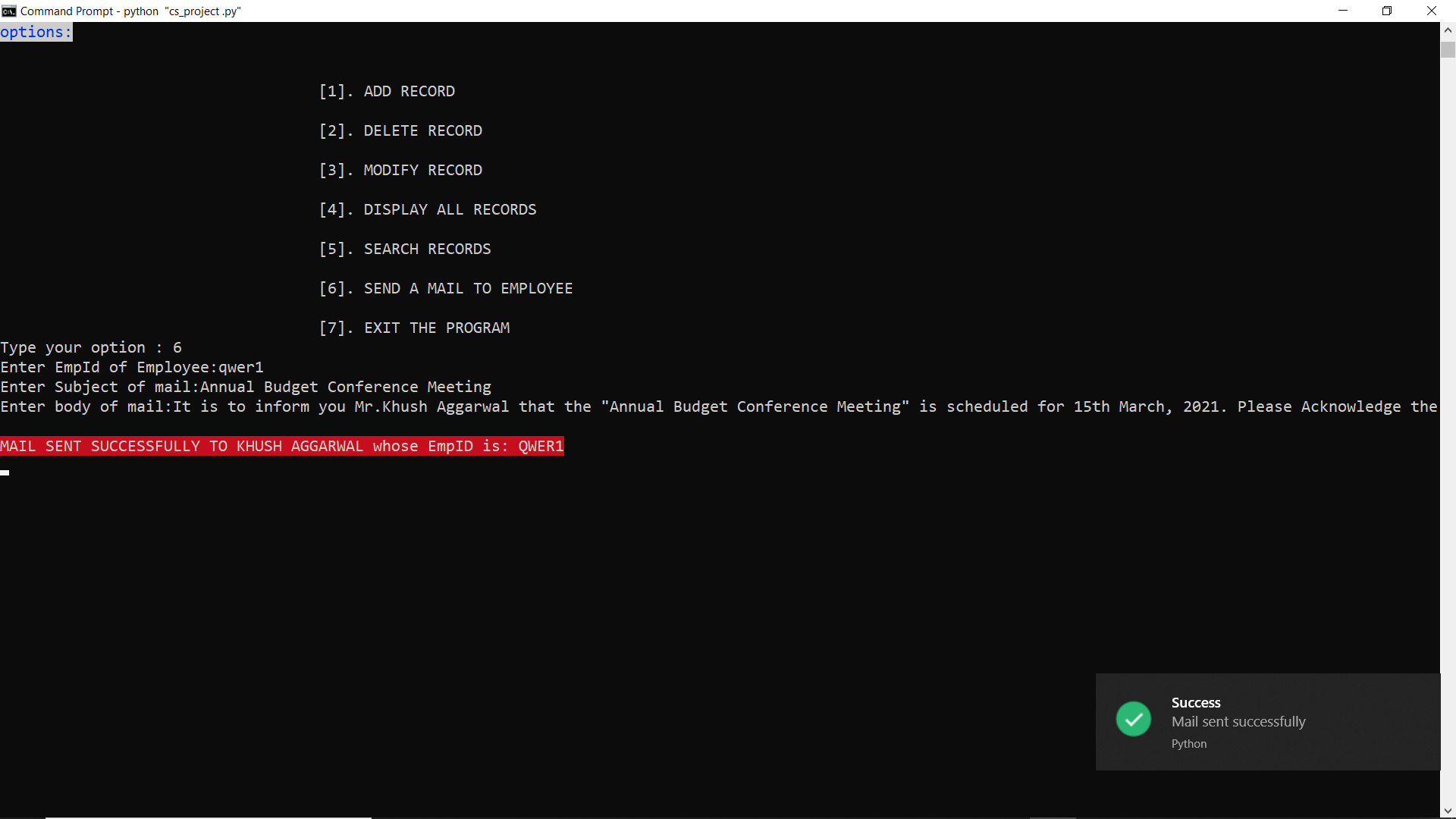


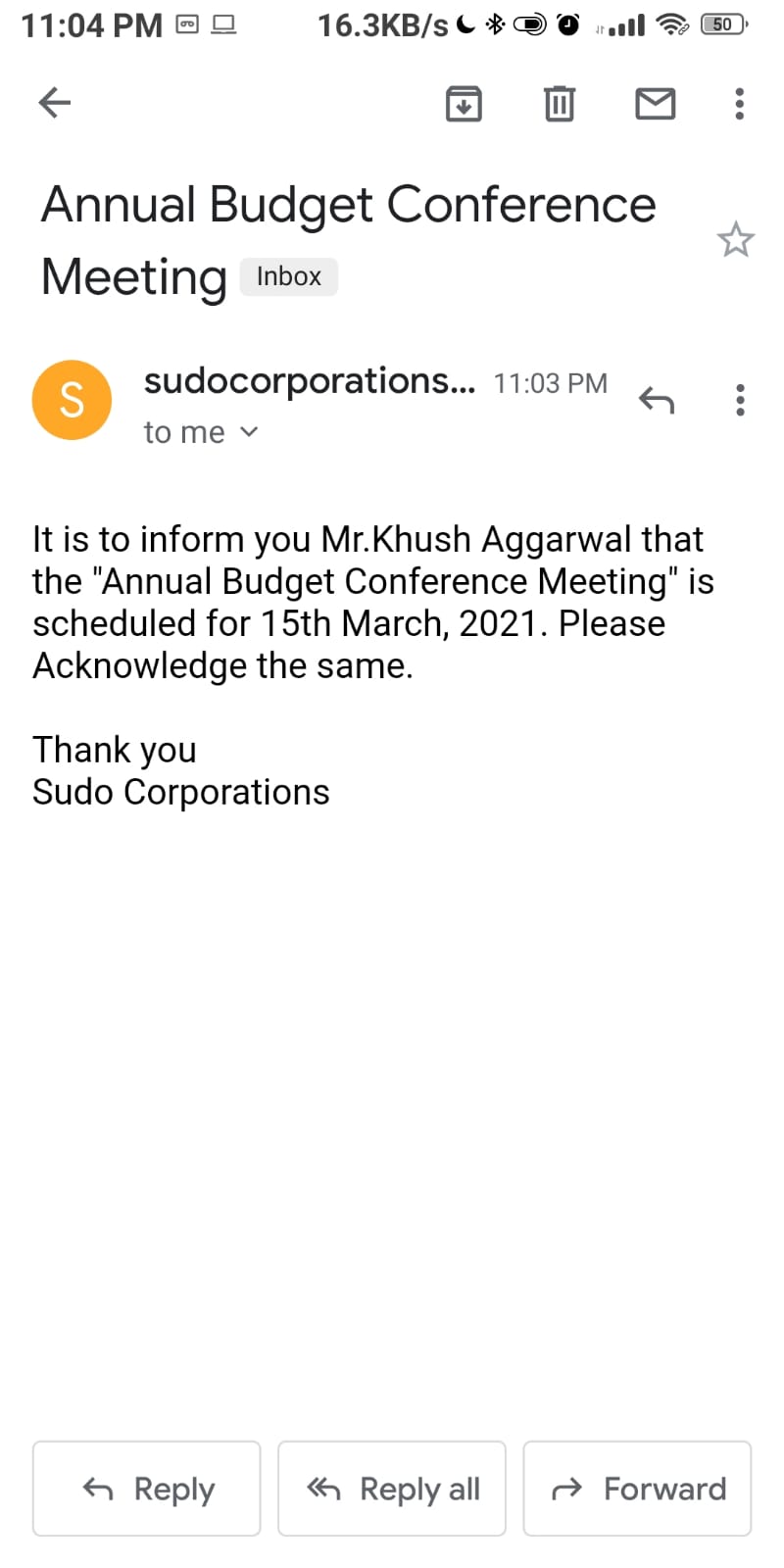
**Deleting record in database.**



**Searching record in database.**

**Notification was also displayed with a sound.**

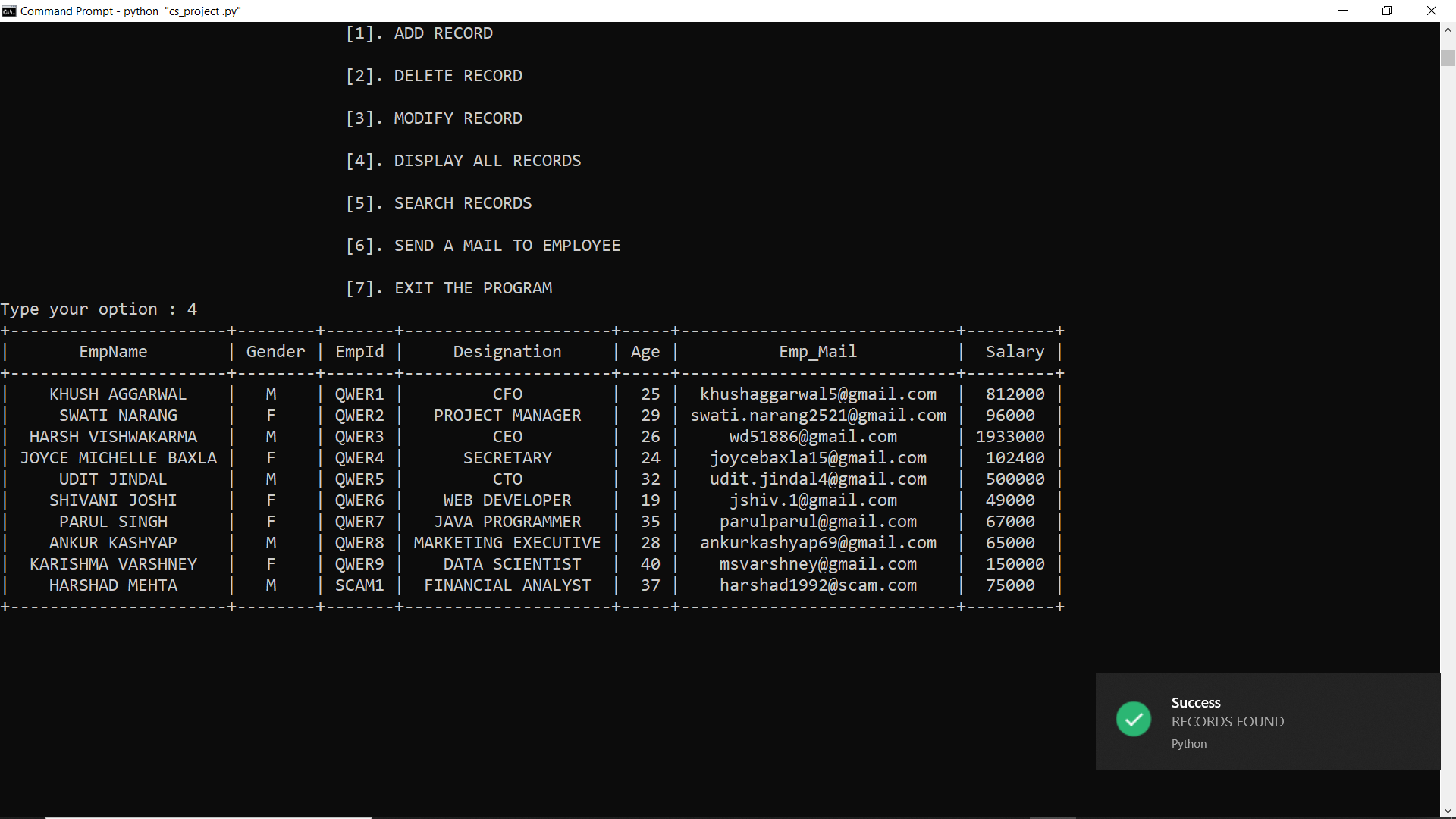
Sending Mail to Employee



**E-mail sent successfully!**

**Notification was also displayed with a sound.**

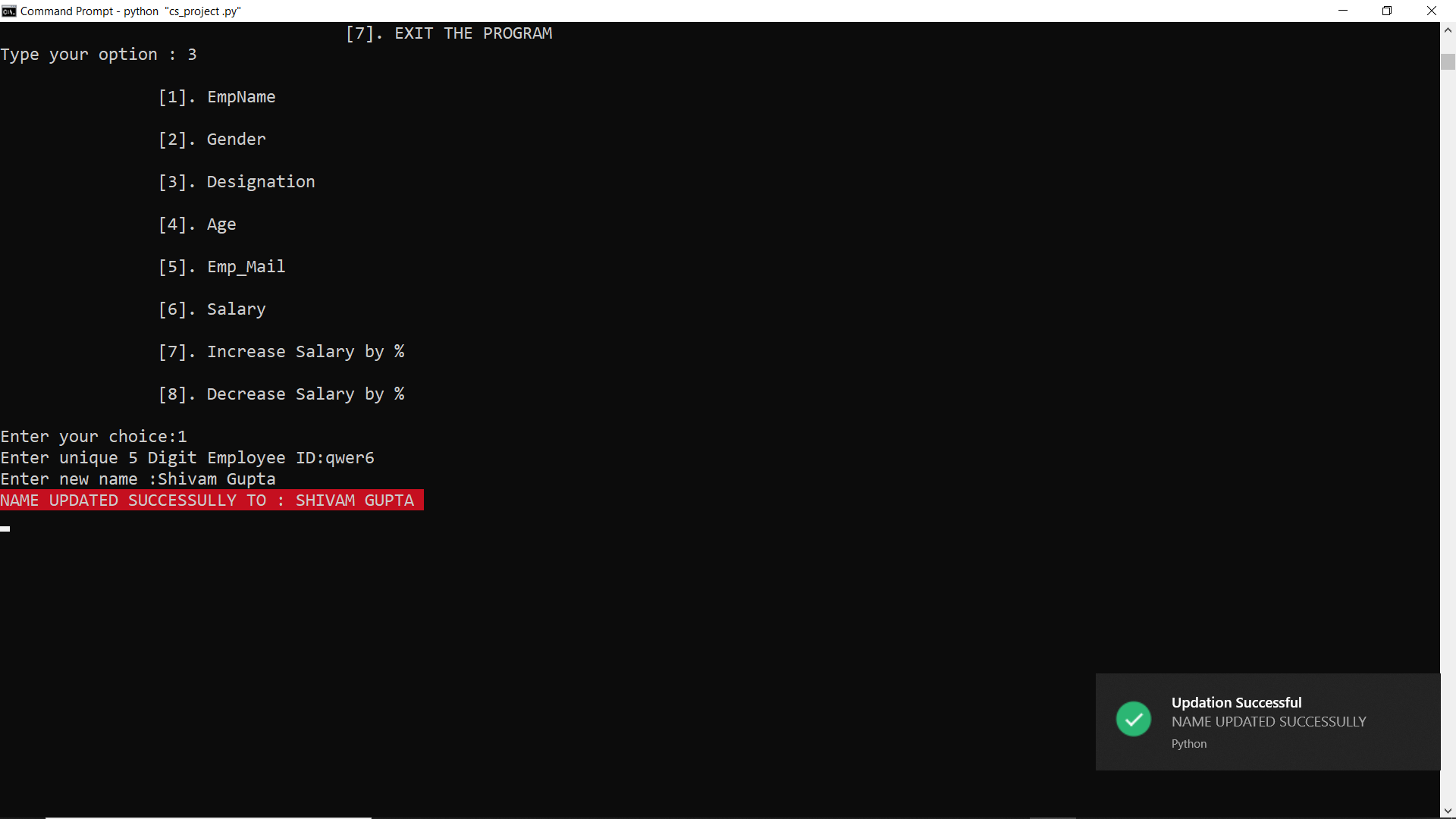
Modifiying records in database.

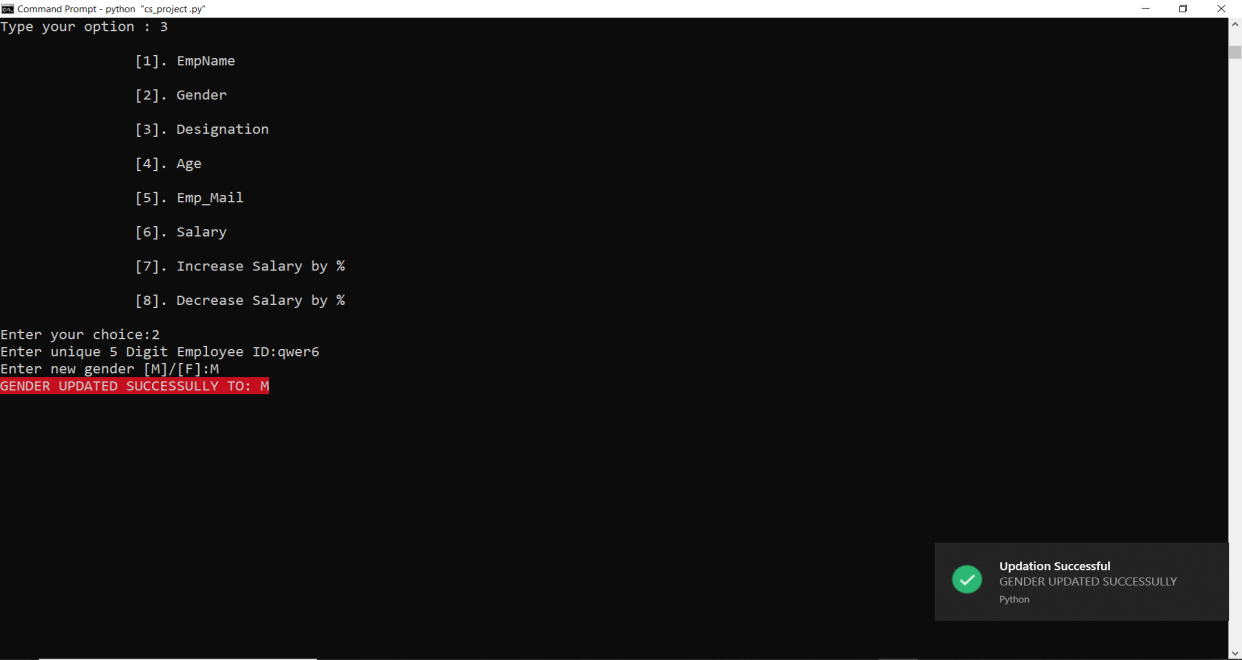
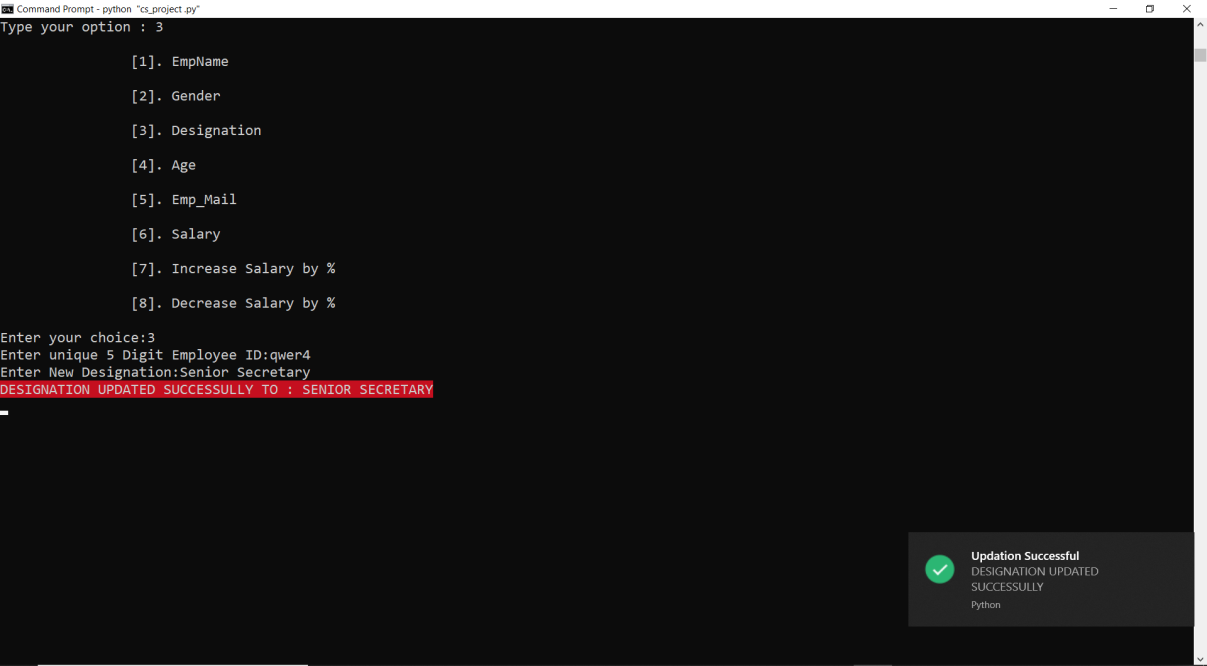




**Modification Menu**

**Table Before Modification of records**

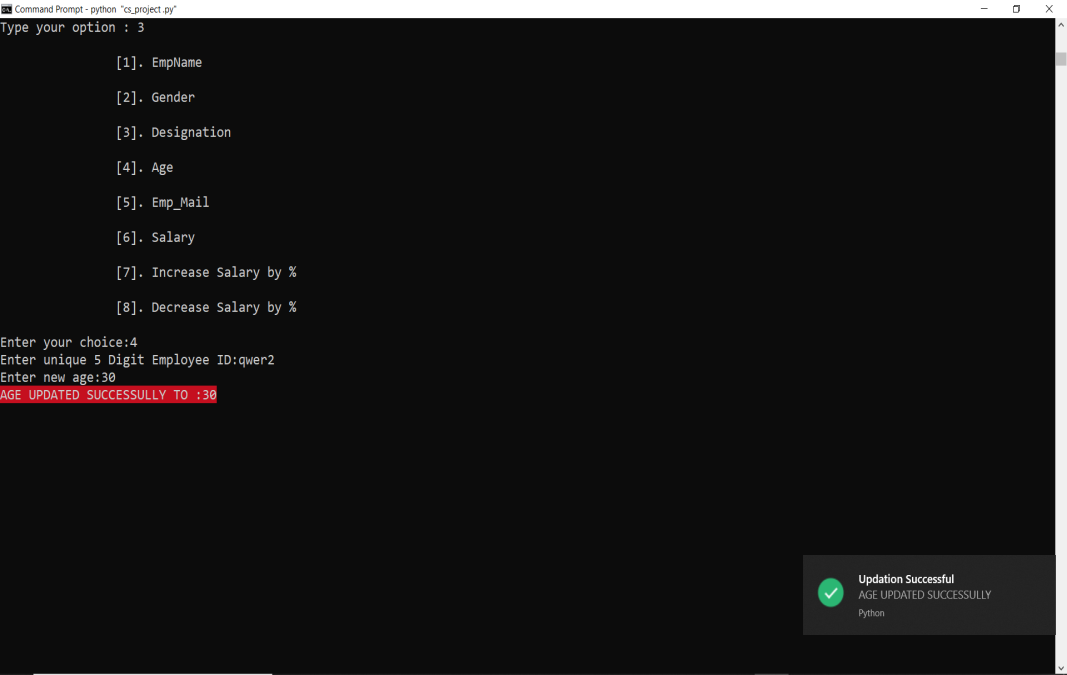


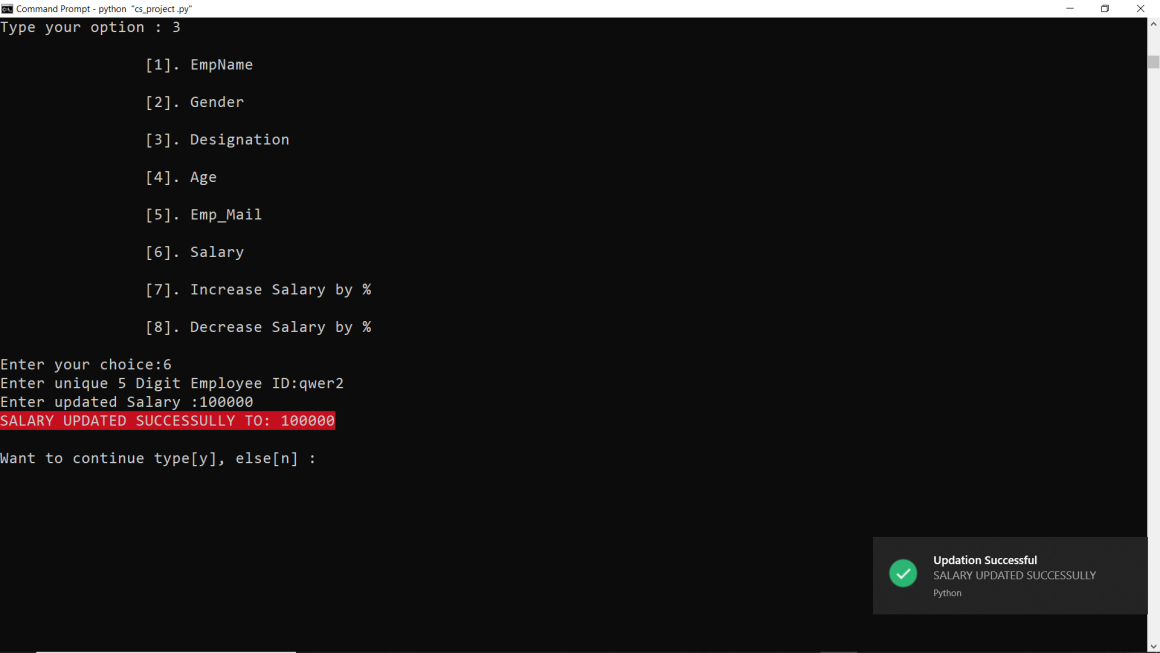
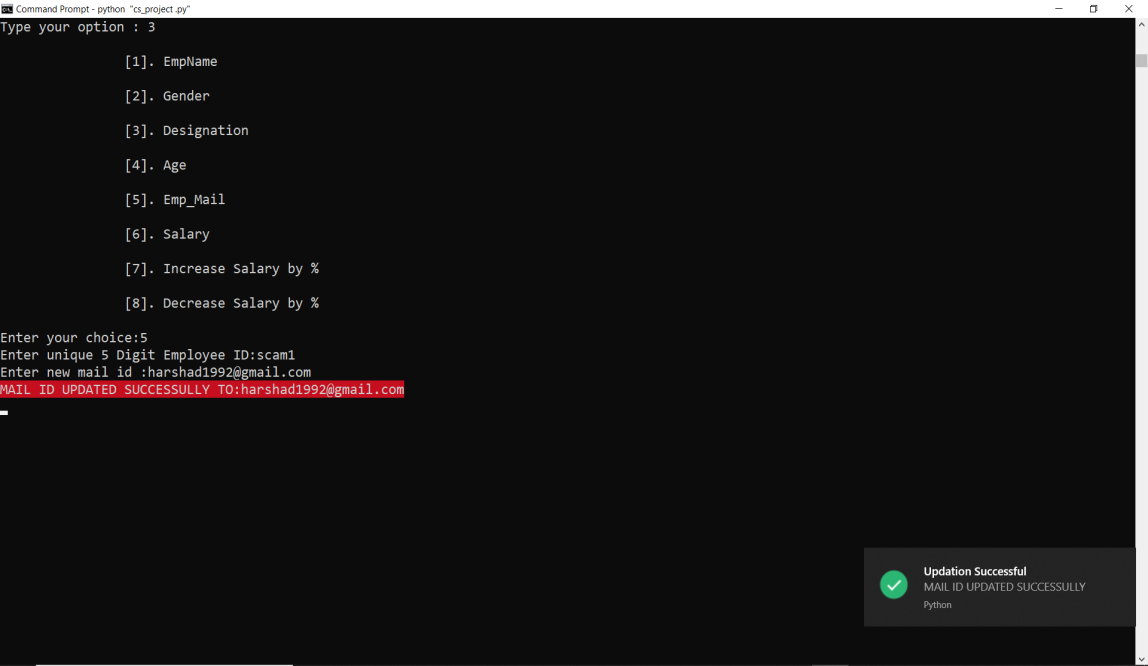


**Modifying Employee Designation**

**Modifying Employee Gender**

**Modifying Employee Name**

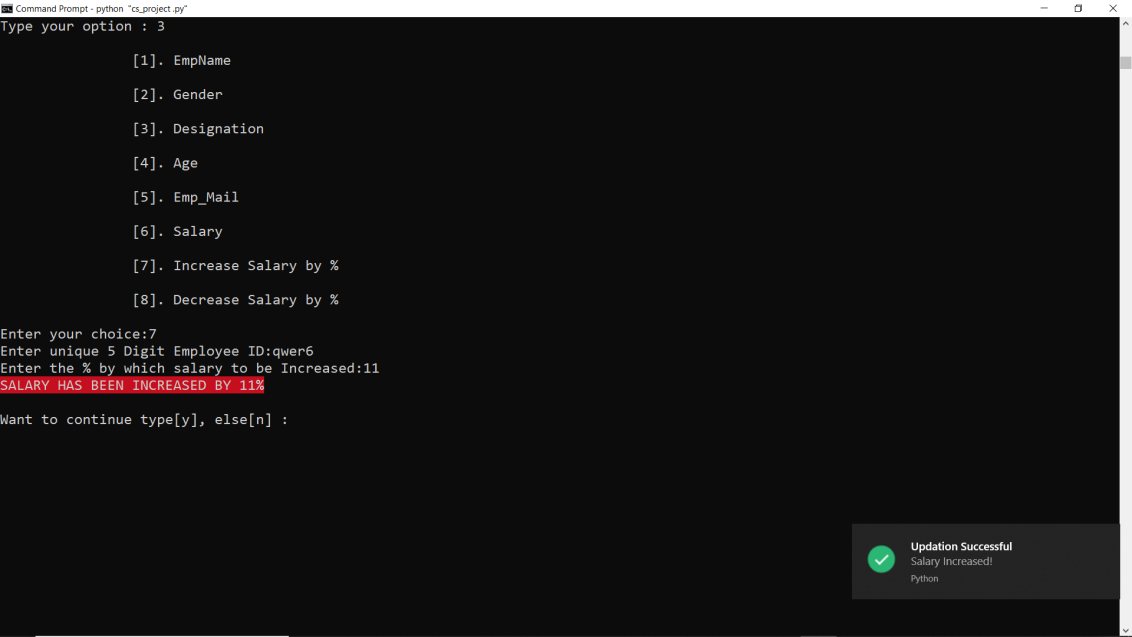


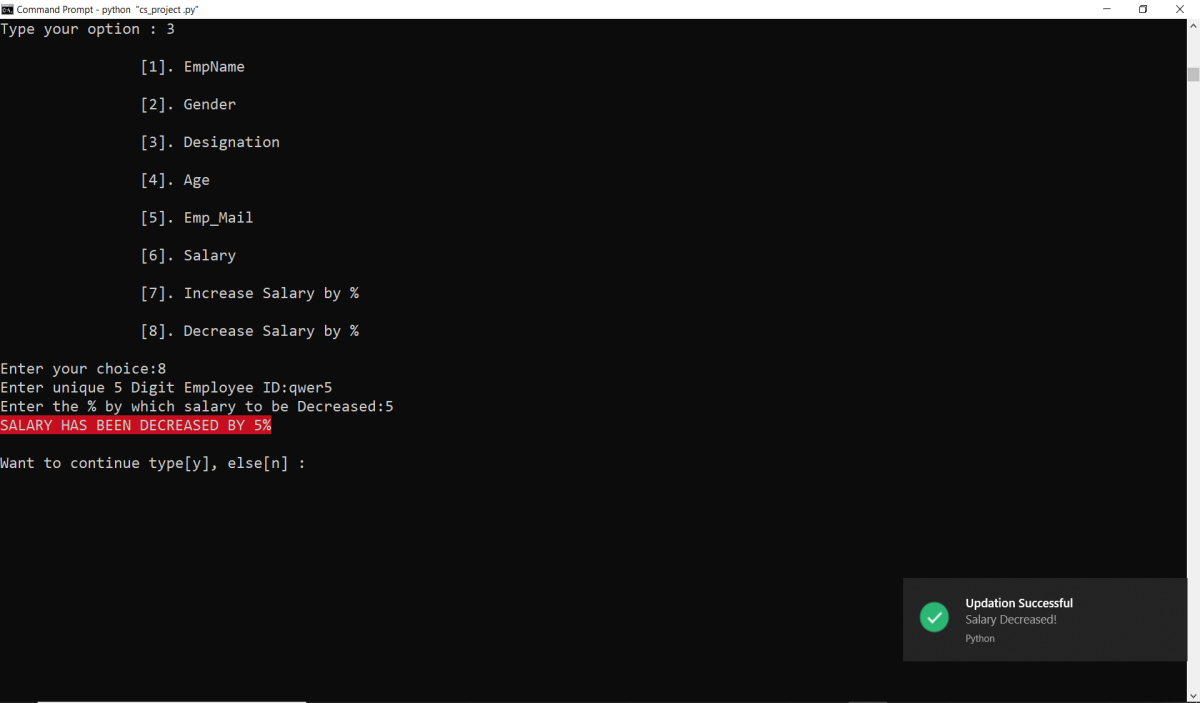
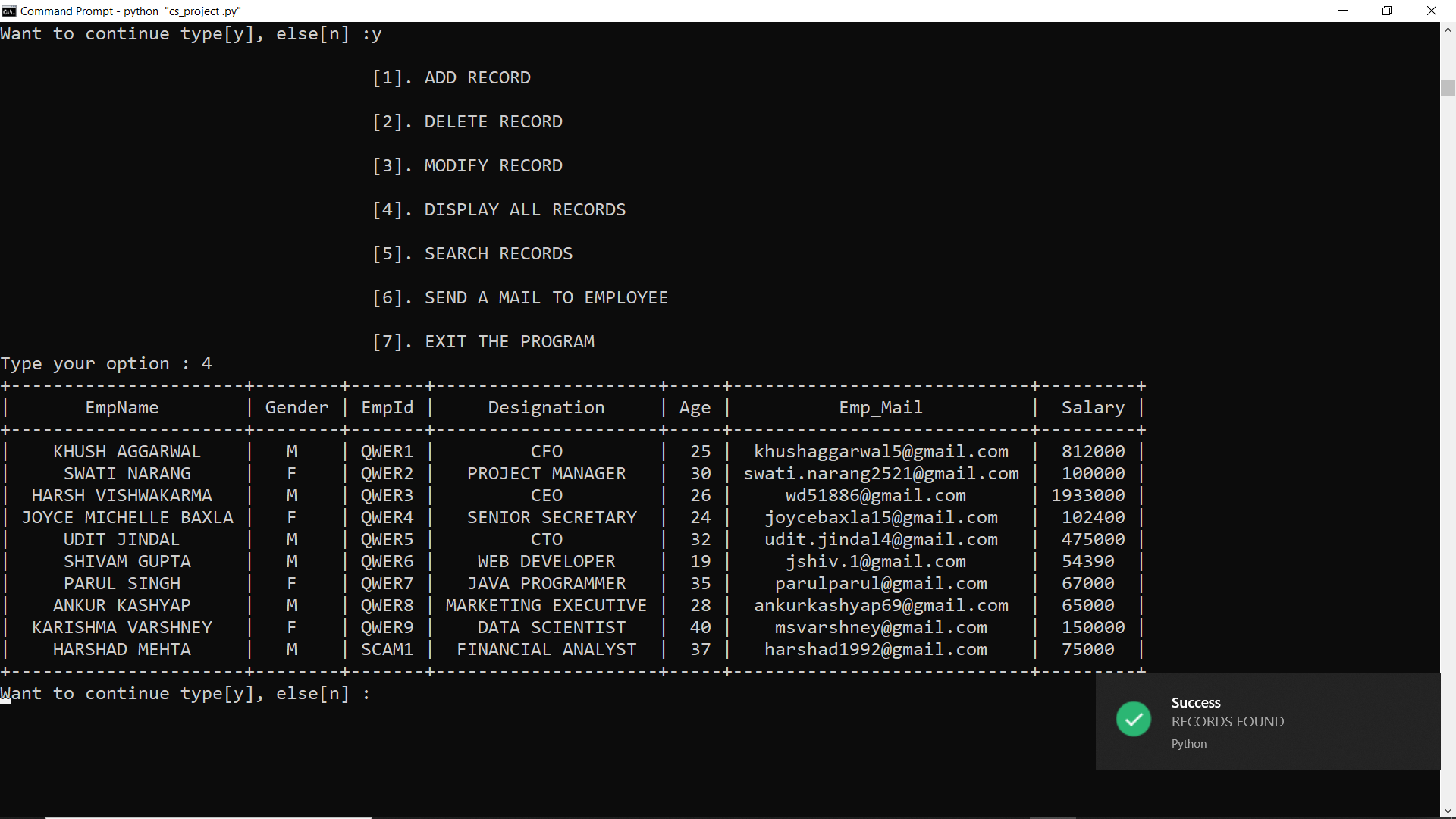


**Modifying Employee Salary**

**Modifying Employee Email**

**Modifying Employee Age**



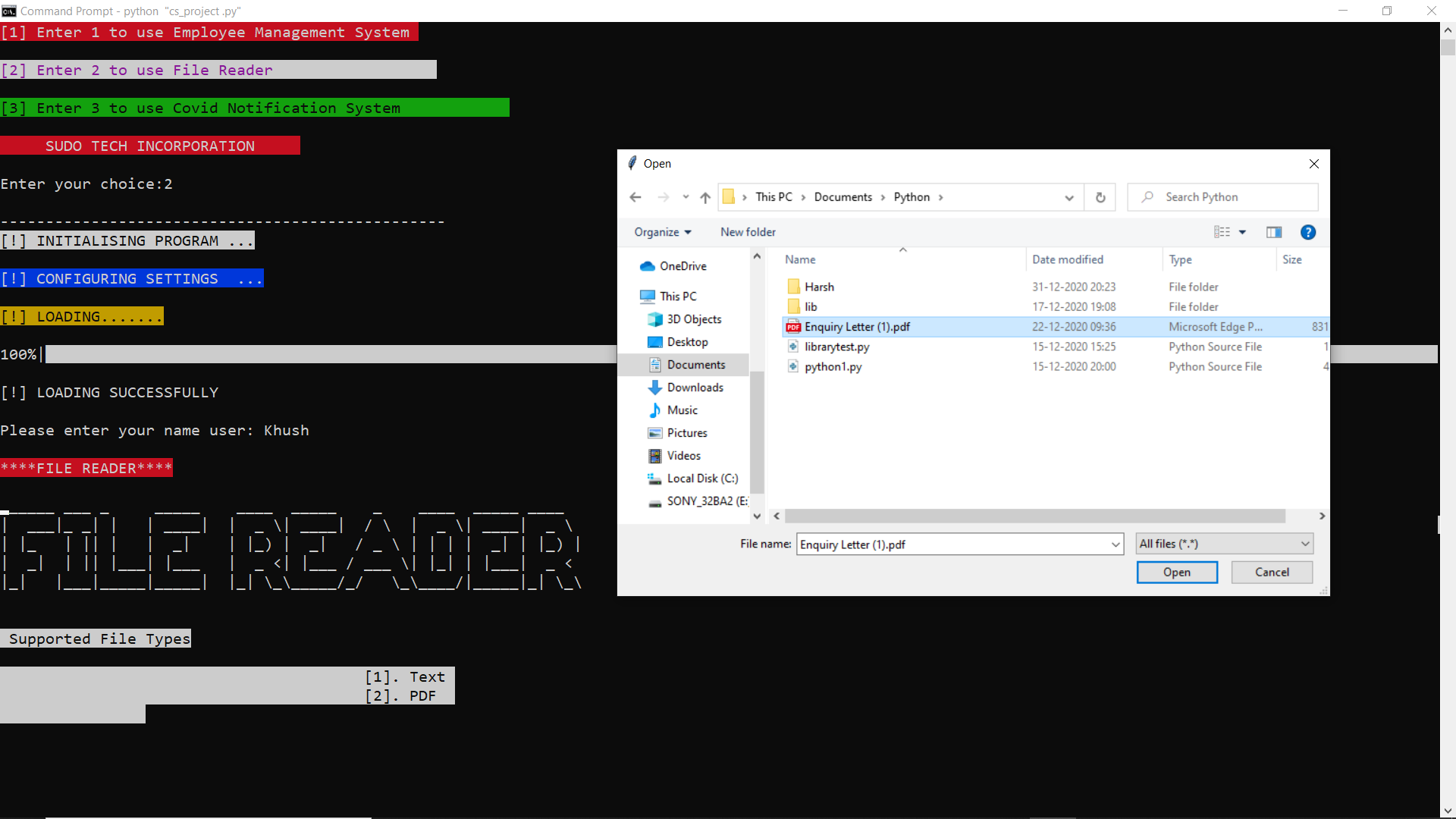


**Table after doing above modifications**

**Decreasing Employee Salary**

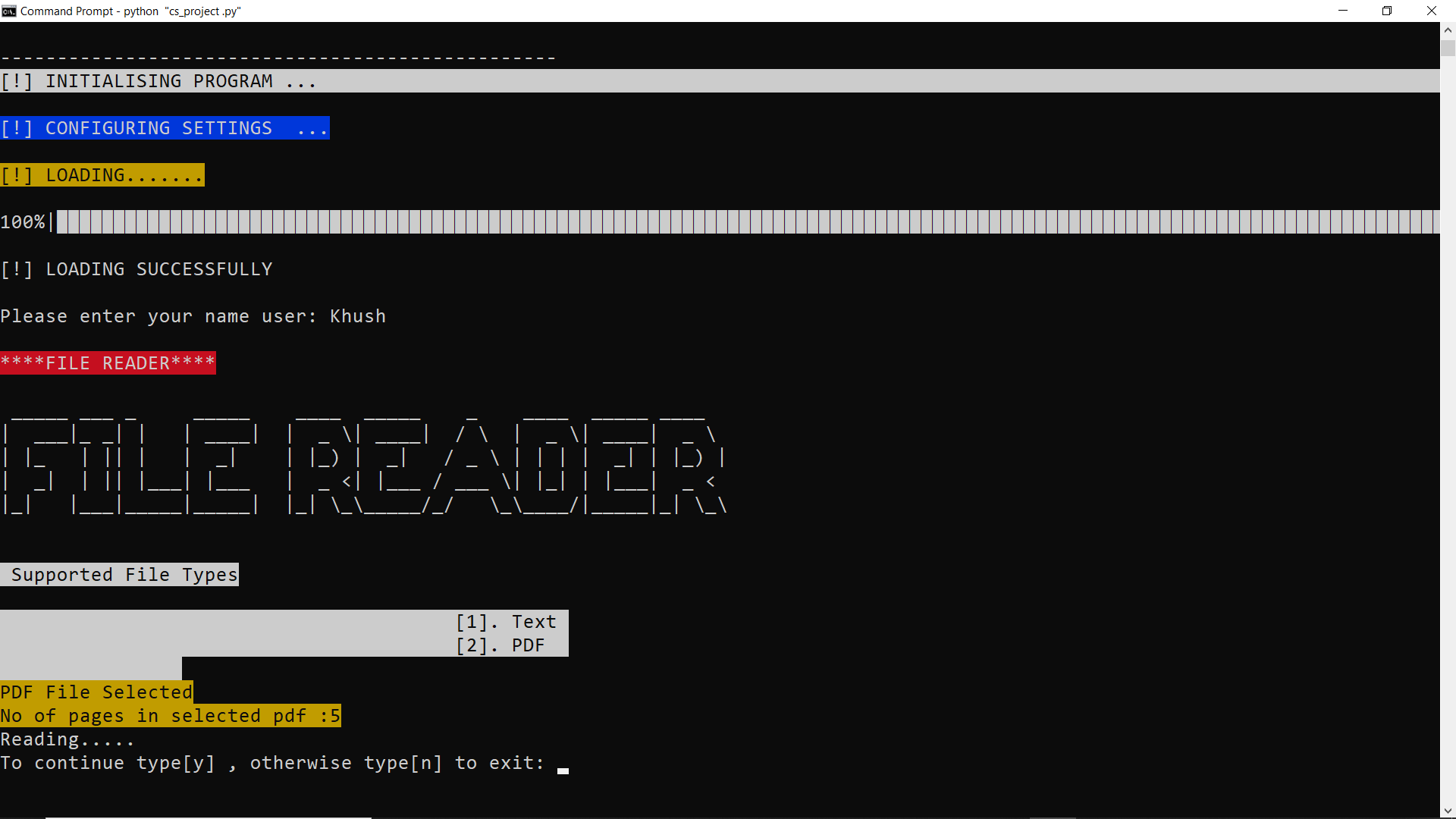
**Increasing Employee Salary**



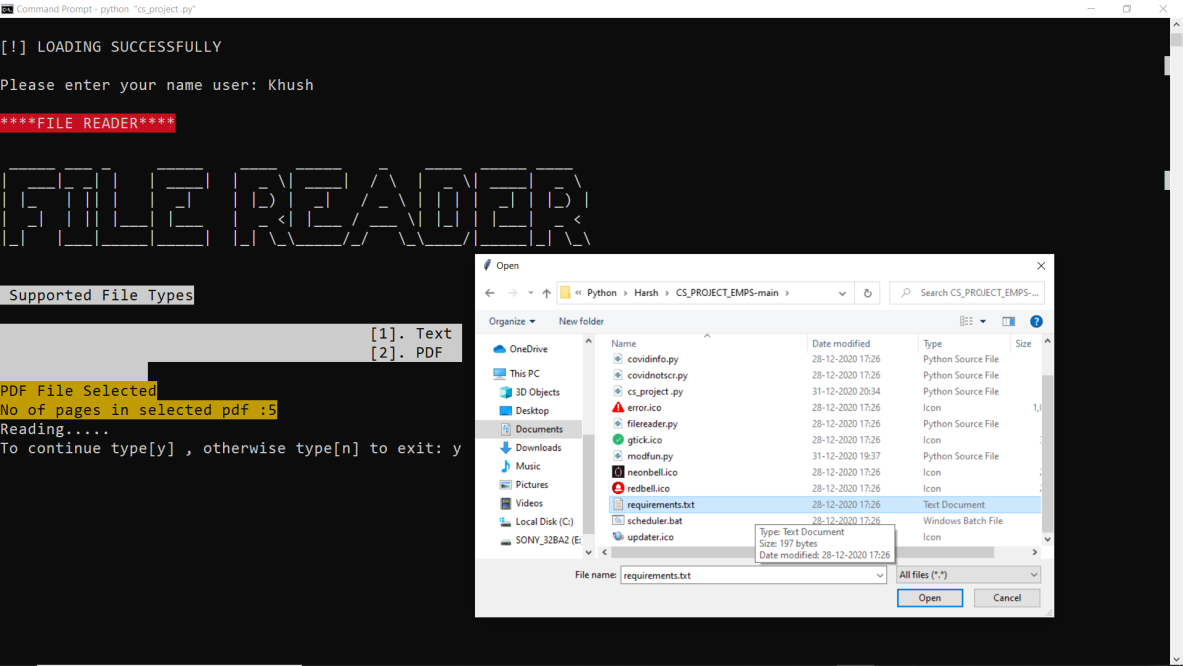
File Reader

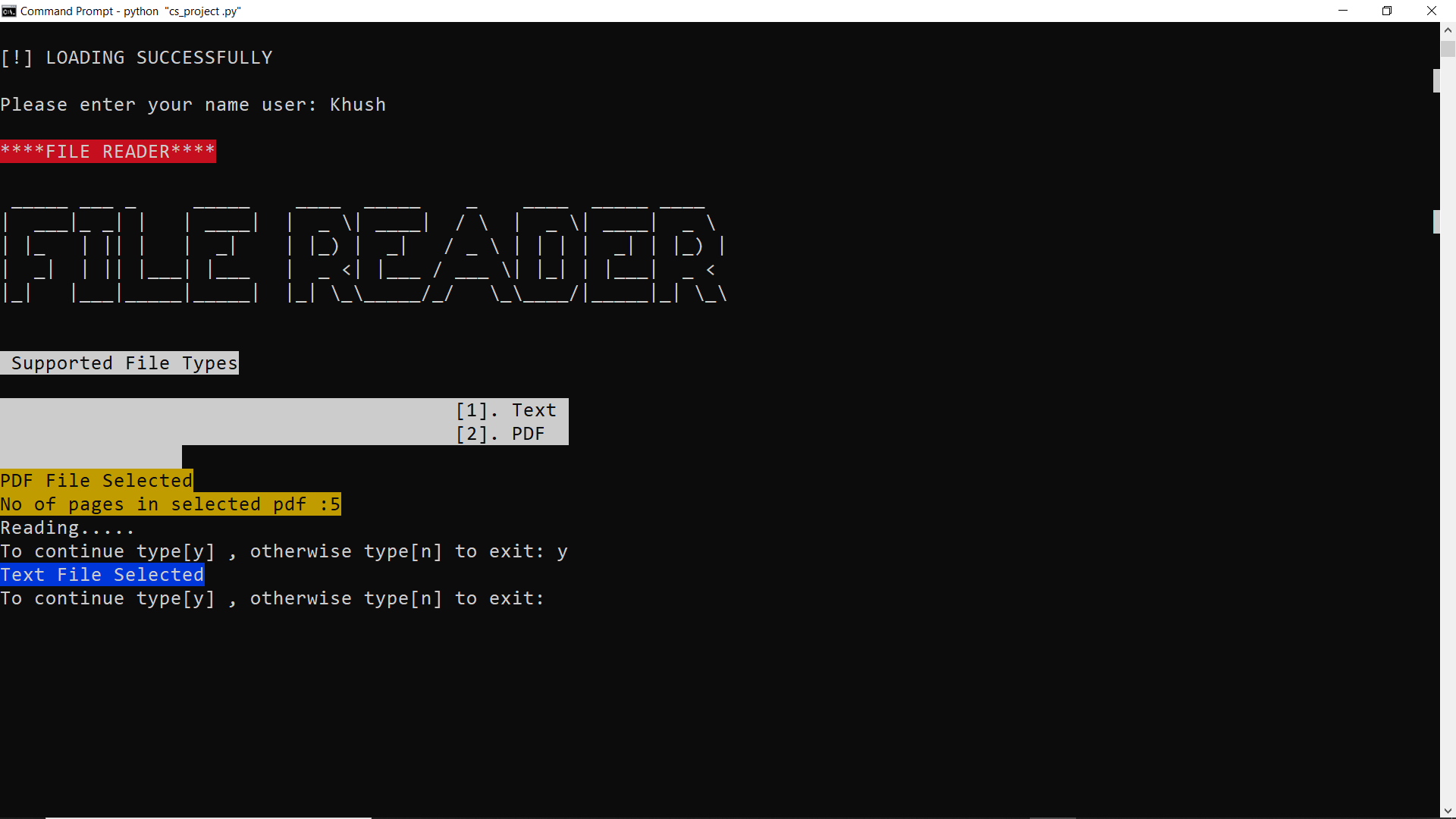
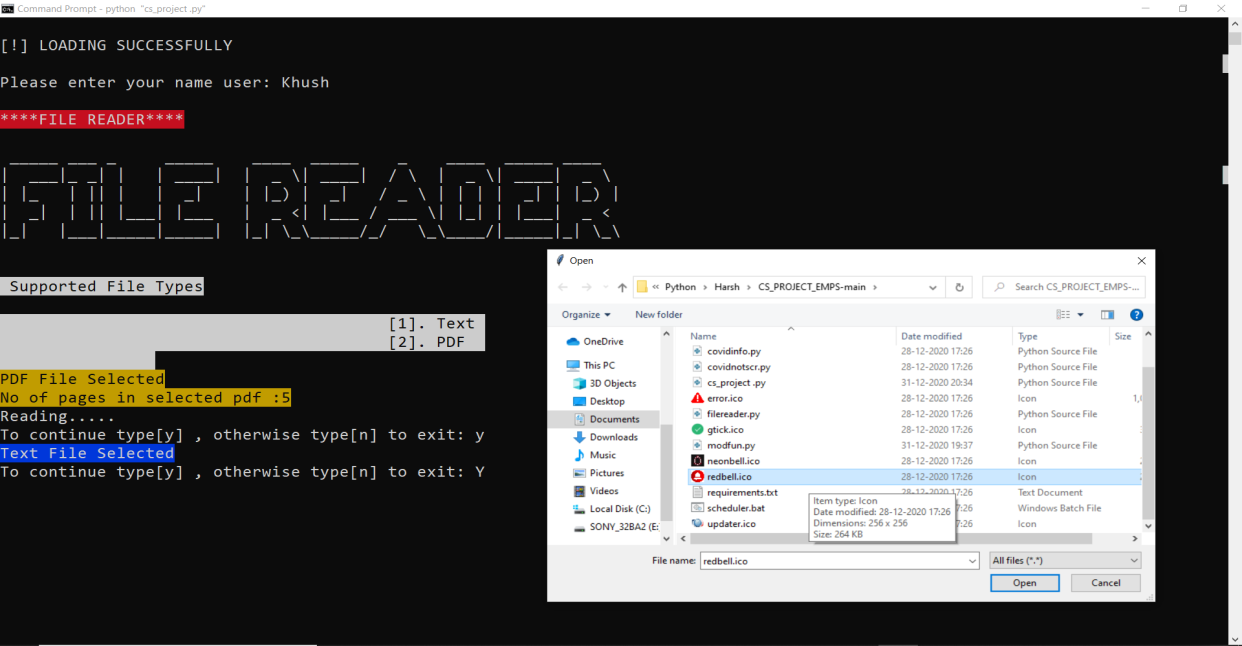
**Exit**

**Selecting Pdf File from System**



**PDF File selected and was read by computer**

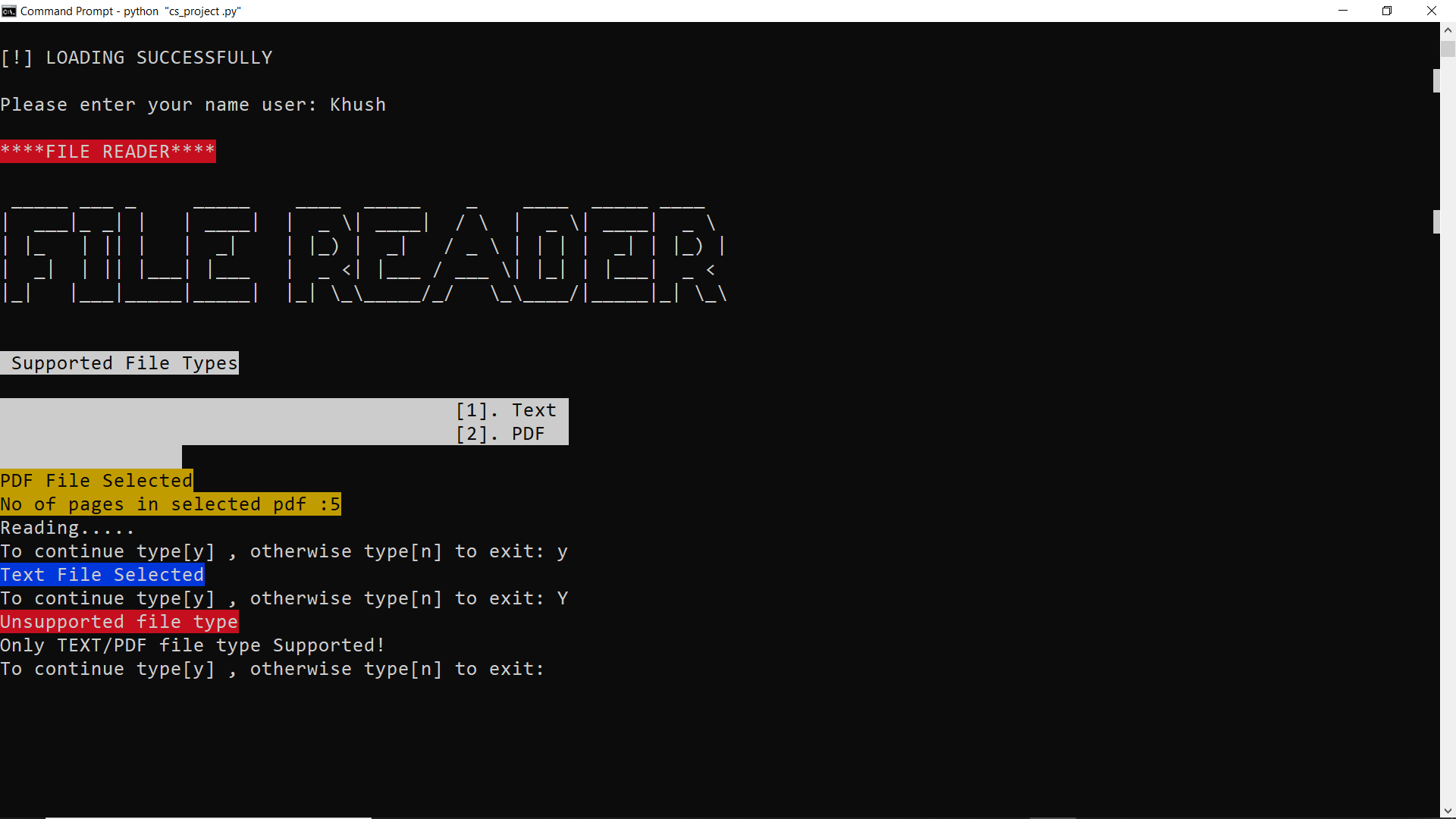


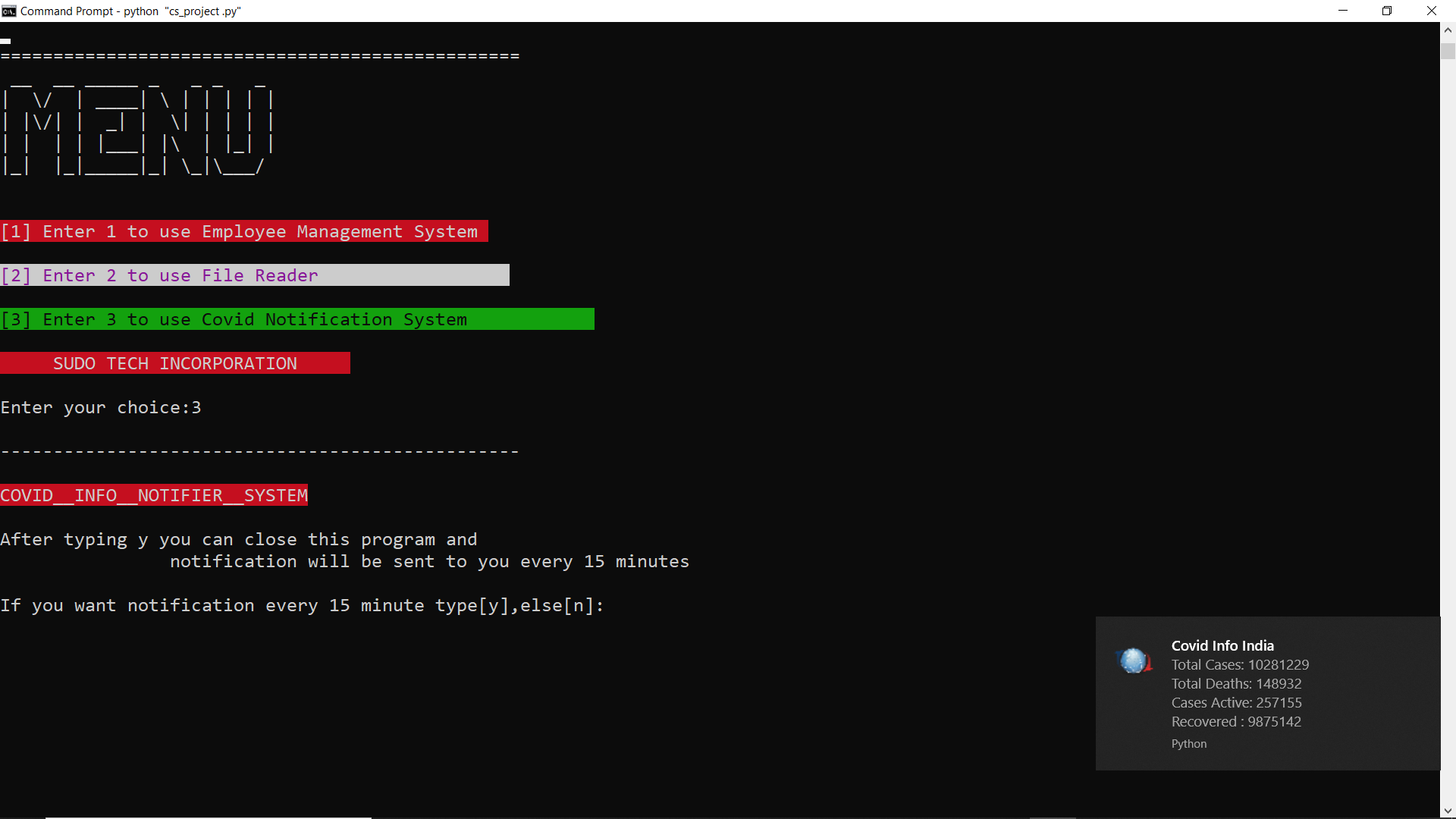


**Unsupported file (.ico) has been selected**

**Text File Selected and was read by computer**

**Selecting Text File from System**

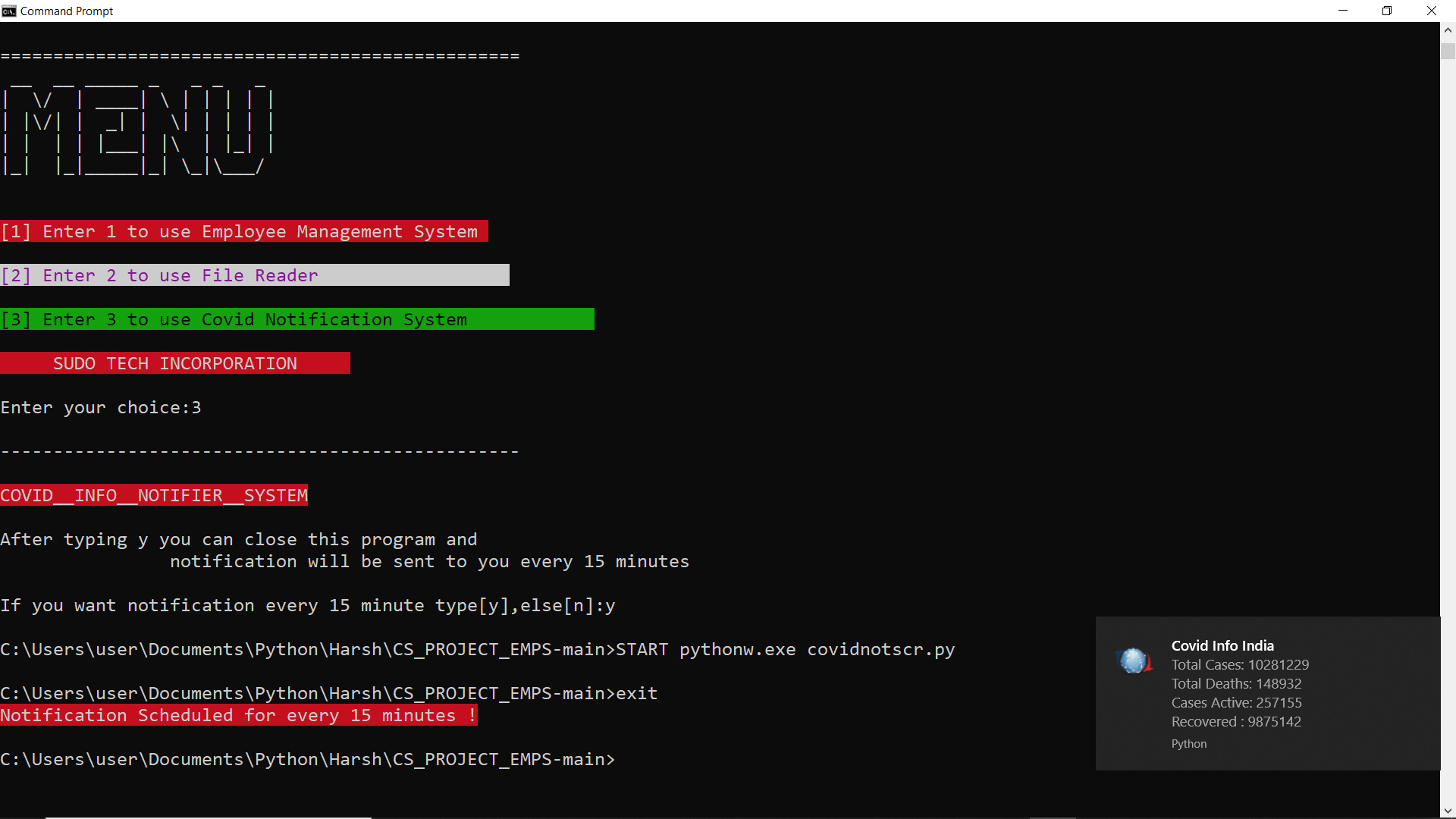




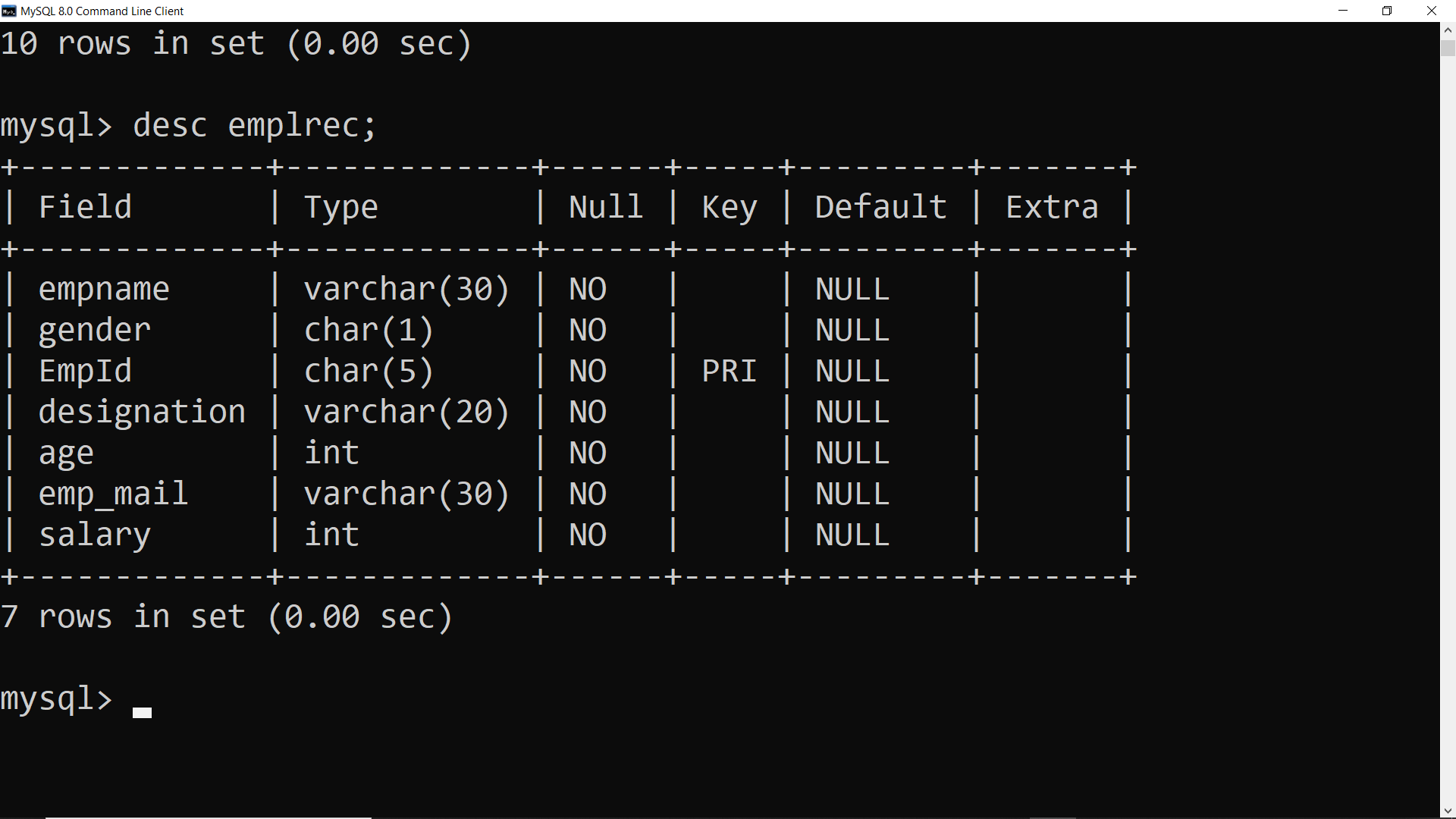
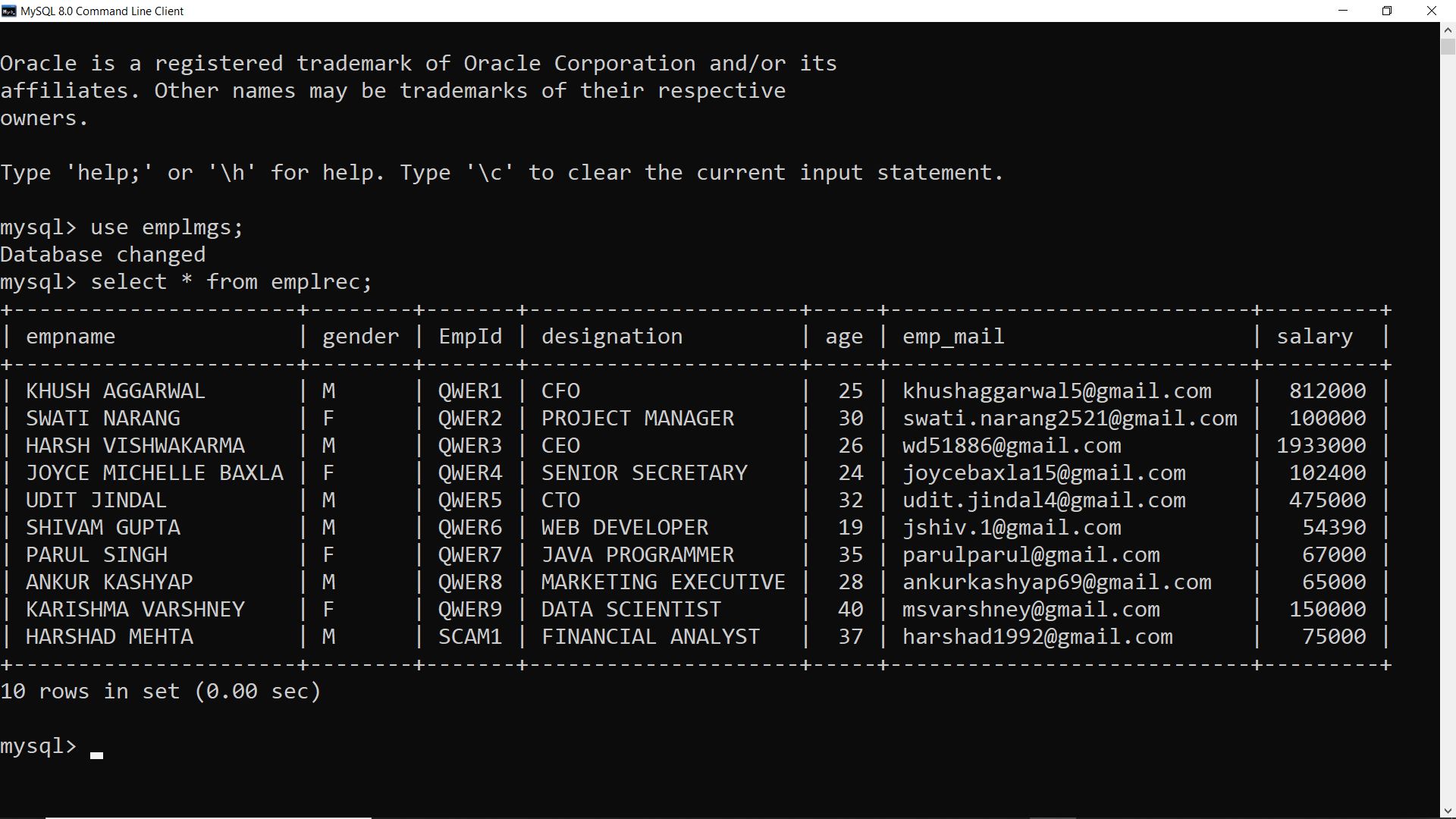
**Notification has been sent on the system regarding the corona virus updates in INDIA.**

**CORONAVIRUS NOTIFICATION SYSTEM**

**Unsupported file type error message displayed .**



**Push-notifications scheduled for every 15 minutes .**

Database and Tables used in MYSQl

**STRUCTURE OF TABLE**

**TABLE: EMPLREC**

BIBILIOGRAPHY

* COMPUTER SCIENCE SUMITA ARORA CLASS XI
* COMPUTER SCIENCE SUMITA ARORA CLASS XII