PROBLEM DEFINITION:-

Every policy agent has to keep the record of their clients because they have to send reminder to their clients, when the renewal date of their client policy arrives. It's vital for the agent because if the client forgets to pay the premium he or she might not get the advantage of the policy and the agent will lose the client .And its not easy to remember the renewal date of all the clients .So in order to tackle this problem ,we have built a software which helps the agents to tackle this problem.

Requirements:- PYTHON,SQL,TKINTER

PROPOSED SOLUTION:-

The software helps the agent to save the details of the policy sold by them In a database . The software will loop through the database daily when the system boots up and check the entries whose renewal are about to due and send an autogenerated email to the agent containing an excel sheet Which contains their details regarding the policy whose renewal dates are about to due .Apart from this the software also helps the agent to search the database with various attributes . So rather than worrying about the complexity of using database we have created an abstract software to get rid of the complexity.

Implementation:

autogenerated.py:-

```
import datetime
import os
import sqlite3
import sys
import tkinter

import pandas as pd
import numpy as np
import shelve



import smtplib
import mimetypes
from email.mime.multipart import MIMEMultipart
from email import encoders
from email.mime.audio import MIMEAudio
from email.mime.base import MIMEBase
from email.mime.image import MIMEImage
from email.mime.text import MIMEText
import socket

def checkInternetSocket(host="8.8.8.8", port=53, timeout=3):
    try:
        socket.setdefaulttimeout(timeout)
        socket.socket(socket.AF_INET, socket.SOCK_STREAM).connect((host, port))
        return True
    except socket.error as ex:
        print(ex)
        return False

def daily():
    todays_date=datetime.datetime.now()
    print(os.path.dirname(os.getcwd())+r"\GUI\LIFEINSURANCE.sqlite")
    conn=sqlite3.connect(os.path.dirname(os.getcwd())+r"\GUI\LIFEINSURANCE.sqlite")
    cursor=conn.cursor()
    entry = None
    cursor = conn.cursor()
    try:
        print(cursor.execute(
            "SELECT RENEWAL_DATE,POLICY_NUMBER FROM LIFEINSURANCE
").fetchall())
        entry = cursor.execute(
            "SELECT RENEWAL_DATE,POLICY_NUMBER FROM LIFEINSURANCE ",
).fetchall()
        policy_number_list=[]
    except sqlite3.OperationalError:
        return
    else:
        for selected_policy in entry:
            renewal_date = datetime.datetime.strptime(selected_policy[0], '%Y-%m-%d
%H:%M:%S')
            if datetime.datetime.strftime(todays_date+datetime.timedelta(days=15),"%Y-
%m-%d") == datetime.datetime.strftime(renewal_date,"%Y-%m-%d") or
datetime.datetime.strftime(todays_date+datetime.timedelta(days=5),"%Y-%m-%d") ==
datetime.datetime.strftime(renewal_date,"%Y-%m-%d"):
                policy_number_list.append(selected_policy[1])
        print(policy_number_list)
        if policy_number_list == []:
            return
        else:
            entries=[]
            for policy_number in policy_number_list:
                statement = cursor.execute(
                    "SELECT
RENEWAL_DATE,CUSTOMER_NAME,PREMIUM_AMOUNT,POLICY_NUMBER
FROM LIFEINSURANCE WHERE POLICY_NUMBER = ?",
                    (policy_number,)).fetchone()
                entries.append(list(statement))
            dateframe = pd.DataFrame(np.array(entries),
columns=["RENEWAL_DATE","CUSTOMER_NAME","PREMIUM_AMOUNT","POLICY
_NUMBER"])
            print(dateframe)
            dateframe.to_csv("report.csv")
            with open("report.csv","rb") as f:
                f_data=f.read()
            s=shelve.open(os.path.dirname(os.getcwd())+r"\GUI\login_details.shelve")
```

```
email_id=s["EMAIL"]
password=s["PASSWORD"]
s.close()
emailfrom = email_id
emailto = email_id
fileToSend = "report.csv"
password = password

msg = MIMEMultipart()
msg["From"] = emailfrom
msg["To"] = emailto
msg["Subject"] = "RENEWAL ALERTS"


ctype, encoding = mimetypes.guess_type(fileToSend)
if ctype is None or encoding is not None:
    ctype = "application/octet-stream"

maintype, subtype = ctype.split("/", 1)

if maintype == "text":
    fp = open(fileToSend)
    # Note: we should handle calculating the charset
    attachment = MIMEText(fp.read(), _subtype=subtype)
    fp.close()
elif maintype == "image":
    fp = open(fileToSend, "rb")
    attachment = MIMEImage(fp.read(), _subtype=subtype)
    fp.close()
elif maintype == "audio":
    fp = open(fileToSend, "rb")
    attachment = MIMEAudio(fp.read(), _subtype=subtype)
    fp.close()
else:
    fp = open(fileToSend, "rb")
    attachment = MIMEBase(maintype, subtype)
    attachment.set_payload(fp.read())
    fp.close()
    encoders.encode_base64(attachment)
attachment.add_header("Content-Disposition", "attachment",
filename=fileToSend)
            msg.attach(attachment)
            if checkInternetSocket():
                server = smtplib.SMTP("smtp.gmail.com:587")
                server.starttls()
                server.login(email_id,password)
                server.sendmail(emailfrom, emailto, msg.as_string())
                server.quit()
                print("successful")
                sys.exit()
            else:
                window=tkinter.Tk()
                window.geometry("600x300")
                window.title("NO INTERNET CONNECTION")
                tkinter.Label(window,text="ALERTS WEREN'T
CHECKED",font="Courier").grid(row=0,column=0)
                tkinter.Label(window,text="CONNECT TO THE INTERNET AND TRY AGAIN
OR EXIT",font="Courier").grid(row=1,column=0)
                tkinter.Button(window,text="TRY
AGAIN",font="Courier",command=daily).grid(row=2,column=0)
tkinter.Button(window,text="EXIT",font="Courier",command=sys.exit).grid(row=3,column
=0)
                window.mainloop()
daily()
```

Life_insurance_class.py:-

```
import datetime

mode = {
    "ANNUAL": 365,
    "HALF-YEARLY": 180,
    "QUARTERLY": 90,
    "MONTHLY": 30
}


class NewPolicy:
    def __init__(self, company: str, customer_name: str, email: str, contact_no: str,
address: str, nominee: str,
                 policy_status: str, policy_name: str, policy_number: str, issue_date:
```

```
datetime.date,
                maturity_date: datetime.date,
                premium_amount: int, sum_assured: int, policy_term: int,
premium_paying_term: int,
                payment_mode: str, first_year_commission_percent: int,
renewal_commission_for_2_3_year_percent: int,
                renewal_commission_for_4_5_year_percent: int,
renewal_commission_for_6_year_onwards_percent: int,
                gst_on_first_year_commission: int, gst_on_renewal_commission: int):
        self.company = company
        self.customer_name = customer_name
        self.email = email
        self.contact_no = contact_no
        self.address = address
        self.nominee = nominee
        self.policy_status = policy_status
        self.policy_name = policy_name
        self.policy_number = policy_number
        self.issue_date = issue_date
        self.maturity_date = maturity_date
        self.premium_amount = premium_amount
        self.sum_assured = sum_assured
        self.policy_term = policy_term
        self.premium_paying_term = premium_paying_term
        self.payment_mode = payment_mode
        self.first_year_commission_percent = first_year_commission_percent
        self.renewal_commission_for_2_3_year_percent =
renewal_commission_for_2_3_year_percent
        self.renewal_commission_for_4_5_year_percent =
renewal_commission_for_4_5_year_percent
        self.renewal_commission_for_6_year_onwards_percent =
renewal_commission_for_6_year_onwards_percent
        self.gst_on_first_year_commission = gst_on_first_year_commission
        self.gst_on_renewal_commission = gst_on_renewal_commission
        self.renewal_date = self.issue_date +
datetime.timedelta(days=mode[self.payment_mode])
        self.first_year_commission = (self.first_year_commission_percent/ 100) *
(self.premium_amount - (self.premium_amount *
                                                                                          (
self.gst_on_first_year_commission / 100)))
```

```
        self.renewal_commission_for_2_3_year =
(self.renewal_commission_for_2_3_year_percent / 100) * (
                self.premium_amount - (self.premium_amount *
                                       (self.gst_on_renewal_commission / 100)))
        self.renewal_commission_for_4_5_year =
(self.renewal_commission_for_4_5_year_percent / 100) * (
                self.premium_amount - (self.premium_amount *
                                       (self.gst_on_renewal_commission / 100)))
        self.renewal_commission_for_6_year_onwards =
(self.renewal_commission_for_6_year_onwards_percent / 100) * (
                self.premium_amount - (self.premium_amount *
                                       (self.gst_on_renewal_commission / 100)))

        self.note=""

    # def update_renewal(self):
    #     if datetime.date.today > self.renewal_date and self.renewal_status:
    #         self.renewal_date = self.renewal_date +
datetime.timedelta(days=mode[self.payment_mode])
    #         self.renewal_status = False
    #
    # def add_renewal_status(self, renewal_date: datetime.datetime):
    #     self.premium_paid.append(renewal_date)
    #     self.renewal_status = True
```

console.py:-

```
from Life_insurance_class import NewPolicy, mode
import sqlite3
import datetime
import pandas as pd

conn = sqlite3.connect("LIFEINSURANCE.sqlite")
conn.execute(
    "CREATE TABLE IF NOT EXISTS LIFEINSURANCE (COMPANY TEXT,
CUSTOMER_NAME TEXT, EMAIL TEXT, CONTACT_NO TEXT, "
    "ADDRESS TEXT, NOMINEE TEXT, "       +
    "POLICY_STATUS TEXT, POLICY_NAME TEXT, POLICY_NUMBER TEXT,
ISSUE_DATE TEXT,MATURITY_DATE TEXT,PREMIUM_AMOUNT "
    "INTEGER, SUM_ASSURED INTEGER, "
    "POLICY_TERM INTEGER, PREMIUM_PAYING_TERM
```

```
INTEGER,PAYMENT_MODE INTEGER, FIRST_YEAR_COMMISSION_PERCENT
INTEGER, "
   "RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT
INTEGER,RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT INTEGER, "
   "RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT
INTEGER,GST_ON_FIRST_YEAR_COMMISSION INTEGER, "
   "GST_ON_RENEWAL_COMMISSION INTEGER, "
   "RENEWAL_DATE TEXT,"
   "FIRST_YEAR_COMMISSION INTEGER,"
   "RENEWAL_COMMISSION_FOR_2_3_YEAR INTEGER,"
   "RENEWAL_COMMISSION_FOR_4_5_YEAR INTEGER,"
   "RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS INTEGER,"
   "NOTE TEXT)")


def create_an_entry_in_the_database(policy_info: dict):
   policy_to_be_added = NewPolicy(str(policy_info["COMPANY"]),
policy_info["CUSTOMER_NAME"], policy_info["EMAIL"],
                           policy_info["CONTACT_NO"], policy_info["ADDRESS"],
policy_info["NOMINEE"],
                           policy_info["POLICY_STATUS"], policy_info["POLICY_NAME"],
                           policy_info["POLICY_NUMBER"],
                           policy_info["ISSUE_DATE"],
                           policy_info["MATURITY_DATE"],
                           policy_info["PREMIUM_AMOUNT"],
                           policy_info["SUM_ASSURED"], policy_info["POLICY_TERM"],
                           policy_info["PREMIUM_PAYING_TERM"],
                           policy_info["PAYMENT_MODE"],
policy_info["FIRST_YEAR_COMMISSION_PERCENT"],
policy_info["RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT"],
policy_info["RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT"],
policy_info["RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT"],
                           policy_info["GST_ON_FIRST_YEAR_COMMISSION"],
                           policy_info["GST_ON_RENEWAL_COMMISSION"]
                           )
   conn=sqlite3.connect("LIFEINSURANCE.sqlite")
   cursor = conn.cursor()
   cursor.execute("INSERT  INTO LIFEINSURANCE (COMPANY , CUSTOMER_NAME
, EMAIL , CONTACT_NO , "
            "ADDRESS , NOMINEE , "
```

```
                "POLICY_STATUS , POLICY_NAME , POLICY_NUMBER , ISSUE_DATE
,MATURITY_DATE ,PREMIUM_AMOUNT "
                ", SUM_ASSURED , "
                "POLICY_TERM , PREMIUM_PAYING_TERM ,PAYMENT_MODE ,
FIRST_YEAR_COMMISSION_PERCENT , "
                "RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT
,RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT , "
                "RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT
,GST_ON_FIRST_YEAR_COMMISSION , "
                "GST_ON_RENEWAL_COMMISSION , "
                "RENEWAL_DATE ,"
                "FIRST_YEAR_COMMISSION ,"
                "RENEWAL_COMMISSION_FOR_2_3_YEAR ,"
                "RENEWAL_COMMISSION_FOR_4_5_YEAR ,"
                "RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS ,"
                "NOTE ) VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)",
                (policy_to_be_added.company,
                 policy_to_be_added.customer_name,
                 policy_to_be_added.email,
                 str(policy_to_be_added.contact_no),
                 policy_to_be_added.address,
                 policy_to_be_added.nominee,
                 policy_to_be_added.policy_status,
                 policy_to_be_added.policy_name,
                 str(policy_to_be_added.policy_number),
                 str(policy_to_be_added.issue_date),
                 str(policy_to_be_added.maturity_date),
                 int(policy_to_be_added.premium_amount),
                 int(policy_to_be_added.sum_assured),
                 int(policy_to_be_added.policy_term),
                 int(policy_to_be_added.premium_paying_term),
                 str(policy_to_be_added.payment_mode),
                 int(policy_to_be_added.first_year_commission_percent),
                 int(policy_to_be_added.renewal_commission_for_2_3_year_percent),
                 int(policy_to_be_added.renewal_commission_for_4_5_year_percent),
int(policy_to_be_added.renewal_commission_for_6_year_onwards_percent),
                 int(policy_to_be_added.gst_on_first_year_commission),
                 int(policy_to_be_added.gst_on_renewal_commission),
                 str(policy_to_be_added.renewal_date),
int(policy_to_be_added.first_year_commission),
                int(policy_to_be_added.renewal_commission_for_2_3_year),
                int(policy_to_be_added.renewal_commission_for_4_5_year),
                int(policy_to_be_added.renewal_commission_for_6_year_onwards),
str(policy_to_be_added.note)))
    cursor.connection.commit()
    cursor.close()


def insert_to_database(path:str):
    frame = pd.read_excel(path)
    print(frame.index)
    print(dict(frame.iloc[0]))

    for j in frame.index:
        dictionary = dict(frame.iloc[j])
        create_an_entry_in_the_database(dictionary)



conn.commit()
conn.close()
```

gui.py:-

```
from tkinter import *
import sqlite3
import shelve
import pandas
from tkinter import messagebox, filedialog
from console import insert_to_database, mode
import datetime
import numpy
import subprocess
import os

# making a connection to the database

conn = sqlite3.connect("LIFEINSURANCE.sqlite")
main_window = Tk()
main_window.title("Renewal Alerts")
main_window.geometry("840x650")
```

```
# TO MAKE A TABLE NAMED LIFEINSURANCE IF ALREADY NOT EXIST BY WHICH
WE WILL AVOID TABLE NOT FOUND ERROR IF CLIENT
# TRIES TO DO OPERATION WITHOUT MAKING THE DATABASE
conn.execute(
    "CREATE TABLE IF NOT EXISTS LIFEINSURANCE (COMPANY TEXT,
CUSTOMER_NAME TEXT, EMAIL TEXT, CONTACT_NO TEXT, "
    "ADDRESS TEXT, NOMINEE TEXT, "
    "POLICY_STATUS TEXT, POLICY_NAME TEXT, POLICY_NUMBER TEXT,
ISSUE_DATE TEXT,MATURITY_DATE TEXT,PREMIUM_AMOUNT "
    "INTEGER, SUM_ASSURED INTEGER, "
    "POLICY_TERM INTEGER, PREMIUM_PAYING_TERM
INTEGER,PAYMENT_MODE INTEGER, FIRST_YEAR_COMMISSION_PERCENT
INTEGER, "
    "RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT
INTEGER,RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT INTEGER, "
    "RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT
INTEGER,GST_ON_FIRST_YEAR_COMMISSION INTEGER, "
    "GST_ON_RENEWAL_COMMISSION INTEGER, "
    "RENEWAL_DATE TEXT,"
    "FIRST_YEAR_COMMISSION INTEGER,"
    "RENEWAL_COMMISSION_FOR_2_3_YEAR INTEGER,"
    "RENEWAL_COMMISSION_FOR_4_5_YEAR INTEGER,"
    "RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS INTEGER,"
    "NOTE TEXT)")

label = Label(main_window, text="Welcome", padx=40, pady=20, fg="#000000",
font=("Helvetica", 30, "bold italic"))
label.pack()


# This function is called when button_gmail_details is pressed
def gmail_details():
    # Creating an interface to get the gmail details of the user
    global main_window
    gmail_frame = Toplevel(main_window)
    gmail_frame.title("GMAIL DETAILS")
    gmail_frame.geometry("500x350")
    label_email = Label(gmail_frame, text="Email", padx=10, pady=20, width=10,
font="Courier")
    label_email.grid(row=0, column=0)
    label_password = Label(gmail_frame, text="Password", padx=10, pady=20,
width=10, font="Courier")
    label_password.grid(row=1, column=0)
    entry_email = Entry(gmail_frame, width=30)
    entry_email.grid(row=0, column=1)
    entry_password = Entry(gmail_frame, width=30, show="*")
    entry_password.grid(row=1, column=1)

    # This function is called when the user wants to see the password while typing
    def show_password():
        entry_password = Entry(gmail_frame, width=30)
        entry_password.grid(row=1, column=1)

    # This function is called when the user wants to hide the password while typing
    def hide_password():
        entry_password = Entry(gmail_frame, width=30, show="*")
        entry_password.grid(row=1, column=1)

    button_visible = Button(gmail_frame, text="visible", command=show_password,
padx=3)
    button_visible.grid(row=1, column=2)
    button_visible = Button(gmail_frame, text="hide", command=hide_password, padx=3)
    button_visible.grid(row=1, column=3)

    # this function is called when the user wants to save the gmail details
    def submit():
        # in order to save the mail address and password we will be using shelves
        s = shelve.open("login_details.shelve")
        email = entry_email.get()
        password = entry_password.get()
        # if user tries to submit the empty entries the user will get the error dialog box
        # and if both the entries are filled we will save the entries in the shelves
        # so that we get access to them using shelves
        # and after saving the details successfully we will destroy the frame and take the
user back to home interface
        if email != "" and password != "":
            s["EMAIL"] = entry_email.get()
            s["PASSWORD"] = entry_password.get()
            message = messagebox.showinfo("SUCCESSFULL", "YOUR EMAIL AND
PASSWORD HAVE BEEN SUCCESSFULLY SAVED")
```

```
            Label(gmail_frame, text=message)
            gmail_frame.destroy()
            gmail_details()
        else:
            message = messagebox.showinfo("UNSUCCESSFULL", "PLEASE ENTER
BOTH YOUR EMAIL AND PASSWORD TO UPDATE THE "
                                               "DETAILS")
            Label(gmail_frame, text=message)
        s.close()

    button_submit = Button(gmail_frame, text="Submit", command=submit,
font="Courier", width=10)
    button_submit.grid(row=2, column=0, columnspan=2, padx=10, pady=10)
    button_exit_gmail_frame = Button(gmail_frame, text="Exit",
command=gmail_frame.destroy, width=10, font="Courier")
    button_exit_gmail_frame.grid(row=3, column=0, columnspan=2, padx=10, pady=10)


# this function is called when button_edit_database is pressed
def update_existing_policy():
    global main_window
    update_frame = Toplevel(main_window)
    update_frame.title("UPDATE THE DATABASE")
    update_frame.geometry("840x600")

    # we have assigned different radio buttons to different options and each radio button
has different value
    # we have used an IntVar variable which can extract the value of the radio button the
user has selected
    # so when the user select any one of them and  submits it
    # we will know that user selected option

    label_option = Label(update_frame, text="SELECT ANY ONE OPTIONS", padx=3,
pady=5, width=30, font="Courier")
    label_option.grid(row=0, column=0, columnspan=1, sticky="w")
    radio_button_controller = IntVar()
    radio_button_option_1 = Radiobutton(update_frame, text="EDIT THE ENTRIES OF A
SINGLE POLICY",
                                        variable=radio_button_controller, value=1, padx=3, pady=5,
font="Courier")
```

```
    radio_button_option_1.grid(row=1, column=0, columnspan=1, sticky="w")
    radio_button_option_2 = Radiobutton(update_frame, text="EDIT THE ENTRIES
COMMON TO ALL THE POLICIES",
                                        variable=radio_button_controller, value=2, padx=3, pady=5,
font="Courier")
    radio_button_option_2.grid(row=2, column=0, columnspan=1, sticky="w")

    # this function is called when the user select any on radio button and submits it
    def submit_function():
        # first we will try to know which radio button user has selected using INtVar
variable
        # and then move on creating the toplevel window depending on the user choice
        if radio_button_controller.get() == 1:
            # if user has choose to update the entries of a single policy we don't need the
whole table we will
            # extract the policy number from the user whose entries user needs to update
since policy number is
            # unique for all the entries

            update_frame = Toplevel(main_window)
            update_frame.title("EDIT THE ENTRIES OF A SINGLE POLICY")
            update_frame.geometry("840x600")
            label_policy_number = Label(update_frame, text="ENTER THE POLICY
NUMBER", padx=3, pady=5, width=30,
                                        font="Courier")
            label_policy_number.grid(row=0, column=0, columnspan=1, sticky="w")
            entry_policy_number = Entry(update_frame, width=30)
            entry_policy_number.grid(row=0, column=2, columnspan=3, sticky="e")

            # this function will extract the entry having the policy number user has submitted
            def view_function():
                # first we will extract the policy number from the entry widget and then search
the database
                policy_number = entry_policy_number.get()
                # we will need a temp cursor to point to the entry we need to update
                cursor = conn.cursor()
                print(cursor.execute(
                    "SELECT EMAIL , CONTACT_NO ,ADDRESS , NOMINEE ,
POLICY_STATUS  ,PREMIUM_AMOUNT , SUM_ASSURED , POLICY_TERM ,
PREMIUM_PAYING_TERM ,PAYMENT_MODE   FROM LIFEINSURANCE WHERE
POLICY_NUMBER = ?",
                    (policy_number,)).fetchone())
                entry = cursor.execute(
                    "SELECT EMAIL , CONTACT_NO ,ADDRESS , NOMINEE ,
POLICY_STATUS  ,PREMIUM_AMOUNT , SUM_ASSURED , POLICY_TERM ,
PREMIUM_PAYING_TERM ,PAYMENT_MODE   FROM LIFEINSURANCE WHERE
POLICY_NUMBER = ?",
                    (policy_number,)).fetchone()
                # if we don't find the data we will pop an error message showing user that the
database does not have
                # any entries with given policy number
                if entry == None:
                    message = messagebox.showinfo("UNSUCCESSFULL!!",
                                                 "THERE IS NO POLICY HAVING THE SPECIFIED
POLICY NUMBER ."
                                                 "PLEASE CHECK THE POLICY NUMBER AND TRY
AGAIN")
                    Label(update_frame, text=message)
                    update_frame.destroy()
                # we will retrieve the data that can be updated and sort them in a frame so
that user can know what
                # currently is in the database
                # and then user can change any entry user wants
                else:
                    detail_frame = Frame(update_frame)
                    detail_frame.config(height=600, padx=10)
                    detail_frame.grid(row=2, column=0, columnspan=4, sticky="nsew")

                    detail_frame.columnconfigure(0, weight=20)
                    detail_frame.columnconfigure(1, weight=20)
                    detail_frame.columnconfigure(2, weight=5)
                    detail_frame.columnconfigure(3, weight=5)

                    email_label = Label(detail_frame, text="EMAIL", padx=10, pady=5,
width=10, font="Courier")
                    email_label.grid(row=0, column=0, columnspan=1)
                    email_var = StringVar()
                    email_var.set(entry[0])
                    email_entry = Entry(detail_frame, textvariable=email_var, width=30)
                    email_entry.grid(row=0, column=2)
```

```
        contact_no_label = Label(detail_frame, text="CONTACT NO.", padx=10,
pady=5, width=10,
                        font="Courier")
        contact_no_label.grid(row=1, column=0, columnspan=1)
        contact_no_var = StringVar()
        contact_no_var.set(entry[1])
        contact_no_entry = Entry(detail_frame, textvariable=contact_no_var,
width=30)
        contact_no_entry.grid(row=1, column=2)

        address_label = Label(detail_frame, text="ADDRESS", padx=10, pady=5,
width=10, font="Courier")
        address_label.grid(row=2, column=0, columnspan=1)
        address_var = StringVar()
        address_var.set(entry[2])
        address_entry = Entry(detail_frame, textvariable=address_var, width=30)
        address_entry.grid(row=2, column=2)

        nominee_label = Label(detail_frame, text="NOMINEE", padx=10, pady=5,
width=10, font="Courier")
        nominee_label.grid(row=3, column=0, columnspan=1)
        nominee_var = StringVar()
        nominee_var.set(entry[3])
        nominee_entry = Entry(detail_frame, textvariable=nominee_var, width=30)
        nominee_entry.grid(row=3, column=2)

        status_label = Label(detail_frame, text="POLICY STATUS", padx=10,
pady=5, width=15, font="Courier")
        status_label.grid(row=4, column=0, columnspan=2)
        status_var = StringVar()
        status_var.set(entry[4])
        status_entry = Entry(detail_frame, textvariable=status_var, width=30)
        status_entry.grid(row=4, column=2)

        premium_amount_label = Label(detail_frame, text="PREMIUM AMOUNT",
padx=10, pady=5, width=15,
                         font="Courier")
        premium_amount_label.grid(row=5, column=0, columnspan=2)
        premium_amount_var = StringVar()
```

```
            premium_amount_var.set(entry[5])
            premium_amount_entry = Entry(detail_frame,
textvariable=premium_amount_var, width=30)
            premium_amount_entry.grid(row=5, column=2)

            sum_assured_label = Label(detail_frame, text="SUM ASSURED",
padx=10, pady=5, width=10,
                              font="Courier")
            sum_assured_label.grid(row=6, column=0, columnspan=2)
            sum_assured_var = StringVar()
            sum_assured_var.set(entry[6])
            sum_assured_entry = Entry(detail_frame, textvariable=sum_assured_var,
width=30)
            sum_assured_entry.grid(row=6, column=2)

            policy_term_label = Label(detail_frame, text="POLICY TERM", padx=10,
pady=5, width=15,
                              font="Courier")
            policy_term_label.grid(row=7, column=0, columnspan=2)
            policy_term_var = StringVar()
            policy_term_var.set(entry[7])
            policy_term_entry = Entry(detail_frame, textvariable=policy_term_var,
width=30)
            policy_term_entry.grid(row=7, column=2)

            premium_paying_term_label = Label(detail_frame, text="PREMIUM
PAYING TERM", padx=8, pady=5,
                                   width=20,
                                   font="Courier")
            premium_paying_term_label.grid(row=8, column=0, columnspan=2)
            premium_paying_term_var = StringVar()
            premium_paying_term_var.set(entry[8])
            premium_paying_term_entry = Entry(detail_frame,
textvariable=premium_paying_term_var, width=30)
            premium_paying_term_entry.grid(row=8, column=2)

            payment_mode_label = Label(detail_frame, text="PAYMENT MODE",
padx=10, pady=5, width=10,
                               font="Courier")
            payment_mode_label.grid(row=9, column=0, columnspan=2)
```

```
                    payment_mode_var = StringVar()
                    payment_mode_var.set(entry[9])
                    payment_mode_entry = Entry(detail_frame,
textvariable=payment_mode_var, width=30)
                    payment_mode_entry.grid(row=9, column=2)

                    # after updating the entries user will press the save button which we call
the below function
                    # which will commit the changes in the database
                    def save_the_changes():
                        #     print(email_var.get())
                        # in order to save the changes in the database we will use a cursor and
an update query
                        # which will update the entry in the database and then we will commit the
cursor to save the changes on the database
                        temp_cursor = conn.cursor()
                        try:
                            temp_cursor.execute(
                                "UPDATE LIFEINSURANCE SET EMAIL=? , CONTACT_NO=?
,ADDRESS=? , NOMINEE=? , POLICY_STATUS=?  ,PREMIUM_AMOUNT=? ,
SUM_ASSURED=? , POLICY_TERM=? , PREMIUM_PAYING_TERM=?
,PAYMENT_MODE=?  WHERE POLICY_NUMBER = ?",
                                (email_var.get(), contact_no_var.get(), address_var.get(),
nominee_var.get(),
                                 status_var.get(), premium_amount_var.get(),
sum_assured_var.get(),
                                 policy_term_var.get(),
                                 premium_paying_term_var.get(), payment_mode_var.get(),
policy_number))
                            temp_cursor.connection.commit()
                            conn.commit()
                        except:
                            # if any error occurs we will catchg the exception and will show an
error dialog box to let user know that changes weren't saved
                            message = messagebox.showinfo("UNSUCCESSFULL!!", "THE
CHANGES WERE NOT SAVED")
                            Label(update_frame, text=message)
                            detail_frame.destroy()
                        else:
                            # if no error occurs in the try block we will
                            # show a dialog box reffering that the changes were done successfully
                            message = messagebox.showinfo("SUCCESSFULL!!", "THE
CHANGES WERE SAVED")
                            Label(update_frame, text=message)
                            detail_frame.destroy()
                            entry_policy_number = Entry(update_frame, width=30)
                            entry_policy_number.grid(row=0, column=2, columnspan=3,
sticky="e")

                    save_button = Button(detail_frame, text="SAVE",
command=save_the_changes, width=5, font="Courier",
                                     height=1)
                    save_button.grid(row=10, column=2, columnspan=3, padx=10, pady=10)
                cursor.close()

            view_button = Button(update_frame, text="VIEW", command=view_function,
width=5, font="Courier", height=1)
            view_button.grid(row=1, column=2, columnspan=3, padx=10, pady=10)
            detail_frame = Frame(update_frame)
            detail_frame.config(height=600, padx=10)
            detail_frame.grid(row=2, column=0, columnspan=4, sticky="nsew")

        # after doing some research i found that there are some parameters in the
database which are common for same
        # attribute
        # i.e every policy has its own commission rates so rather than updating each
entries commission
        # rate we can update every entries commission rate whose policy name is same
        elif radio_button_controller.get() == 2:
            # hence we will ask for the policy name from the user
            # and retrieve the commission rate from the policy name and sort them in the
frame
            update_frame = Toplevel(main_window)
            update_frame.title("UPDATE THE DATABASE ")
            update_frame.geometry("840x600")
            label_policy_name = Label(update_frame, text="ENTER THE POLICY NAME",
padx=3, pady=5, width=30,
                                  font="Courier")
            label_policy_name.grid(row=0, column=0, columnspan=1, sticky="w")
            entry_policy_name = Entry(update_frame, width=30)
            entry_policy_name.grid(row=0, column=2, columnspan=3, sticky="e")

            # this function will retrieve the data from the database and sort them in the frame

            def view_function():
                policy_name = entry_policy_name.get()
                cursor = conn.cursor()
                # print(cursor.execute( "SELECT
ISSUE_DATE,FIRST_YEAR_COMMISSION_PERCENT ,
                # RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT
,RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT ,
                # RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT
,GST_ON_FIRST_YEAR_COMMISSION ,
                #
GST_ON_RENEWAL_COMMISSION,POLICY_NUMBER,PREMIUM_AMOUNT  FROM
LIFEINSURANCE WHERE POLICY_NAME = ?",
                # (policy_name,)).fetchall())

                # we will retrieve all the entries from the database whose policy name is
                # equal to the users given name this will be done using a cursor and a select
query
                entry = cursor.execute(
                    "SELECT ISSUE_DATE,FIRST_YEAR_COMMISSION_PERCENT
,RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT ,"
                    "RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT ,
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT ,"
                    "GST_ON_FIRST_YEAR_COMMISSION ,
GST_ON_RENEWAL_COMMISSION,POLICY_NUMBER,PREMIUM_AMOUNT
FROM "
                    "LIFEINSURANCE WHERE POLICY_NAME = ?",
                    (policy_name,)).fetchall()
                if entry == None:
                    # if there are no entries in the database of the policy name equal to the
user specified name
                    # we will let the user know that no entries for the given policy name exist
using a dialog box
                    message = messagebox.showinfo("UNSUCCESSFULL!!",
                                                  "THERE IS NO POLICY HAVING THE SPECIFIED
POLICY NAME ."
                                                  "PLEASE CHECK THE POLICY NAME AND TRY
AGAIN")
                    Label(update_frame, text=message)
                    update_frame.destroy()
                else:

                    detail_frame = Frame(update_frame)
                    detail_frame.config(height=600, padx=10)
                    detail_frame.grid(row=2, column=0, columnspan=4, sticky="nsew")

                    detail_frame.columnconfigure(0, weight=20)
                    detail_frame.columnconfigure(1, weight=20)
                    detail_frame.columnconfigure(2, weight=5)
                    detail_frame.columnconfigure(3, weight=5)

                    FIRST_YEAR_COMMISSION_PERCENT_label = Label(detail_frame,
text="FIRST_YEAR_COMMISSION_PERCENT",
                                        padx=10, pady=5, width=40, font="Courier")
                    FIRST_YEAR_COMMISSION_PERCENT_label.grid(row=0, column=0,
columnspan=1)
                    FIRST_YEAR_COMMISSION_PERCENT_var = IntVar()
                    FIRST_YEAR_COMMISSION_PERCENT_var.set(entry[0][1])
                    FIRST_YEAR_COMMISSION_PERCENT_entry = Entry(detail_frame,
textvariable=FIRST_YEAR_COMMISSION_PERCENT_var,
                                        width=10)
                    FIRST_YEAR_COMMISSION_PERCENT_entry.grid(row=0, column=2)

                    RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_label =
Label(detail_frame,
text="RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT",
                                              padx=10, pady=5, width=50,
                                              font="Courier")
RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_label.grid(row=1, column=0,
columnspan=1)
                    RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_var =
StringVar()
RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_var.set(entry[0][2])
                    RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_entry =
Entry(detail_frame,
textvariable=RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_var,
                                              width=10)
```

```
RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_entry.grid(row=1, column=2)

                RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_label =
Label(detail_frame,
text="RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT",
                                                            padx=10, pady=5, width=50,
font="Courier")
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_label.grid(row=2, column=0,
columnspan=1)
                RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_var = IntVar()
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_var.set(entry[0][3])
                RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_entry =
Entry(detail_frame,
textvariable=RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_var,
                                                            width=10)
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_entry.grid(row=2, column=2)

                RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_label
= Label(detail_frame,
text="RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT",
                                                                 padx=10, pady=5, width=50,
                                                                 font="Courier")
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_label.grid(row=3,
column=0, columnspan=1)
                RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_var =
IntVar()
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_var.set(entry[0][4])
                RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_entry
= Entry(detail_frame,
textvariable=RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_var,
                                                                 width=10)
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_entry.grid(row=3,
column=2)

                GST_ON_FIRST_YEAR_COMMISSION_label = Label(detail_frame,
text="GST_ON_FIRST_YEAR_COMMISSION",
                                                   padx=10, pady=5, width=50, font="Courier")
                GST_ON_FIRST_YEAR_COMMISSION_label.grid(row=4, column=0,
columnspan=2)
                GST_ON_FIRST_YEAR_COMMISSION_var = IntVar()
                GST_ON_FIRST_YEAR_COMMISSION_var.set(entry[0][5])
                GST_ON_FIRST_YEAR_COMMISSION_entry = Entry(detail_frame,
textvariable=GST_ON_FIRST_YEAR_COMMISSION_var, width=10)
                GST_ON_FIRST_YEAR_COMMISSION_entry.grid(row=4, column=2)

                GST_ON_RENEWAL_COMMISSION_label = Label(detail_frame,
text="GST_ON_RENEWAL_COMMISSION", padx=10,
                                        pady=5, width=50,
                                        font="Courier")
                GST_ON_RENEWAL_COMMISSION_label.grid(row=5, column=0,
columnspan=2)
                GST_ON_RENEWAL_COMMISSION_var = StringVar()
                GST_ON_RENEWAL_COMMISSION_var.set(entry[0][6])
                GST_ON_RENEWAL_COMMISSION_entry = Entry(detail_frame,
textvariable=GST_ON_RENEWAL_COMMISSION_var,
                                        width=10)
                GST_ON_RENEWAL_COMMISSION_entry.grid(row=5, column=2)
                # this function will be triggered when the user presses save_button
                def save_the_changes():
                    # in order to update the parameters for all the entries with the policy
name
                    # we will use cursor and  the update command ,
                    # and update all the parameters for all the entries with policy
                    # name equal to the name of the user specified name
                    cursor = conn.cursor()
                    try:
                        # since our cursor(entry) has now all the entries in the form of list
                        # we will loop through each entry and update the attributes
                        # if first year commision percent changes first year commission also
changes hence they both will updated together
                        # same goes with other two percentage attributes and there respective
commission

                        for selected_policy in entry:
                            if selected_policy[0] + datetime.timedelta(days=365) >=
datetime.datetime.now():
                                first_year_commission =
(FIRST_YEAR_COMMISSION_PERCENT_var.get() / 100) * (
                                        selected_policy[8] - (selected_policy[8] *
                                                              (
GST_ON_FIRST_YEAR_COMMISSION_var.get() / 100)))
                                sql = "UPDATE LIFEINSURANCE SET
FIRST_YEAR_COMMISSION_PERCENT=?,GST_ON_FIRST_YEAR_COMMISSION=
?,FIRST_YEAR_COMMISSION WHERE POLICY_NUMBER=?"
                                cursor.execute(sql, (
                                    FIRST_YEAR_COMMISSION_PERCENT_var.get(),
GST_ON_FIRST_YEAR_COMMISSION_var.get(),
                                    first_year_commission,
                                    selected_policy[7]))
                                cursor.connection.commit()
                                conn.commit()
                            if selected_policy[0] + datetime.timedelta(days=730) >=
datetime.datetime.now():
                                renewal_commission_for_2_3_year = (
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_var.get() / 100) * (
                                                                        selected_policy[8] - (selected_policy[8]
*
                                                                                                    (
GST_ON_RENEWAL_COMMISSION_var.get() / 100)))
                                sql = "UPDATE LIFEINSURANCE SET
RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT=?,GST_ON_RENEWAL_CO
MMISSION=?,RENEWAL_COMMISSION_FOR_2_3_YEAR=? WHERE
POLICY_NUMBER=?"
                                cursor.execute(sql,
(RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_var.get(),
                                                    GST_ON_RENEWAL_COMMISSION_var.get(),
                                                    renewal_commission_for_2_3_year,
selected_policy[7]))
                                cursor.connection.commit()
                                conn.commit()
                            if selected_policy[0] + datetime.timedelta(days=1460) >=
datetime.datetime.now():
                                renewal_commission_for_4_5_year = (
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_var.get() / 100) * (
                                                                        selected_policy[8] - (selected_policy[8]
*
                                                                                                    (
GST_ON_RENEWAL_COMMISSION_var.get() / 100)))
                                sql = "UPDATE LIFEINSURANCE SET
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT=?,GST_ON_RENEWAL_CO
```

```
MMISSION=?,RENEWAL_COMMISSION_FOR_4_5_YEAR=? WHERE
POLICY_NUMBER=?"
                              cursor.execute(sql,
(RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_var.get(),
                                               GST_ON_RENEWAL_COMMISSION_var.get(),
                                               renewal_commission_for_4_5_year,
selected_policy[7]))
                              cursor.connection.commit()
                              conn.commit()
                          if selected_policy[0] + datetime.timedelta(days=2190) >=
datetime.datetime.now():
                              renewal_commission_for_6_year_onwards = (
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_var.get() / 100) * (
                                                                      selected_policy[8] - (
                                                                      selected_policy[8] *
                                                                      (
GST_ON_RENEWAL_COMMISSION_var.get() / 100)))
                              sql = "UPDATE LIFEINSURANCE SET
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT=?,GST_ON_REN
EWAL_COMMISSION=?,RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS=?
WHERE POLICY_NUMBER=?"
                              cursor.execute(sql,
(RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_var.get(),
                                               GST_ON_RENEWAL_COMMISSION_var.get(),
                                               renewal_commission_for_6_year_onwards,
selected_policy[7]))
                              cursor.connection.commit()
                              conn.commit()

                              # since there are no derives attributes dependent on get
percentage they will be be update independently
                          if selected_policy[0] + datetime.timedelta(days=365) >=
datetime.datetime.now():
                              sql = "UPDATE LIFEINSURANCE SET
GST_ON_FIRST_YEAR_COMMISSION=? WHERE POLICY_NUMBER=?"
                              cursor.execute(sql,
(GST_ON_FIRST_YEAR_COMMISSION_var.get(), selected_policy[7]))
                              cursor.connection.commit()
                              conn.commit()
                          if selected_policy[0] + datetime.timedelta(days=730) >=
datetime.datetime.now():
                                sql = "UPDATE LIFEINSURANCE SET
GST_ON_RENEWAL_COMMISSION=? WHERE POLICY_NUMBER=?"
                                cursor.execute(sql,
(GST_ON_RENEWAL_COMMISSION_var.get(), selected_policy[7]))
                                cursor.connection.commit()
                                conn.commit()
                      except:
                          # if any error occurs we will catch the exception and will show an error
dialog box to let user know that changes weren't saved
                          message = messagebox.showinfo("UNSUCCESSFULL!!", "THE
CHANGES WERE NOT SAVED")
                          Label(update_frame, text=message)
                          detail_frame.destroy()
                      else:
                          # if no error occurs in the try block we will
                          # show a dialog box reffering that the changes were done successfully
                          message = messagebox.showinfo("SUCCESSFULL!!", "THE
CHANGES WERE SAVED")
                          Label(update_frame, text=message)
                          detail_frame.destroy()
                          label_policy_name = Label(update_frame, text="ENTER THE POLICY
NAME", padx=3, pady=5,
                                                    width=30,
                                                    font="Courier")
                          label_policy_name.grid(row=0, column=0, columnspan=1, sticky="w")
                          entry_policy_name = Entry(update_frame, width=30)
                          entry_policy_name.grid(row=0, column=2, columnspan=3, sticky="e")

                  save_button = Button(detail_frame, text="SAVE",
command=save_the_changes, width=5, font="Courier",
                                       height=1)
                  save_button.grid(row=10, column=2, columnspan=3, padx=10, pady=10)

          view_button = Button(update_frame, text="VIEW", command=view_function,
width=5, font="Courier", height=1)
          view_button.grid(row=1, column=2, columnspan=3, padx=10, pady=10)
          detail_frame = Frame(update_frame)
          detail_frame.config(height=600, padx=10)
          detail_frame.grid(row=2, column=0, columnspan=4, sticky="nsew")
```

```
    submit_button = Button(update_frame, text="SUBMIT", command=submit_function,
padx=3, pady=5, width=5,
                           font="Courier", height=1)
    submit_button.grid(row=3, column=0, columnspan=1, sticky="w")

# this function is called when the user presses on button_add_to_batabase
#  this function is used to add the entries to the database
def add_new_policy():
    global main_window
    new_frame = Toplevel(main_window)
    new_frame.title("ADD TO THE DATABASE")
    new_frame.geometry("700x400")
    label = Label(new_frame, text="SELECT THE FILE YOU WANT TO UPLOAD",
padx=5, pady=10, width=40, font="Courier")
    label.grid(row=0, column=0, columnspan=2)

    # this function is called when user presses on import
    # the function will open your file explorer and it expects you to selEct
    # aN EXCEL FILE THAT CONTAINS THE ENTRIES YOU WANT TO ADD TO THE
DATABASE
    # THERE A SAMPLE GIVEN IN THE FOLDER SAMPLE REFER IT WILL GUIDE
HOW TO USE THIS OPTION
    def filepath():
        path = filedialog.askopenfilename(initialdir="C:\\Users\\ROMIL\\Desktop",
title="Select an excel file",
                                          filetypes=(("Excel Workbook (*.xlsx)", "*.xlsx"), ("all files",
"*.*")))
        if path != " ":
            # A FUNCTION IN CONSOLE.PY IS USED WHICH HANDLES THIS
INSERTING OPERATION
            insert_to_database(path)
            message = messagebox.showinfo("SUCCESSFULL!!", "THE CHANGES WERE
SAVED")
            Label(new_frame, text=message)
            new_frame.destroy()
        else:
            message = messagebox.showinfo("UNSUCCESSFULL!!", "PLEASE SELECT A
VALID PATH")
            Label(new_frame, text=message)
            new_frame.destroy()

    button = Button(new_frame, text="import", command=filepath, pady=5)
    button.grid(row=0, column=2, sticky="w")

# THIS FUNCTION IS ACLLED WHEN THE USER WANTS TO EMPTY THE
DATABAS E
# THE  FUNCTION USUALLY TRUNCATES AL  THE ENTRIES PRESENT IN THE
DATABASE
def clear_the_database():
    try:
        cursor = conn.cursor()
        cursor.execute("DROP TABLE LIFEINSURANCE")
        cursor.execute(
            "CREATE TABLE IF NOT EXISTS LIFEINSURANCE (COMPANY TEXT,
CUSTOMER_NAME TEXT, EMAIL TEXT, CONTACT_NO TEXT, "
            "ADDRESS TEXT, NOMINEE TEXT, "
            "POLICY_STATUS TEXT, POLICY_NAME TEXT, POLICY_NUMBER TEXT,
ISSUE_DATE TEXT,MATURITY_DATE TEXT,PREMIUM_AMOUNT "
            "INTEGER, SUM_ASSURED INTEGER, "
            "POLICY_TERM INTEGER, PREMIUM_PAYING_TERM
INTEGER,PAYMENT_MODE INTEGER, FIRST_YEAR_COMMISSION_PERCENT
INTEGER, "
            "RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT
INTEGER,RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT INTEGER, "
            "RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT
INTEGER,GST_ON_FIRST_YEAR_COMMISSION INTEGER, "
            "GST_ON_RENEWAL_COMMISSION INTEGER, "
            "RENEWAL_DATE TEXT,"
            "FIRST_YEAR_COMMISSION INTEGER,"
            "RENEWAL_COMMISSION_FOR_2_3_YEAR INTEGER,"
            "RENEWAL_COMMISSION_FOR_4_5_YEAR INTEGER,"
            "RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS INTEGER,"
            "NOTE TEXT)")
        cursor.connection.commit()
        conn.commit()
        cursor.close()
    except:
        message = messagebox.showinfo("UNSUCCESSFULL!!", "SOMETHING WENT
WRONG")
```

```
        Label(main_window, text=message)
    else:
        message = messagebox.showinfo("SUCCESSFULL!!", "DATABASE WAS
CLEARED")
        Label(main_window, text=message)


def search_the_database():
    global main_window
    search_frame = Toplevel(main_window)
    search_frame.title("SEARCH THE DATABASE")
    search_frame.geometry("840x600")

    label_option = Label(search_frame, text="FILTERS", padx=3, pady=5, width=30,
font="Courier")
    label_option.grid(row=0, column=0, columnspan=1, sticky="w")
    radio_button_controller = IntVar()
    radio_button_controller.set(1)
    radio_button_option_1 = Radiobutton(search_frame, text="POLICY NUMBER",
                                        variable=radio_button_controller, value=1, padx=3, pady=5,
font="Courier")
    radio_button_option_1.grid(row=1, column=0, columnspan=1, sticky="w")
    radio_button_option_2 = Radiobutton(search_frame, text="POLICY NAME",
                                        variable=radio_button_controller, value=2, padx=3, pady=5,
font="Courier")
    radio_button_option_2.grid(row=2, column=0, columnspan=1, sticky="w")
    radio_button_option_3 = Radiobutton(search_frame, text="ISSUE DATE",
                                        variable=radio_button_controller, value=3, padx=3, pady=5,
font="Courier")
    radio_button_option_3.grid(row=3, column=0, columnspan=1, sticky="w")
    radio_button_option_4 = Radiobutton(search_frame, text="PAYMENT MODE",
                                        variable=radio_button_controller, value=4, padx=3, pady=5,
font="Courier")
    radio_button_option_4.grid(row=4, column=0, columnspan=1, sticky="w")
    radio_button_option_5 = Radiobutton(search_frame, text="POLICY STATUS",
                                        variable=radio_button_controller, value=5, padx=3, pady=5,
font="Courier")
    radio_button_option_5.grid(row=5, column=0, columnspan=1, sticky="w")

    def submit_the_radio_button_value():
```

```
    if radio_button_controller.get() == 1:
        # SINCE POLICY NUMBER IS UNIQUE FOR ALL THE ENTRIES SO RATHER
THAN CREATING AN EXCEL
        # WE WILL SHOW ALL THE ATTRIBUTES ON THE CURRENT WINDOW
ONLY
        # SO WE WILL TAKE THE POILICY NUMBER INPUT FROM THE USER
        # AND WILL SERACH FOR ENTRY HAVING POLICY NUMBER EQUAL TO
THE SPECIFIED ONE USING SELECT QUERY
        # ONCE WE FIND IT WE WILL SHOW ALL THE ATTRIBUTES ON THE
SCREEN
        search_frame = Toplevel()
        search_frame.title("SEARCHING THE ENTRIES OF A SINGLE POLICY")
        search_frame.geometry("1200x900")
        title_frame = Frame(search_frame)
        title_frame.grid(row=0, column=0, sticky="nsew")
        label_policy_number = Label(title_frame, text="ENTER THE POLICY
NUMBER", padx=3, pady=5, width=30,
                                    font="Courier")
        label_policy_number.grid(row=0, column=0, columnspan=1, sticky="w")
        entry_policy_number = Entry(title_frame, width=30)
        entry_policy_number.grid(row=0, column=2, sticky="e")

        # THIS FUNCTION RETRIEVES THE DATA FROM THE DATABASE AND
DISPLAYS IT ON THE SCRREN
        def view_function():
            policy_number = entry_policy_number.get()
            cursor = conn.cursor()
            print(cursor.execute(
                "SELECT * FROM LIFEINSURANCE WHERE POLICY_NUMBER = ?",
                (policy_number,)).fetchone())
            entry = cursor.execute(
                "SELECT * FROM LIFEINSURANCE WHERE POLICY_NUMBER = ?",
                (policy_number,)).fetchone()
            if entry == None:
                message = messagebox.showinfo("UNSUCCESSFULL!!",
                                              "THERE IS NO POLICY HAVING THE SPECIFIED
POLICY NUMBER ."
                                              "PLEASE CHECK THE POLICY NUMBER AND TRY
AGAIN")
```

```
                Label(search_frame, text=message)
                search_frame.destroy()
            else:
                frame = Frame(search_frame)
                frame.grid(row=2, column=0, sticky="nsew")
                canvas = Canvas(frame)
                canvas.config(height=600, width=1200)
                canvas.grid(row=0, column=0, sticky="nsew")
                detail_frame = Frame(canvas)
                detail_frame.config(height=600, padx=10)
                detail_frame.grid(row=0, column=0, columnspan=4, sticky="nsew")

                detail_frame.columnconfigure(0, weight=20)
                detail_frame.columnconfigure(1, weight=20)
                detail_frame.columnconfigure(2, weight=5)
                detail_frame.columnconfigure(3, weight=5)

                scrollbar = Scrollbar(frame, orient="vertical", command=canvas.yview)
                scrollbar.grid(row=0, column=0, sticky="nse")

                detail_frame.bind(
                    "<Configure>",
                    lambda e: canvas.configure(
                        scrollregion=canvas.bbox("all")
                    )
                )

                canvas.create_window((0, 0), window=detail_frame, anchor="nw")
                canvas.configure(yscrollcommand=scrollbar.set)

                company_label = Label(detail_frame, text="COMPANY", padx=10, pady=5,
width=10, font="Courier")
                company_label.grid(row=0, column=0, columnspan=1)
                company_var = StringVar()
                company_var.set(entry[0])
                company_entry = Label(detail_frame, textvariable=company_var, width=30,
font="Courier")
                company_entry.grid(row=0, column=2)

                CUSTOMER_NAME_label = Label(detail_frame, text="CUSTOMER
NAME", padx=10, pady=5, width=30,
                                  font="Courier")
                CUSTOMER_NAME_label.grid(row=1, column=0, columnspan=1)
                CUSTOMER_NAME_var = StringVar()
                CUSTOMER_NAME_var.set(entry[1])
                CUSTOMER_NAME_entry = Label(detail_frame,
textvariable=CUSTOMER_NAME_var, width=30, font="Courier")
                CUSTOMER_NAME_entry.grid(row=1, column=2)

                email_label = Label(detail_frame, text="EMAIL", padx=10, pady=5,
width=10, font="Courier")
                email_label.grid(row=2, column=0, columnspan=1)
                email_var = StringVar()
                email_var.set(entry[2])
                email_entry = Label(detail_frame, textvariable=email_var, width=30,
font="Courier")
                email_entry.grid(row=2, column=2)

                contact_no_label = Label(detail_frame, text="CONTACT NO.", padx=10,
pady=5, width=10,
                                font="Courier")
                contact_no_label.grid(row=3, column=0, columnspan=1)
                contact_no_var = StringVar()
                contact_no_var.set(entry[3])
                contact_no_entry = Label(detail_frame, textvariable=contact_no_var,
width=30, font="Courier")
                contact_no_entry.grid(row=3, column=2)

                address_label = Label(detail_frame, text="ADDRESS", padx=10, pady=5,
width=10, font="Courier")
                address_label.grid(row=4, column=0, columnspan=1)
                address_var = StringVar()
                address_var.set(entry[4])
                address_entry = Label(detail_frame, textvariable=address_var, width=30,
font="Courier")
                address_entry.grid(row=4, column=2)

                nominee_label = Label(detail_frame, text="NOMINEE", padx=10, pady=5,
width=10, font="Courier")
                nominee_label.grid(row=5, column=0, columnspan=1)
```

```
                nominee_var = StringVar()
                nominee_var.set(entry[5])
                nominee_entry = Label(detail_frame, textvariable=nominee_var, width=15,
font="Courier")
                nominee_entry.grid(row=5, column=2)

                status_label = Label(detail_frame, text="POLICY STATUS", padx=10,
pady=5, width=15, font="Courier")
                status_label.grid(row=6, column=0, columnspan=2)
                status_var = StringVar()
                status_var.set(entry[6])
                status_entry = Label(detail_frame, textvariable=status_var, width=15,
font="Courier")
                status_entry.grid(row=6, column=2)

                policy_name_label = Label(detail_frame, text="POLICY NAME", padx=10,
pady=5, width=15,
                                     font="Courier")
                policy_name_label.grid(row=7, column=0, columnspan=2)
                policy_name_var = StringVar()
                policy_name_var.set(entry[7])
                policy_name_entry = Label(detail_frame, textvariable=policy_name_var,
width=15, font="Courier")
                policy_name_entry.grid(row=7, column=2)

                policy_number_label = Label(detail_frame, text="POLICY NUMBER",
padx=10, pady=5, width=15,
                                       font="Courier")
                policy_number_label.grid(row=8, column=0, columnspan=2)
                policy_number_var = StringVar()
                policy_number_var.set(entry[8])
                policy_number_entry = Label(detail_frame,
textvariable=policy_number_var, width=10, font="Courier")
                policy_number_entry.grid(row=8, column=2)

                issue_date_label = Label(detail_frame, text="ISSUE DATE", padx=10,
pady=5, width=15, font="Courier")
                issue_date_label.grid(row=9, column=0, columnspan=2)
                issue_date_var = StringVar()
                issue_date_var.set(entry[9])
                issue_date_entry = Label(detail_frame, textvariable=issue_date_var,
width=40, font="Courier")
                issue_date_entry.grid(row=9, column=2)

                maturity_date_label = Label(detail_frame, text="MATURITY DATE",
padx=10, pady=5, width=15,
                                        font="Courier")
                maturity_date_label.grid(row=10, column=0, columnspan=2)
                maturity_date_var = StringVar()
                maturity_date_var.set(entry[10].split(" ")[0])
                maturity_date_entry = Label(detail_frame, textvariable=maturity_date_var,
width=40, font="Courier")
                maturity_date_entry.grid(row=10, column=2)

                premium_amount_label = Label(detail_frame, text="PREMIUM AMOUNT",
padx=10, pady=5, width=15,
                                         font="Courier")
                premium_amount_label.grid(row=11, column=0, columnspan=2)
                premium_amount_var = StringVar()
                premium_amount_var.set(entry[11])
                premium_amount_entry = Label(detail_frame,
textvariable=premium_amount_var, width=10,
                                         font="Courier")
                premium_amount_entry.grid(row=11, column=2)

                sum_assured_label = Label(detail_frame, text="SUM ASSURED",
padx=10, pady=5, width=10,
                                      font="Courier")
                sum_assured_label.grid(row=12, column=0, columnspan=2)
                sum_assured_var = StringVar()
                sum_assured_var.set(entry[12])
                sum_assured_entry = Label(detail_frame, textvariable=sum_assured_var,
width=10, font="Courier")
                sum_assured_entry.grid(row=12, column=2)

                policy_term_label = Label(detail_frame, text="POLICY TERM", padx=10,
pady=5, width=15,
                                      font="Courier")
                policy_term_label.grid(row=13, column=0, columnspan=2)
                policy_term_var = StringVar()
                policy_term_var.set(entry[13])
                policy_term_entry = Label(detail_frame, textvariable=policy_term_var,
width=10, font="Courier")
                policy_term_entry.grid(row=13, column=2)

                premium_paying_term_label = Label(detail_frame, text="PREMIUM
PAYING TERM", padx=8, pady=5,
                                          width=20,
                                          font="Courier")
                premium_paying_term_label.grid(row=14, column=0, columnspan=2)
                premium_paying_term_var = StringVar()
                premium_paying_term_var.set(entry[14])
                premium_paying_term_entry = Label(detail_frame,
textvariable=premium_paying_term_var, width=10,
                                          font="Courier")
                premium_paying_term_entry.grid(row=14, column=2)

                payment_mode_label = Label(detail_frame, text="PAYMENT MODE",
padx=10, pady=5, width=10,
                                     font="Courier")
                payment_mode_label.grid(row=15, column=0, columnspan=2)
                payment_mode_var = StringVar()
                payment_mode_var.set(entry[15])
                payment_mode_entry = Label(detail_frame,
textvariable=payment_mode_var, width=10, font="Courier")
                payment_mode_entry.grid(row=15, column=2)

                FIRST_YEAR_COMMISSION_PERCENT_label = Label(detail_frame,
text="FIRST_YEAR_COMMISSION_PERCENT",
                                                  padx=10, pady=5, width=40, font="Courier")
                FIRST_YEAR_COMMISSION_PERCENT_label.grid(row=16, column=0,
columnspan=1)
                FIRST_YEAR_COMMISSION_PERCENT_var = IntVar()
                FIRST_YEAR_COMMISSION_PERCENT_var.set(entry[16])
                FIRST_YEAR_COMMISSION_PERCENT_entry = Label(detail_frame,
textvariable=FIRST_YEAR_COMMISSION_PERCENT_var,
                                                  width=10, font="Courier")
                FIRST_YEAR_COMMISSION_PERCENT_entry.grid(row=16, column=2)

                RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_label =
Label(detail_frame,
text="RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT",
                                                  padx=10, pady=5, width=50,
                                                  font="Courier")
RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_label.grid(row=17,
column=0, columnspan=1)
             RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_var = IntVar()
RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_var.set(entry[17])
             RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_entry =
Label(detail_frame,
textvariable=RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_var,
                                                  width=10, font="Courier")
RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT_entry.grid(row=17,
column=2)

             RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_label =
Label(detail_frame,
text="RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT",
                                                  padx=10, pady=5, width=50,
font="Courier")
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_label.grid(row=18,
column=0, columnspan=1)
             RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_var = IntVar()
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_var.set(entry[18])
             RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_entry =
Label(detail_frame,
textvariable=RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_var,
                                                  width=10, font="Courier")
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT_entry.grid(row=18,
column=2)

             RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_label
= Label(detail_frame,
text="RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT",
                                                      padx=10, pady=5, width=50,
                                                      font="Courier")
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_label.grid(row=19,
column=0, columnspan=1)
             RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_var =
IntVar()
```

```
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_var.set(entry[19])
            RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_entry
= Label(detail_frame,
textvariable=RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_var,
                                                 width=10, font="Courier")
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT_entry.grid(row=19,
column=2)

            GST_ON_FIRST_YEAR_COMMISSION_label = Label(detail_frame,
text="GST_ON_FIRST_YEAR_COMMISSION",
                                      padx=10, pady=5, width=50, font="Courier")
            GST_ON_FIRST_YEAR_COMMISSION_label.grid(row=20, column=0,
columnspan=2)
            GST_ON_FIRST_YEAR_COMMISSION_var = IntVar()
            GST_ON_FIRST_YEAR_COMMISSION_var.set(entry[20])
            GST_ON_FIRST_YEAR_COMMISSION_entry = Label(detail_frame,
textvariable=GST_ON_FIRST_YEAR_COMMISSION_var, width=10,
                                      font="Courier")
            GST_ON_FIRST_YEAR_COMMISSION_entry.grid(row=20, column=2)

            GST_ON_RENEWAL_COMMISSION_label = Label(detail_frame,
text="GST_ON_RENEWAL_COMMISSION", padx=10,
                                    pady=5, width=50,
                                    font="Courier")
            GST_ON_RENEWAL_COMMISSION_label.grid(row=21, column=0,
columnspan=2)
            GST_ON_RENEWAL_COMMISSION_var = IntVar()
            GST_ON_RENEWAL_COMMISSION_var.set(entry[21])
            GST_ON_RENEWAL_COMMISSION_entry = Label(detail_frame,
textvariable=GST_ON_RENEWAL_COMMISSION_var,
                                    width=10, font="Courier")
            GST_ON_RENEWAL_COMMISSION_entry.grid(row=21, column=2)

            renewal_date_label = Label(detail_frame, text="RENEWAL DATE",
padx=10, pady=5, width=15,
                            font="Courier")
            renewal_date_label.grid(row=22, column=0, columnspan=2)
            renewal_date_var = StringVar()
            renewal_date_var.set(entry[22])
            renewal_date_entry = Label(detail_frame, textvariable=renewal_date_var,
width=40, font="Courier")
            renewal_date_entry.grid(row=22, column=2)

            FIRST_YEAR_COMMISSION_label = Label(detail_frame,
text="FIRST_YEAR_COMMISSION",
                                  padx=10, pady=5, width=40, font="Courier")
            FIRST_YEAR_COMMISSION_label.grid(row=23, column=0,
columnspan=1)
            FIRST_YEAR_COMMISSION_var = IntVar()
            FIRST_YEAR_COMMISSION_var.set(entry[23])
            FIRST_YEAR_COMMISSION_entry = Label(detail_frame,
                                  textvariable=FIRST_YEAR_COMMISSION_var,
                                  width=10, font="Courier")
            FIRST_YEAR_COMMISSION_entry.grid(row=23, column=2)

            RENEWAL_COMMISSION_FOR_2_3_YEAR_label = Label(detail_frame,
text="RENEWAL_COMMISSION_FOR_2_3_YEAR",
                                        padx=10, pady=5, width=50,
                                        font="Courier")
            RENEWAL_COMMISSION_FOR_2_3_YEAR_label.grid(row=24,
column=0, columnspan=1)
            RENEWAL_COMMISSION_FOR_2_3_YEAR_var = IntVar()
            RENEWAL_COMMISSION_FOR_2_3_YEAR_var.set(entry[24])
            RENEWAL_COMMISSION_FOR_2_3_YEAR_entry = Label(detail_frame,
textvariable=RENEWAL_COMMISSION_FOR_2_3_YEAR_var,
                                        width=10, font="Courier")
            RENEWAL_COMMISSION_FOR_2_3_YEAR_entry.grid(row=24,
column=2)

            RENEWAL_COMMISSION_FOR_4_5_YEAR_label = Label(detail_frame,
text="RENEWAL_COMMISSION_FOR_4_5_YEAR",
                                        padx=10, pady=5, width=50, font="Courier")
            RENEWAL_COMMISSION_FOR_4_5_YEAR_label.grid(row=25,
column=0, columnspan=1)
            RENEWAL_COMMISSION_FOR_4_5_YEAR_var = IntVar()
            RENEWAL_COMMISSION_FOR_4_5_YEAR_var.set(entry[25])
            RENEWAL_COMMISSION_FOR_4_5_YEAR_entry = Label(detail_frame,
textvariable=RENEWAL_COMMISSION_FOR_4_5_YEAR_var,
                                        width=10, font="Courier")
            RENEWAL_COMMISSION_FOR_4_5_YEAR_entry.grid(row=25,
column=2)

                RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_label =
Label(detail_frame,
text="RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS",
                                                    padx=10, pady=5, width=50,
                                                    font="Courier")
                RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_label.grid(row=26,
column=0, columnspan=1)
                RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_var = IntVar()
                RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_var.set(entry[26])
                RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_entry =
Label(detail_frame,
textvariable=RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_var,
                                                    width=10, font="Courier")
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_entry.grid(row=26, column=2)

                note_label = Label(detail_frame, text="NOTE", padx=10, pady=5, width=15,
font="Courier")
                note_label.grid(row=27, column=0, columnspan=2)
                note_var = StringVar()
                note_var.set(entry[27])
                note_entry = Label(detail_frame, textvariable=note_var, width=50,
font="Courier")
                note_entry.grid(row=27, column=2)

            view_button = Button(title_frame, text="VIEW", command=view_function,
width=5, font="Courier", height=1)
            view_button.grid(row=1, column=2, columnspan=3, padx=10, pady=10)



        elif radio_button_controller.get() == 2:
            # SINCE THERE CAN BE MULKTIPLE ENTRIES FOR THE USER SPECIFIED
POLICY NAME
            # WE WILL CREATE AN EXCEL FOR ALL THE ENTRIES HAVING THE
POLICY NAME EQUAL T0O THE USER SPECIFIED ONE
            # AND THEN LET USER DECIDE WHERE TO SAVE THE EXCEL SHEET
PREPARED BY US CONTANING ALL THE REQUIRED ENTRIES
            search_frame = Toplevel()
            search_frame.title("POLICY NAME")
            search_frame.geometry("700x300")
            temp_frame = Frame(search_frame)
            temp_frame.grid(row=0, column=0, sticky="nsew")
            label_policy_name = Label(temp_frame, text="ENTER THE POLICY NAME",
padx=3, pady=5, width=30,
                                 font="Courier")
            label_policy_name.grid(row=0, column=0, columnspan=1, sticky="w")
            entry_policy_name = Entry(temp_frame, width=30)
            entry_policy_name.grid(row=0, column=2, sticky="e")
            label_policy_name = Label(temp_frame, text="ENTER THE POLICY NAME",
padx=3, pady=5, width=30,
                                 font="Courier")
            label_policy_name.grid(row=0, column=0, columnspan=1, sticky="w")
            entry_policy_name = Entry(temp_frame, width=30)
            entry_policy_name.grid(row=0, column=2, sticky="e")

            # THIS FUNCTION FINDS ALL THE ENTRIES AND CREATE AN EXCEL
SHEET AND
            # THEN OPEN FILE EXPLORER TO LET USER DECIDE WHERE IT SHOULD
BE SAVED
            def create_function():
                entry = None
                policy_name = entry_policy_name.get()
                cursor = conn.cursor()
                # print(cursor.execute(
                #     "SELECT * FROM LIFEINSURANCE WHERE POLICY_NAME = ?",
                #     (policy_name,)).fetchall())
                entry = cursor.execute(
                "SELECT * FROM LIFEINSURANCE WHERE POLICY_NAME = ?",
                (policy_name,)).fetchall()
                count = 0
                for i in entry:
                j = list(i)
                entry.remove(i)
                entry.insert(count, j)
                count += 1
                print(entry)
                if entry == None:
                message = messagebox.showinfo("UNSUCCESSFULL!!",
                                              "THERE IS NO POLICY HAVING THE SPECIFIED
POLICY NAME ."
                                        "PLEASE CHECK THE POLICY NAME AND TRY
AGAIN")
                Label(search_frame, text=message)
                search_frame.destroy()
            else:
                try:
                    filename = filedialog.asksaveasfilename(
                        title="Select an excel file", defaultextension=".xlsx",
                        filetypes=(("Excel Workbook (*.xlsx)", "*.xlsx"),
                                ("all files", "*.*")))
                    print(filename)

                    dateframe = pandas.DataFrame(numpy.array(entry),
                                            columns=["COMPANY", "CUSTOMER_NAME",
"EMAIL", "CONTACT_NO",
                                                "ADDRESS",
                                                "NOMINEE",
                                                "POLICY_STATUS", " POLICY_NAME ",
"POLICY_NUMBER",
                                                "ISSUE_DATE ", "MATURITY_DATE",
"PREMIUM_AMOUNT ",
                                                " SUM_ASSURED  ",
                                                "POLICY_TERM", "
PREMIUM_PAYING_TERM", "PAYMENT_MODE ",
                                                "FIRST_YEAR_COMMISSION_PERCENT",
"RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT",
"RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT",
"RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT",
                                                "GST_ON_FIRST_YEAR_COMMISSION",
                                                "GST_ON_RENEWAL_COMMISSION",
                                                "RENEWAL_DATE",
                                                "FIRST_YEAR_COMMISSION ",
                                                "RENEWAL_COMMISSION_FOR_2_3_YEAR ",
                                                "RENEWAL_COMMISSION_FOR_4_5_YEAR ",
"RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS",
                                                "NOTE "])
                    dateframe.to_excel(str(filename))
                except ValueError as e:
                    message = messagebox.showinfo("UNSUCCESSFULL!!",
                                                  "PLEASE MENTION THE FILENAME")
                    Label(search_frame, text=message)
                    search_frame.destroy()

                except:
                    message = messagebox.showinfo("UNSUCCESSFULL!!",
                                                  "THERE IS NO POLICY HAVING THE SPECIFIED
POLICY NAME ."
                                                  "PLEASE CHECK THE POLICY NAME AND TRY
AGAIN")
                    Label(search_frame, text=message)
                    search_frame.destroy()
                else:
                    message = messagebox.showinfo("SUCCESSFULL!!",
                                                  "AN EXCEL SHEET HAS BEEN SAVED AT
{}".format(str(filename)))
                    Label(search_frame, text=message)
                    search_frame.destroy()

            view_button = Button(temp_frame, text="SAVE", command=create_function,
width=5, font="Courier", height=1)
            view_button.grid(row=6, column=2, columnspan=3, padx=10, pady=10)

        # SINCE THERE CAN BE MULTIPLE ENTRIES FOR THE USER SPECIFIED
PAYMENT MODE HENCE
        # WE WILL CREATE AN EXCEL FOR ALL THE ENTRIES HAVING THE
PAYMENT MODE EQUAL T0O THE USER SPECIFIED ONE
        # AND THEN LET USER DECIDE WHERE TO SAVE THE EXCEL SHEET
PREPARED BY US CONTAINING ALL THE REQUIRED ENTRIES
        elif radio_button_controller.get() == 4:

            search_frame = Toplevel()
            search_frame.title("PAYMENT MODE")
            search_frame.geometry("700x300")
            temp_frame = Frame(search_frame)
            temp_frame.grid(row=0, column=0, sticky="nsew")
            label_policy_mode = Label(temp_frame, text="ENTER THE PAYMENT MODE",
padx=3, pady=5, width=30,
                                font="Courier")
            label_policy_mode.grid(row=0, column=0, columnspan=1, sticky="w")
            entry_policy_mode = Entry(temp_frame, width=30)
            entry_policy_mode.grid(row=0, column=2, sticky="e")
            label_policy_mode = Label(temp_frame, text="ENTER THE PAYMENT MODE",
padx=3, pady=5, width=30,
                                font="Courier")
            label_policy_mode.grid(row=0, column=0, columnspan=1, sticky="w")
            entry_policy_mode = Entry(temp_frame, width=30)
            entry_policy_mode.grid(row=0, column=2, sticky="e")

            # THIS FUNCTION FINDS ALL THE ENTRIES AND CREATE AN EXCEL
SHEET AND
            # THEN OPEN FILE EXPLORER TO LET USER DECIDE WHERE IT SHOULD
BE SAVED
            def create_function():
               entry = None
               policy_mode = entry_policy_mode.get()
               cursor = conn.cursor()
               print(cursor.execute(
                "SELECT * FROM LIFEINSURANCE WHERE PAYMENT_MODE = ?",
                (policy_mode,)).fetchall())
               entry = cursor.execute(
                "SELECT * FROM LIFEINSURANCE WHERE PAYMENT_MODE = ?",
                (policy_mode,)).fetchall()
               count = 0
               for i in entry:
                j = list(i)
                entry.remove(i)
                entry.insert(count, j)
                count += 1
               print(entry)
               if entry == None:
                message = messagebox.showinfo("UNSUCCESSFULL!!",
                                          "THERE IS NO POLICY HAVING THE SPECIFIED
PAYMENT MODE ."
                                          "PLEASE CHECK THE POLICY NAME AND TRY
AGAIN")
                Label(search_frame, text=message)
                search_frame.destroy()
            else:
                try:
                    filename = filedialog.asksaveasfilename(
                        title="Select an excel file", defaultextension=".xlsx",
                        filetypes=(("Excel Workbook (*.xlsx)", "*.xlsx"),
                                   ("all files", "*.*")))
                    print(filename)

                    dateframe = pandas.DataFrame(numpy.array(entry),
                                                 columns=["COMPANY", "CUSTOMER_NAME",
"EMAIL", "CONTACT_NO",
                                                          "ADDRESS",
                                                          "NOMINEE",
                                                          "POLICY_STATUS", " POLICY_NAME ",
"POLICY_NUMBER",
                                                          "ISSUE_DATE ", "MATURITY_DATE",
"PREMIUM_AMOUNT ",
                                                          " SUM_ASSURED  ",
                                                          "POLICY_TERM", "
PREMIUM_PAYING_TERM", "PAYMENT_MODE ",
                                                          "FIRST_YEAR_COMMISSION_PERCENT",
"RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT",
"RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT",
"RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT",
                                                          "GST_ON_FIRST_YEAR_COMMISSION",
                                                          "GST_ON_RENEWAL_COMMISSION",
                                                          "RENEWAL_DATE",
                                                          "FIRST_YEAR_COMMISSION ",
                                                          "RENEWAL_COMMISSION_FOR_2_3_YEAR ",
                                                          "RENEWAL_COMMISSION_FOR_4_5_YEAR ",
"RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS",
                                                          "NOTE "])
                    dateframe.to_excel(str(filename))



                except ValueError as e:
                    message = messagebox.showinfo("UNSUCCESSFULL!!",
                                                  "PLEASE MENTION THE FILENAME")
                    Label(search_frame, text=message)
                    search_frame.destroy()

                except:
                    message = messagebox.showinfo("UNSUCCESSFULL!!",
                                                  "THERE IS NO POLICY HAVING THE SPECIFIED
POLICY NAME ."
                                                  "PLEASE CHECK THE POLICY NAME AND TRY
AGAIN")
                    Label(search_frame, text=message)
                    search_frame.destroy()
                else:
                    message = messagebox.showinfo("SUCCESSFULL!!",
                                                  "AN EXCEL SHEET HAS BEEN SAVED AT
{}".format(str(filename)))
                    Label(search_frame, text=message)
                    search_frame.destroy()

             view_button = Button(temp_frame, text="SAVE", command=create_function,
width=5, font="Courier", height=1)
             view_button.grid(row=6, column=2, columnspan=3, padx=10, pady=10)

          # SINCE THERE CAN BE MULTIPLE ENTRIES FOR THE USER SPECIFIED
PAYMENT MODE HENCE
          # WE WILL CREATE AN EXCEL FOR ALL THE ENTRIES HAVING THE
PAYMENT MODE EQUAL T0O THE USER SPECIFIED ONE
          # AND THEN LET USER DECIDE WHERE TO SAVE THE EXCEL SHEET
PREPARED BY US CONTAINING ALL THE REQUIRED ENTRIES
          elif radio_button_controller.get() == 5:
             search_frame = Toplevel()
             search_frame.title("POLICY STATUS")
             search_frame.geometry("700x300")
             temp_frame = Frame(search_frame)
             temp_frame.grid(row=0, column=0, sticky="nsew")
             label_policy_status = Label(temp_frame, text="ENTER THE POLICY STATUS",
padx=3, pady=5, width=30,
                                     font="Courier")
             label_policy_status.grid(row=0, column=0, columnspan=1, sticky="w")
             entry_policy_status = Entry(temp_frame, width=30)
             entry_policy_status.grid(row=0, column=2, sticky="e")
```

```
          label_policy_status = Label(temp_frame, text="ENTER THE POLICY STATUS",
padx=3, pady=5, width=30,
                                    font="Courier")
          label_policy_status.grid(row=0, column=0, columnspan=1, sticky="w")
          entry_policy_status = Entry(temp_frame, width=30)
          entry_policy_status.grid(row=0, column=2, sticky="e")

          # THIS FUNCTION FINDS ALL THE ENTRIES AND CREATE AN EXCEL
SHEET AND
          # THEN OPEN FILE EXPLORER TO LET USER DECIDE WHERE IT SHOULD
BE SAVED
          def create_function():
              entry = None
              policy_status = entry_policy_status.get()
              cursor = conn.cursor()
              # print(cursor.execute(
              #     "SELECT * FROM LIFEINSURANCE WHERE POLICY_STATUS = ?",
              #     (policy_status,)).fetchall())p
              entry = cursor.execute(
                  "SELECT * FROM LIFEINSURANCE WHERE POLICY_STATUS = ?",
                  (policy_status,)).fetchall()
              count = 0
              for i in entry:
                  j = list(i)
                  entry.remove(i)
                  entry.insert(count, j)
                  count += 1
              print(entry)
              if entry == None:
                  message = messagebox.showinfo("UNSUCCESSFULL!!",
                                                "EITHER TYPE ACTIVE OR INACTIVE "
                                                "NO OTHER OPTIONS ARE ALLOWED")
                  Label(search_frame, text=message)
                  search_frame.destroy()
              else:
                  try:
                      filename = filedialog.asksaveasfilename(
                          title="Select an excel file", defaultextension=".xlsx",
                          filetypes=(("Excel Workbook (*.xlsx)", "*.xlsx"),
                                     ("all files", "*.*")))
```

```
                    print(filename)

                    dateframe = pandas.DataFrame(numpy.array(entry),
                                           columns=["COMPANY", "CUSTOMER_NAME",
"EMAIL", "CONTACT_NO",
                                                  "ADDRESS",
                                                  "NOMINEE",
                                                  "POLICY_STATUS", " POLICY_NAME ",
"POLICY_NUMBER",
                                                  "ISSUE_DATE ", "MATURITY_DATE",
"PREMIUM_AMOUNT ",
                                                  " SUM_ASSURED  ",
                                                  "POLICY_TERM", "
PREMIUM_PAYING_TERM", "PAYMENT_MODE ",
                                                  "FIRST_YEAR_COMMISSION_PERCENT",
"RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT",
"RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT",
"RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT",
                                                  "GST_ON_FIRST_YEAR_COMMISSION",
                                                  "GST_ON_RENEWAL_COMMISSION",
                                                  "RENEWAL_DATE",
                                                  "FIRST_YEAR_COMMISSION ",
                                                  "RENEWAL_COMMISSION_FOR_2_3_YEAR ",
                                                  "RENEWAL_COMMISSION_FOR_4_5_YEAR ",
"RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS",
                                                  "NOTE "])
                    dateframe.to_excel(str(filename))



                except ValueError as e:
                    message = messagebox.showinfo("UNSUCCESSFULL!!",
                                          "PLEASE MENTION THE FILENAME")
                    Label(search_frame, text=message)
                    search_frame.destroy()

                except:
                    message = messagebox.showinfo("UNSUCCESSFULL!!",
                                          "EITHER TYPE ACTIVE OR INACTIVE "
                                          "NO OTHER OPTIONS ARE ALLOWED")
                    Label(search_frame, text=message)
                    search_frame.destroy()
                else:
                    message = messagebox.showinfo("SUCCESSFULL!!",
                                                "AN EXCEL SHEET HAS BEEN SAVED AT
{}".format(str(filename)))
                    Label(search_frame, text=message)
                    search_frame.destroy()

            view_button = Button(temp_frame, text="SAVE", command=create_function,
width=5, font="Courier", height=1)
            view_button.grid(row=6, column=2, columnspan=3, padx=10, pady=10)

        # SINCE THERE CAN BE MULTIPLE ENTRIES between the the two dates
specified by the users HENCE
        # WE WILL CREATE AN EXCEL FOR ALL THE ENTRIES HAVING THE renewal
date between the two  USER SPECIFIED dates
        # AND THEN LET USER DECIDE WHERE TO SAVE THE EXCEL SHEET
PREPARED BY US CONTAINING ALL THE REQUIRED ENTRIES
        elif radio_button_controller.get() == 3:
            search_frame = Toplevel()
            search_frame.title("ISSUE DATE")
            search_frame.geometry("700x300")
            temp_frame = Frame(search_frame, padx=15, pady=10)
            temp_frame.grid(row=0, column=0, sticky="nsew")
            timeFrame = LabelFrame(temp_frame, text="START DATE")
            timeFrame.grid(row=0, column=0, sticky='new')

            Label(timeFrame, text='YEAR').grid(row=0, column=0)
            year_var = IntVar()
            year_var.set(2000)
            yearSpin = Spinbox(timeFrame, textvariable=year_var, width=5, from_=2000,
to=2099)
            yearSpin.grid(row=1, column=0)
            Label(timeFrame, text='MONTH').grid(row=0, column=1)
            month_var = IntVar()
            month_var.set(1)
            monthSpin = Spinbox(timeFrame, textvariable=month_var, width=5, from_=1,
to=12)
            monthSpin.grid(row=1, column=1)
            Label(timeFrame, text='DAY').grid(row=0, column=2)
            day_var = IntVar()
            day_var.set(1)
            daySpin = Spinbox(timeFrame, textvariable=day_var, width=5, from_=1, to=31)
            daySpin.grid(row=1, column=2)

            timeFrame_1 = LabelFrame(temp_frame, text="END DATE")
            timeFrame_1.grid(row=1, column=0, sticky='new')

            Label(timeFrame_1, text='YEAR').grid(row=0, column=0)
            year_var_1 = IntVar()
            year_var_1.set(2000)
            yearSpin_1 = Spinbox(timeFrame_1, textvariable=year_var_1, width=5,
from_=2000, to=2099)
            yearSpin_1.grid(row=1, column=0)
            Label(timeFrame_1, text='MONTH').grid(row=0, column=1)
            month_var_1 = IntVar()
            month_var_1.set(1)
            monthSpin_1 = Spinbox(timeFrame_1, textvariable=month_var_1, width=5,
from_=1, to=12)
            monthSpin_1.grid(row=1, column=1)
            Label(timeFrame_1, text='DAY').grid(row=0, column=2)
            day_var_1 = IntVar()
            day_var_1.set(1)
            daySpin_1 = Spinbox(timeFrame_1, textvariable=day_var_1, width=5, from_=1,
to=31)
            daySpin_1.grid(row=1, column=2)


            # THIS FUNCTION FINDS ALL THE ENTRIES AND CREATE AN EXCEL
SHEET AND
            # THEN OPEN FILE EXPLORER TO LET USER DECIDE WHERE IT SHOULD
BE SAVED
            def create_function():
                entry = None
                cursor = conn.cursor()
                print(cursor.execute(
                "SELECT ISSUE_DATE,POLICY_NUMBER FROM LIFEINSURANCE
").fetchall())
                entry = cursor.execute(
              "SELECT ISSUE_DATE,POLICY_NUMBER FROM LIFEINSURANCE ",
).fetchall()
            start_date = datetime.datetime.strptime(
              str(year_var.get()) + "-" + str(month_var.get()) + "-" + str(day_var.get()) + " "
+ "00:00:00",
              '%Y-%m-%d %H:%M:%S')
            end_date = datetime.datetime.strptime(str(year_var_1.get()) + "-" +
str(month_var_1.get()) + "-" + str(
              day_var_1.get()) + " " + "00:00:00", '%Y-%m-%d %H:%M:%S')
            policy_number_list = []
            for selected_policy in entry:
              issue_date = datetime.datetime.strptime(selected_policy[0], '%Y-%m-%d
%H:%M:%S')
              if start_date <= issue_date <= end_date:
                policy_number_list.append(selected_policy[1])
            if policy_number_list == []:
              message = messagebox.showinfo("UNSUCCESSFULL!!",
                             "THERE ARE NO ENTRIES BETWEEN THE
SPECIFIED DATES")
              Label(search_frame, text=message)
              search_frame.destroy()
            else:
              entries = []
              for policy_number in policy_number_list:
                statement = cursor.execute(
                  "SELECT * FROM LIFEINSURANCE WHERE POLICY_NUMBER =
?",
                  (policy_number,)).fetchone()
                entries.append(list(statement))
              print(entries)
              try:
                filename = filedialog.asksaveasfilename(
                  title="Select an excel file", defaultextension=".xlsx",
                  filetypes=(("Excel Workbook (*.xlsx)", "*.xlsx"),
                       ("all files", "*.*")))
                print(filename)

                dateframe = pandas.DataFrame(numpy.array(entries),
                               columns=["COMPANY", "CUSTOMER_NAME",
"EMAIL", "CONTACT_NO",
                                              "ADDRESS",
                                              "NOMINEE",
                                              "POLICY_STATUS", " POLICY_NAME ",
"POLICY_NUMBER",
                                              "ISSUE_DATE ", "MATURITY_DATE",
"PREMIUM_AMOUNT ",
                                              " SUM_ASSURED  ",
                                              "POLICY_TERM", "
PREMIUM_PAYING_TERM", "PAYMENT_MODE ",
                                              "FIRST_YEAR_COMMISSION_PERCENT",
"RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT",
"RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT",
"RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT",
                                              "GST_ON_FIRST_YEAR_COMMISSION",
                                              "GST_ON_RENEWAL_COMMISSION",
                                              "RENEWAL_DATE",
                                              "FIRST_YEAR_COMMISSION ",
                                              "RENEWAL_COMMISSION_FOR_2_3_YEAR ",
                                              "RENEWAL_COMMISSION_FOR_4_5_YEAR ",
"RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS",
                                              "NOTE "])
                  dateframe.to_excel(str(filename))



               except ValueError as e:
                  message = messagebox.showinfo("UNSUCCESSFULL!!",
                                        "PLEASE MENTION THE FILENAME")
                  Label(search_frame, text=message)
                  search_frame.destroy()

               except:
                  message = messagebox.showinfo("UNSUCCESSFULL!!",
                                        "THERE ARE NO ENTRIES BETWEEN THE
SPECIFIED DATES")
                  Label(search_frame, text=message)
                  search_frame.destroy()
               else:
                  message = messagebox.showinfo("SUCCESSFULL!!",
                                        "AN EXCEL SHEET HAS BEEN SAVED AT
```

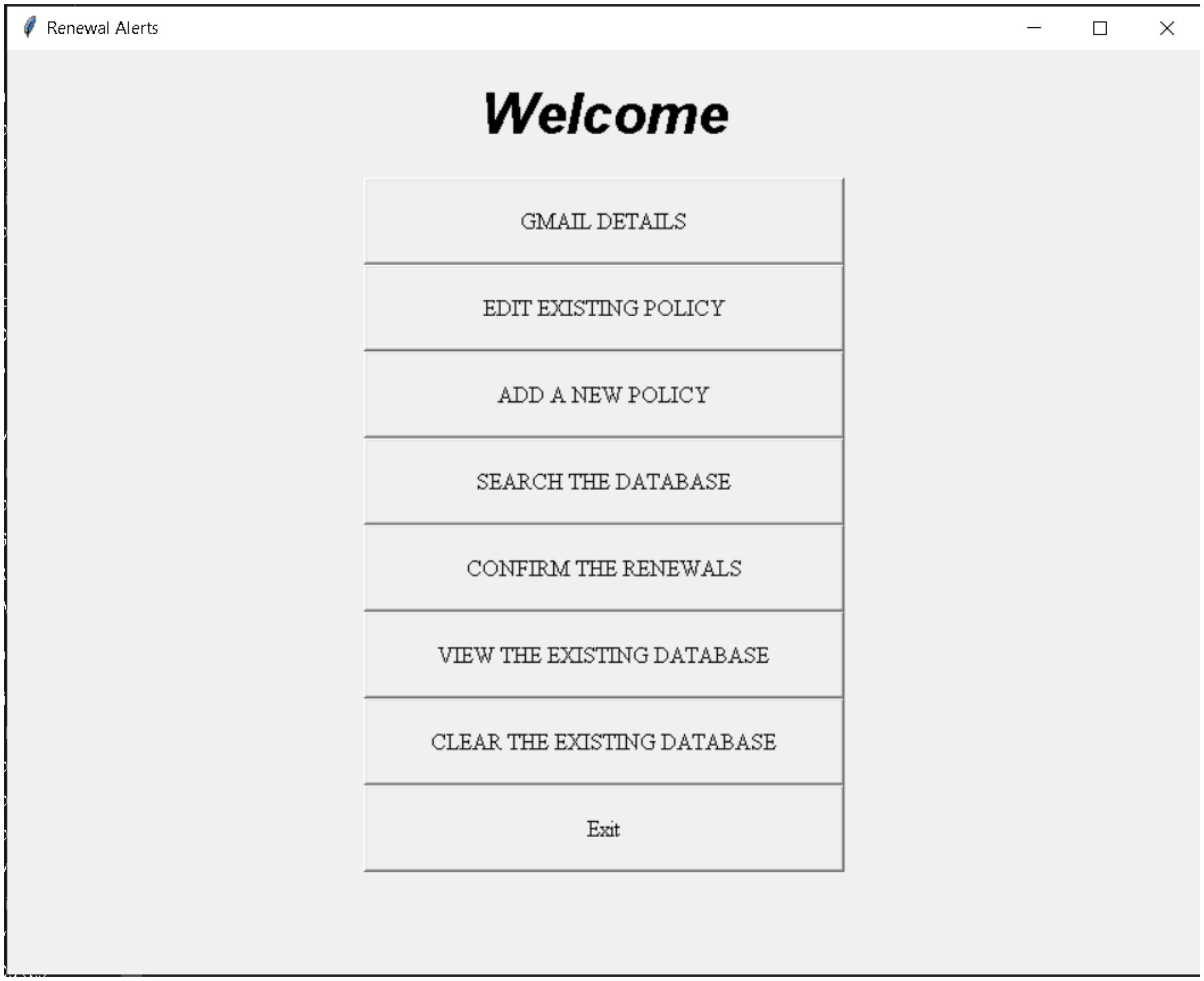
```
{}".format(str(filename)))
                    Label(search_frame, text=message)
                    search_frame.destroy()

          view_button = Button(temp_frame, text="SAVE", command=create_function,
width=5, font="Courier", height=1)
          view_button.grid(row=2, column=0, padx=10, pady=10)

   submit_button = Button(search_frame, text="SUBMIT",
command=submit_the_radio_button_value, padx=3, pady=5, width=5,
                       font="Courier", height=1)
   submit_button.grid(row=6, column=0, columnspan=1, sticky="w")

# this function is called when the users presses on update the renewal button
# the sole purpose of the function is to update the update the reneWal date of the policy
for which the renewal is done
#  for updating the renewal dATE WE MUST KNOW THE PAYMENT MODE
# ONCE THE USER CONFIRMS THAT THE RENEWAL FOR THE SPECIFIC POLICY
NUMBER
# IS DONE WE WILL UPDATE THE RENEWAL DATE DEPENDING THE PAYMENT
MODE
def update_the_renewal_date():
   update_frame = Toplevel(main_window)
   update_frame.title("CONFIRM THE RENEWAL TRANSACTION OF A SINGLE
POLICY")
   update_frame.geometry("700x300")
   label_policy_number = Label(update_frame, text="ENTER THE POLICY NUMBER",
padx=3, pady=5, width=30,
                           font="Courier")
   label_policy_number.grid(row=0, column=0, columnspan=1, sticky="w")
   entry_policy_number = Entry(update_frame, width=30)
   entry_policy_number.grid(row=0, column=2, columnspan=3, sticky="e")

   def confirm_function():
      entry = None
      cursor = conn.cursor()
      entry = cursor.execute("SELECT
POLICY_NUMBER,RENEWAL_DATE,PAYMENT_MODE WHERE
POLICY_NUMBER=?",
                           (entry_policy_number.get(),)).fetchone()
```

```
        cursor.close()
        if entry == None:
            message = messagebox.showinfo("UNSUCCESSFULL!!",
                                          "THERE IS NO POLICY HAVING THE SPECIFIED POLICY
NUMBER ."
                                          "PLEASE CHECK THE POLICY NUMBER AND TRY
AGAIN")
            Label(update_frame, text=message)
            update_frame.destroy()
        else:
            cursor = conn.cursor()
            extracted_date = datetime.datetime.strptime(entry[1], "%Y-%m-%d
%H:%M:%S")
            renewal_date = extracted_date + datetime.timedelta(days=mode[entry[2]])
            cursor.execute("UPDATE LIFEINSURANCE SET RENEWAL_DATE=? WHERE
POLICY_NUMBER=?", (renewal_date, entry[0]))
            cursor.connection.commit()
            conn.commit()
            cursor.close()
            message = messagebox.showinfo("SUCCESSFULL!!",
                                          "DETAILS HAVE BEEN UPDATED")
            Label(update_frame, text=message)
            update_frame.destroy()

    confirm_button = Button(update_frame, text="CONFIRM",
command=confirm_function, width=10, font="Courier", height=1)
    confirm_button.grid(row=1, column=2, columnspan=3, padx=10, pady=10)


# CREATING THE HOME PAGE INTERFACE FOR THE APPLICATION

button_gmail_details = Button(main_window, command=gmail_details, text="GMAIL
DETAILS", padx=30, pady=15, width=30,
                              font="Times")
button_gmail_details.pack()

button_edit_database = Button(main_window, text="EDIT EXISTING POLICY",
command=update_existing_policy, padx=30,
                              pady=15, width=30,
                              font="Times")
```

```
button_edit_database.pack()

button_add_to_batabase = Button(main_window, text="ADD A NEW POLICY",
command=add_new_policy, padx=30, pady=15,
                                width=30, font="Times")
button_add_to_batabase.pack()

button_search_the_database = Button(main_window, text="SEARCH THE
DATABASE", command=search_the_database, padx=30,
                                    pady=15, width=30, font="Times")
button_search_the_database.pack()

button_estimate_the_sales = Button(main_window, text="CONFIRM THE RENEWALS",
command=update_the_renewal_date, padx=30,
                                   pady=15, width=30, font="Times")
button_estimate_the_sales.pack()

button_view_the_existing_database = Button(main_window, text="VIEW THE
EXISTING DATABASE",
                                           command=lambda: subprocess.call(
                                               os.getcwd() + r'\sqlite3 manager\DB Browser for
SQLite.exe'), padx=30,
                                           pady=15, width=30, font="Times")
button_view_the_existing_database.pack()

button_delete_the_existing_database = Button(main_window, text="CLEAR THE
EXISTING DATABASE",
                                             command=clear_the_database, padx=30, pady=15,
width=30, font="Times")
button_delete_the_existing_database.pack()

button_exit = Button(main_window, text="Exit", command=main_window.quit, padx=30,
pady=15, width=30, font="Times")
button_exit.pack()

main_window.mainloop()
conn.close()

main_window.mainloop()
```

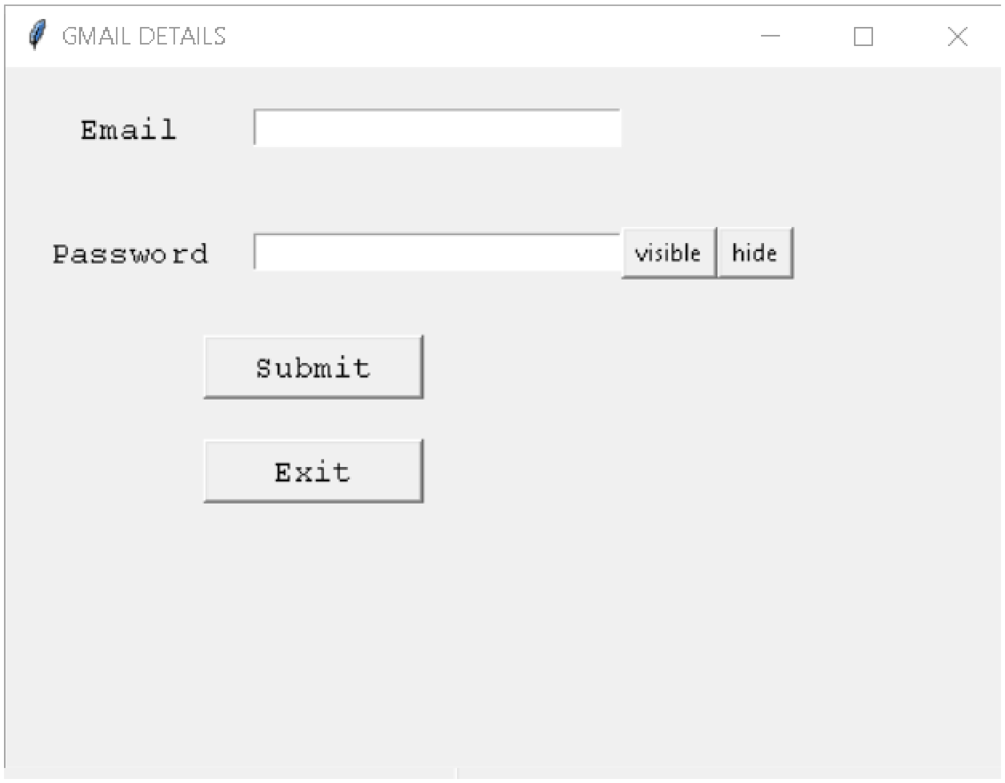
Experiments and Results:

Home page:-

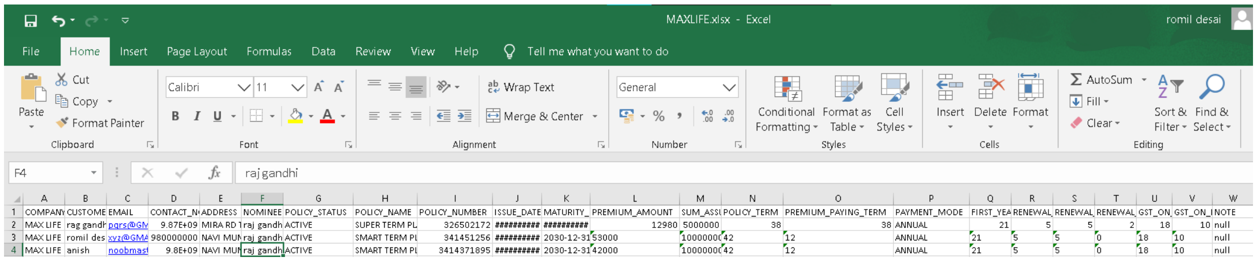


- First you need to provide the google account details to the software in which you want to receive the autogenerated email regarding the alert
- Be sure to add both fields or it will generate an error and you wont get

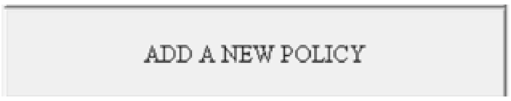
the alerts

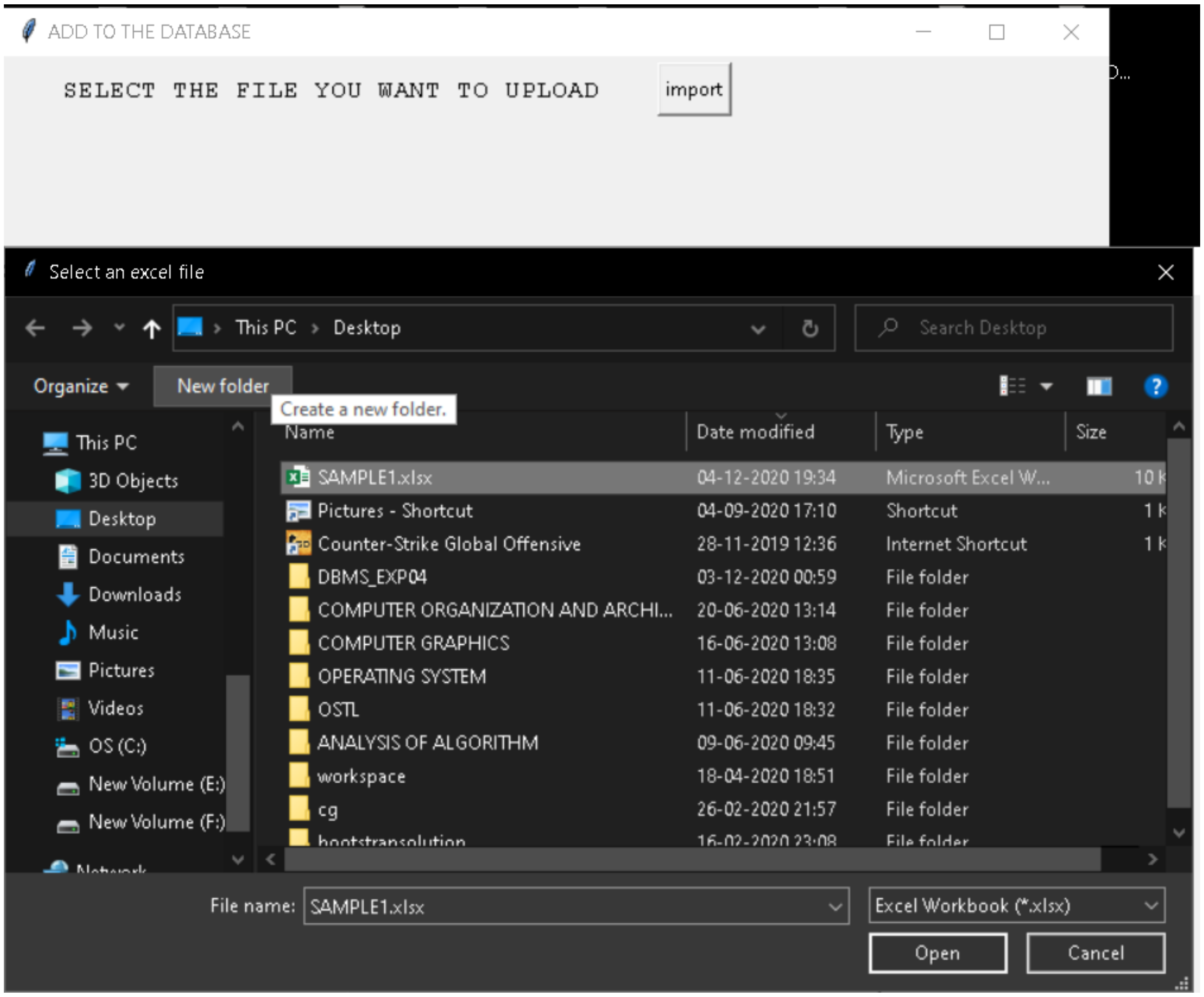


- In order to add a new policy enter the add new policy option and then press on import the excel sheet containing the policy details which you want to add to the database .There's a sample excel sheet given in this folder which you can refer in order to get the attributes order right .



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	COMPANY	CUSTOME	EMAIL	CONTACT_N	ADDRESS	NOMINEE	POLICY_STATUS	POLICY_NAME	POLICY_NUMBER	ISSUE_DATE	MATURITY_	PREMIUM_AMOUNT	SUM_ASSU	POLICY_TERM	PREMIUM_PAYING_TERM	PAYMENT_MODE	FIRST_YEA	RENEWAL_	RENEWAL_	RENEWAL_	GST_ON_	GST_ON_I	NOTE
2	MAX LIFE	rag gandh	pqrs@GM	9.87E+09	MIRA RD 1	raj gandh	ACTIVE	SUPER TERM PL	326502172	##########	##########	12980	5000000	38	38	ANNUAL	21	5	5	2	18	10	null
3	MAX LIFE	romil des	xyz@GMA	980000000	NAVI MUN	raj gandh	ACTIVE	SMART TERM PL	341451256	##########	2030-12-31	53000	10000000	42	12	ANNUAL	21	5	5	0	18	10	null
4	MAX LIFE	anish	noobmast	9.8E+09	NAVI MUN	raj gandh	ACTIVE	SMART TERM PL	3414371895	##########	2030-12-31	42000	10000000	42	12	ANNUAL	21	5	5	0	18	10	null




ADD TO THE DATABASE

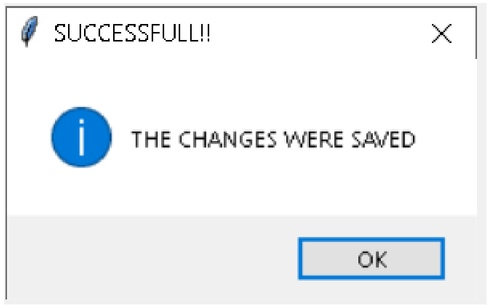
SELECT THE FILE YOU WANT TO UPLOAD

Select an excel file

Name	Date modified	Type	Size
SAMPLE1.xlsx	04-12-2020 19:34	Microsoft Excel W...	10 k
Pictures - Shortcut	04-09-2020 17:10	Shortcut	1 k
Counter-Strike Global Offensive	28-11-2019 12:36	Internet Shortcut	1 k
DBMS_EXP04	03-12-2020 00:59	File folder	
COMPUTER ORGANIZATION AND ARCHI...	20-06-2020 13:14	File folder	
COMPUTER GRAPHICS	16-06-2020 13:08	File folder	
OPERATING SYSTEM	11-06-2020 18:35	File folder	
OSTL	11-06-2020 18:32	File folder	
ANALYSIS OF ALGORITHM	09-06-2020 09:45	File folder	
workspace	18-04-2020 18:51	File folder	
cg	26-02-2020 21:57	File folder	
bootstrapsolution	16-02-2020 23:08	File folder	

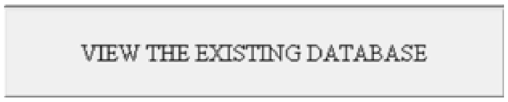
File name: SAMPLE1.xlsx

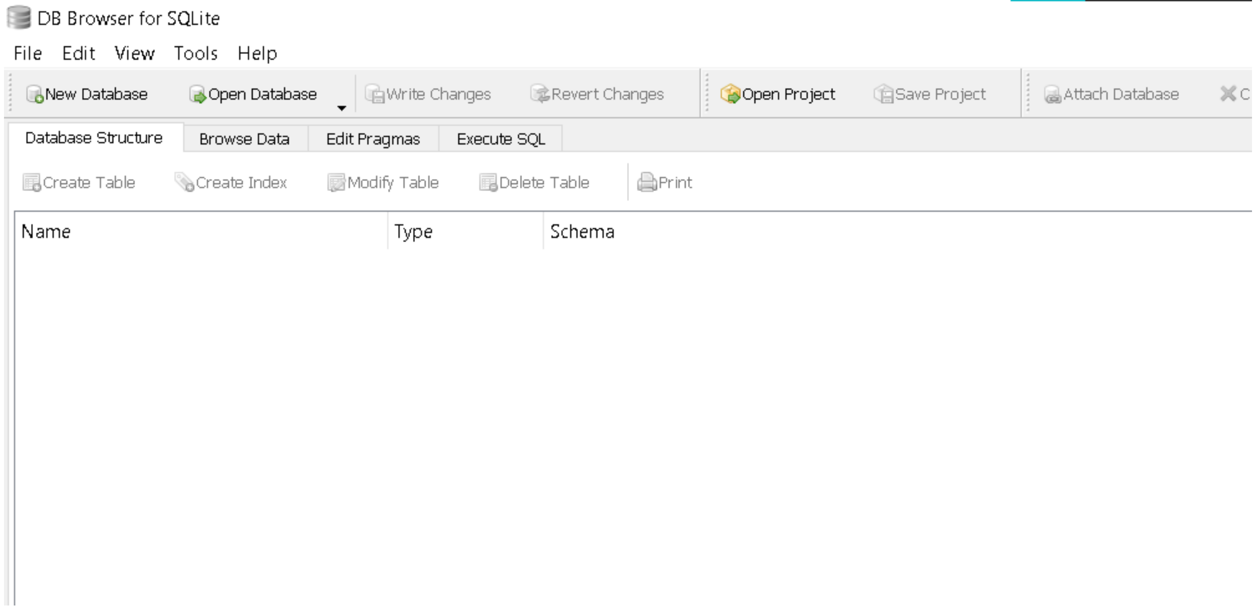
Excel Workbook (*.xlsx)

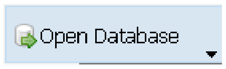

SUCCESSFULL!!

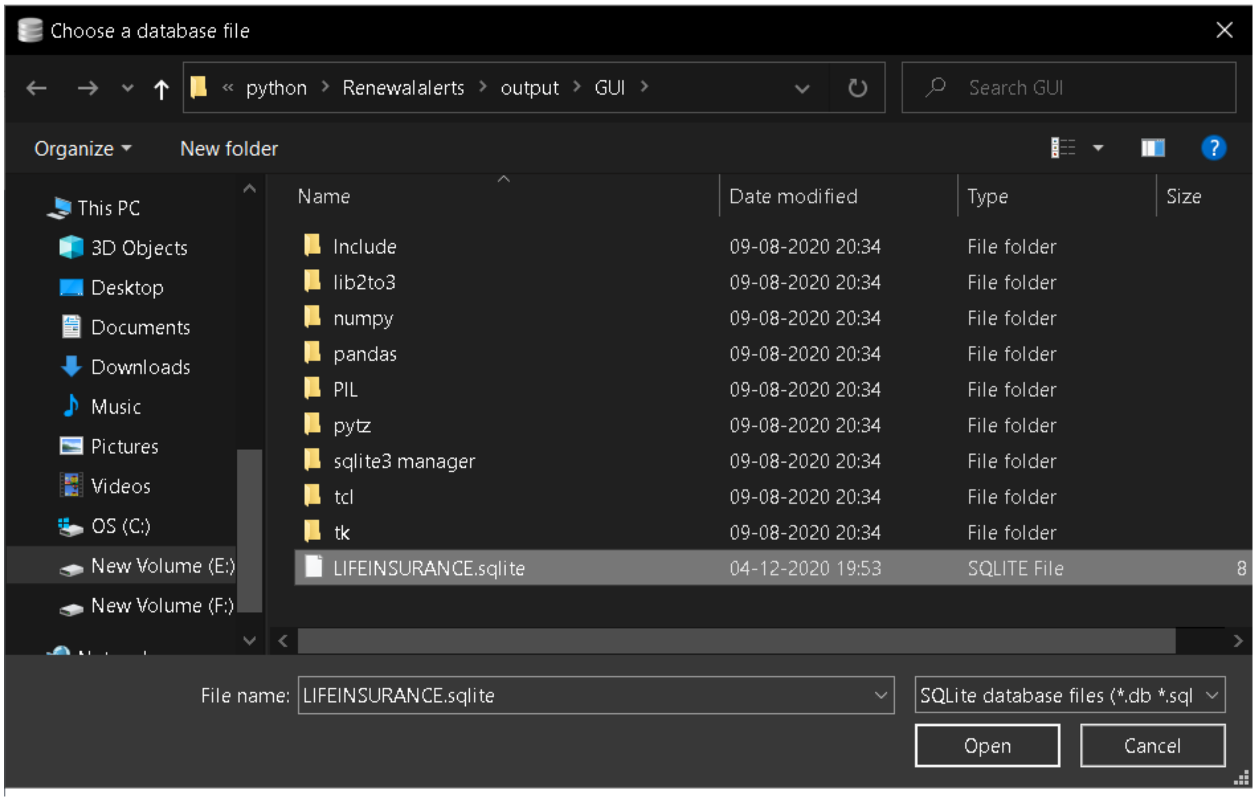
THE CHANGES WERE SAVED

- You can view the database using the given option which will open a new window where you need to click on open database and then you need to select file with a (.sqlite) extension and then you can view the database









Name	Date modified	Type	Size
Include	09-08-2020 20:34	File folder	
lib2to3	09-08-2020 20:34	File folder	
numpy	09-08-2020 20:34	File folder	
pandas	09-08-2020 20:34	File folder	
PIL	09-08-2020 20:34	File folder	
pytz	09-08-2020 20:34	File folder	
sqlite3 manager	09-08-2020 20:34	File folder	
tcl	09-08-2020 20:34	File folder	
tk	09-08-2020 20:34	File folder	
LIFEINSURANCE.sqlite	04-12-2020 19:53	SQLITE File	8

File name: LIFEINSURANCE.sqlite

SQLite database files (*.db *.sql

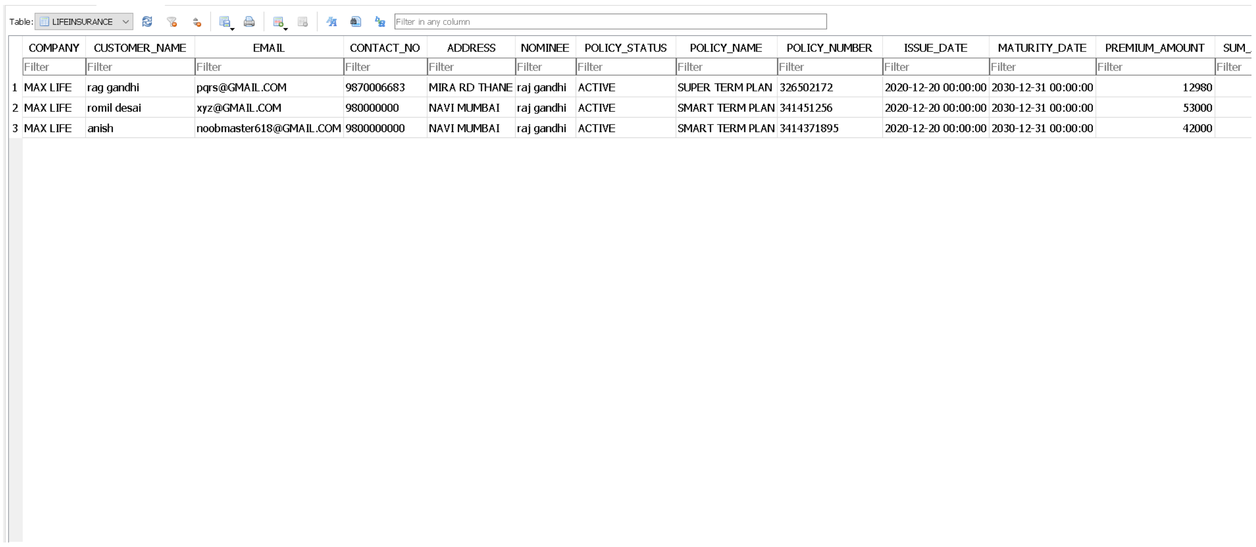
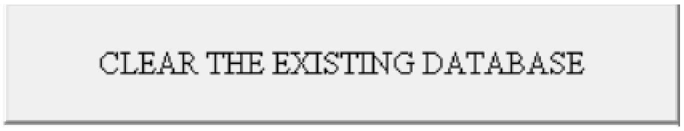


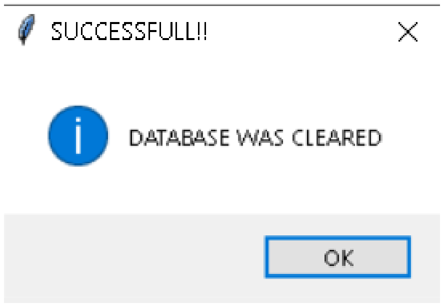
Table: LIFEINSURANCE

Filter in any column

	COMPANY	CUSTOMER_NAME	EMAIL	CONTACT_NO	ADDRESS	NOMINEE	POLICY_STATUS	POLICY_NAME	POLICY_NUMBER	ISSUE_DATE	MATURITY_DATE	PREMIUM_AMOUNT	SUM_
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	MAX LIFE	rag gandhi	pqrs@GMAIL.COM	9870006683	MIRA RD THANE	raj gandhi	ACTIVE	SUPER TERM PLAN	326502172	2020-12-20 00:00:00	2030-12-31 00:00:00	12980	
2	MAX LIFE	romil desai	xyz@GMAIL.COM	980000000	NAVI MUMBAI	raj gandhi	ACTIVE	SMART TERM PLAN	341451256	2020-12-20 00:00:00	2030-12-31 00:00:00	53000	
3	MAX LIFE	anish	noobmaster618@GMAIL.COM	9800000000	NAVI MUMBAI	raj gandhi	ACTIVE	SMART TERM PLAN	3414371895	2020-12-20 00:00:00	2030-12-31 00:00:00	42000	

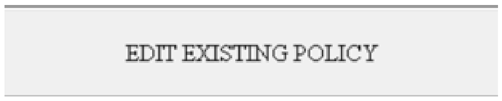
- In order to truncate all the entries from the database click on clear existing database and then you can recheck that by viewing the database again.

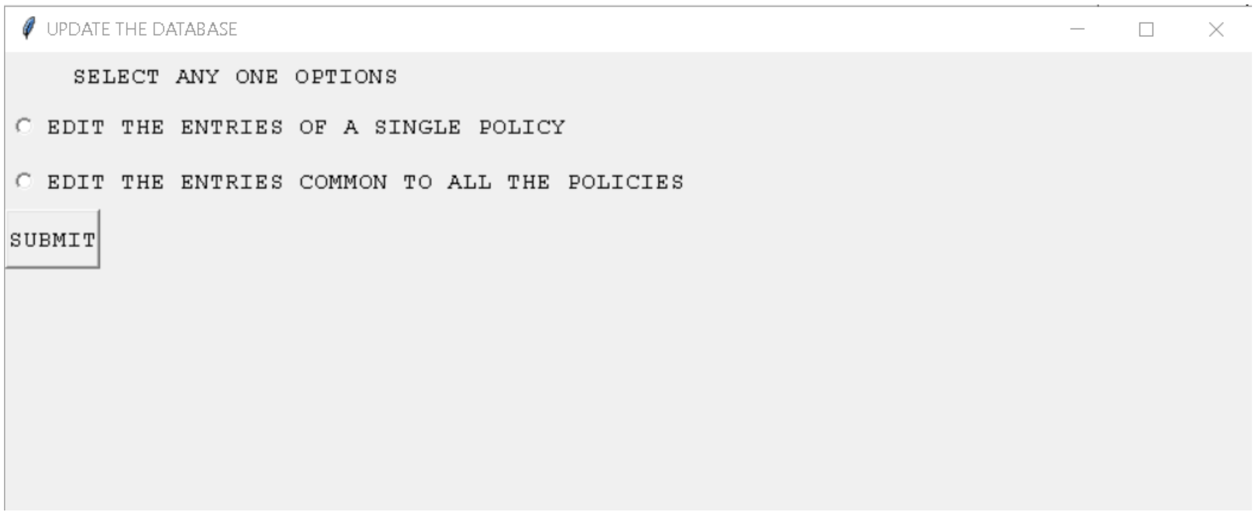



SUCCESSFULL!!

DATABASE WAS CLEARED

- You can update the database using the update the policy option which is pretty straight forward choose the correct option depending on whether you want to change a single entry or you want to change the attributes common to all the entries. Eg:- all the policy have same rate of interest so then we have to change the interest of every entry so you can do this by specifying the policy name.
- But you want to update the personal details of your client then update the entry by specifying the policy number.



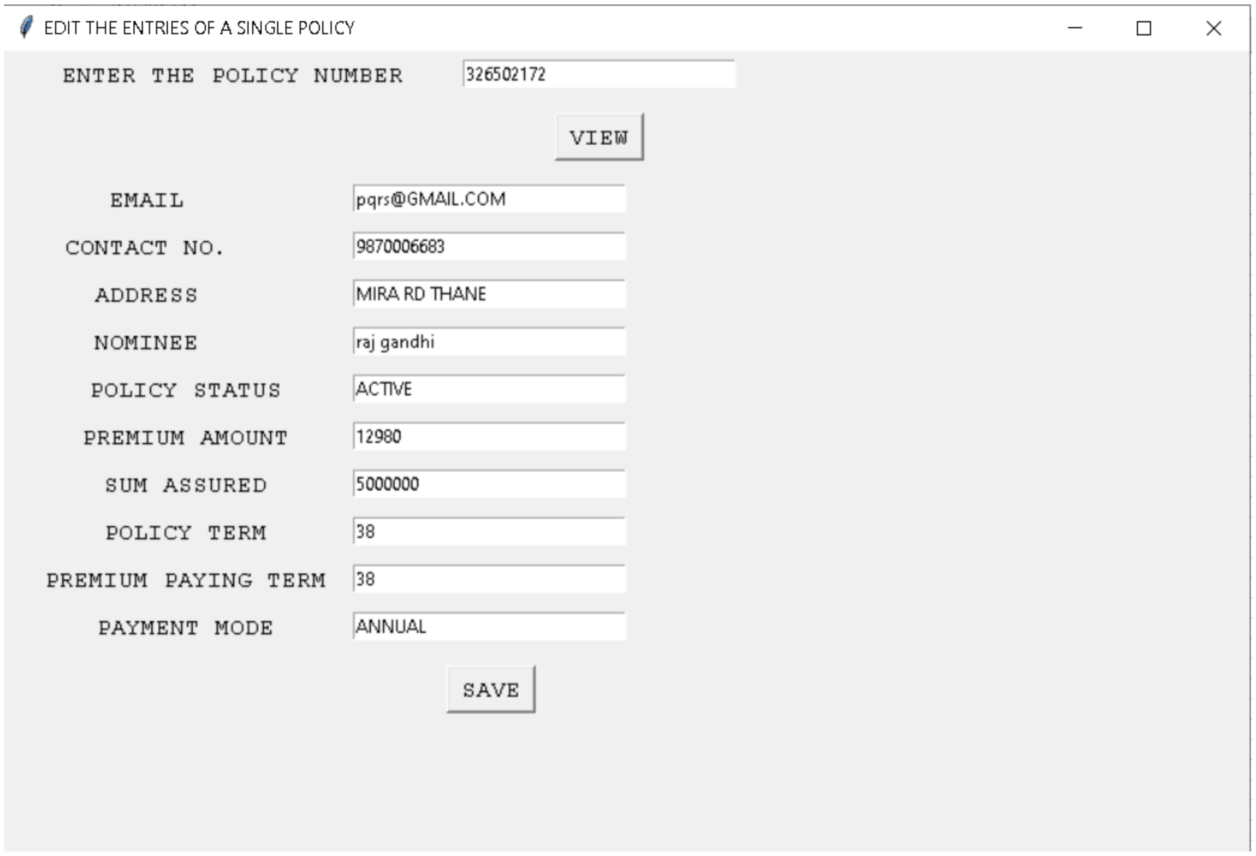


UPDATE THE DATABASE

SELECT ANY ONE OPTIONS

○ EDIT THE ENTRIES OF A SINGLE POLICY

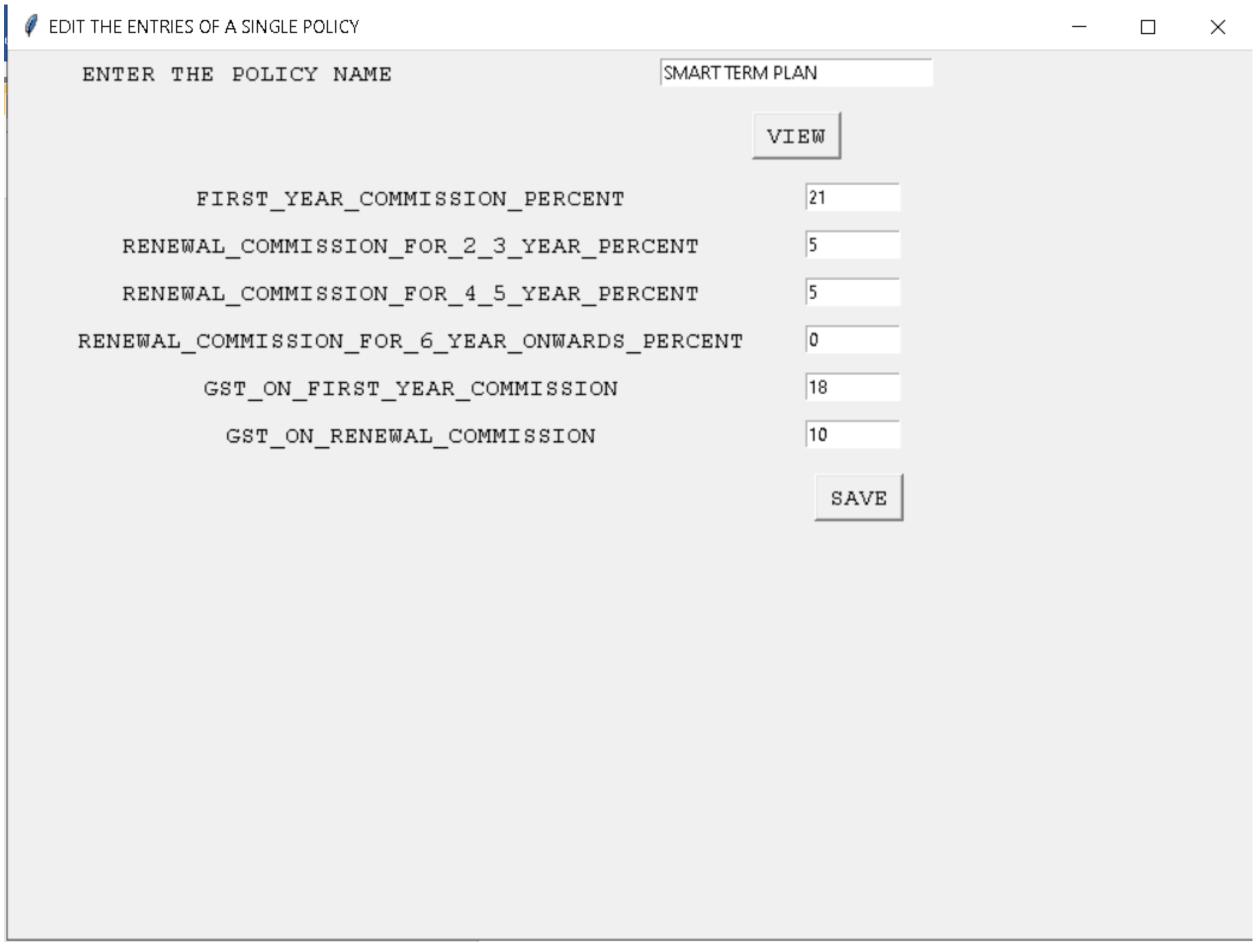
○ EDIT THE ENTRIES COMMON TO ALL THE POLICIES



EDIT THE ENTRIES OF A SINGLE POLICY

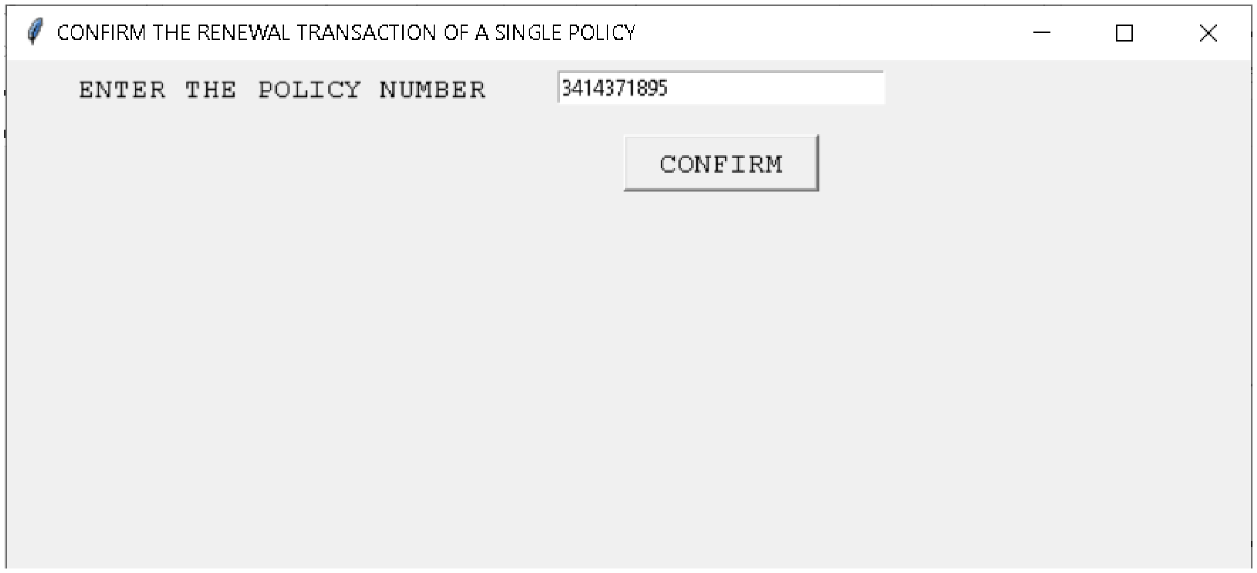
ENTER THE POLICY NUMBER 326502172

EMAIL	pqrs@GMAIL.COM
CONTACT NO.	9870006683
ADDRESS	MIRA RD THANE
NOMINEE	raj gandhi
POLICY STATUS	ACTIVE
PREMIUM AMOUNT	12980
SUM ASSURED	5000000
POLICY TERM	38
PREMIUM PAYING TERM	38
PAYMENT MODE	ANNUAL

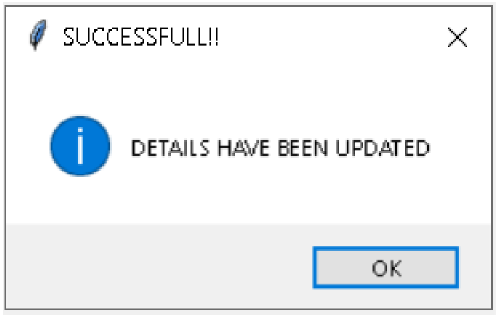


ENTER THE POLICY NAME	SMART TERM PLAN
FIRST_YEAR_COMMISSION_PERCENT	21
RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT	5
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT	5
RENEWAL_COMMISSION_FOR_6_YEAR_ONWARDS_PERCENT	0
GST_ON_FIRST_YEAR_COMMISSION	18
GST_ON_RENEWAL_COMMISSION	10

- Pressing the save button will commit the changes to the database
- Once the client pays their premium the renewal date must be updated ,so in order to perform this task you just need to click on confirm the renewals and then specify the policy number for which the premium has been payed and then the renewal date will be automatically updated in the database.

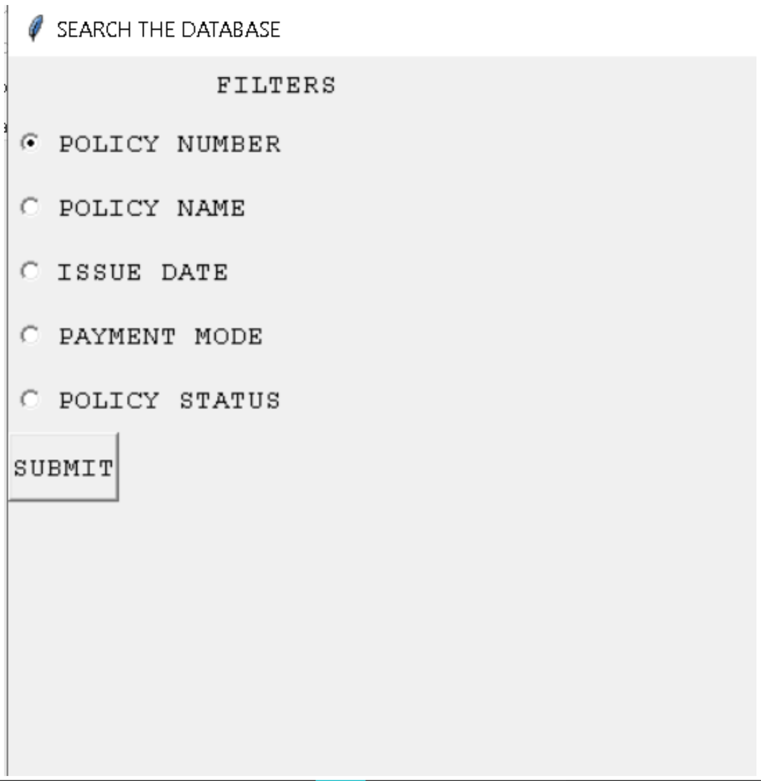

CONFIRM THE RENEWAL TRANSACTION OF A SINGLE POLICY

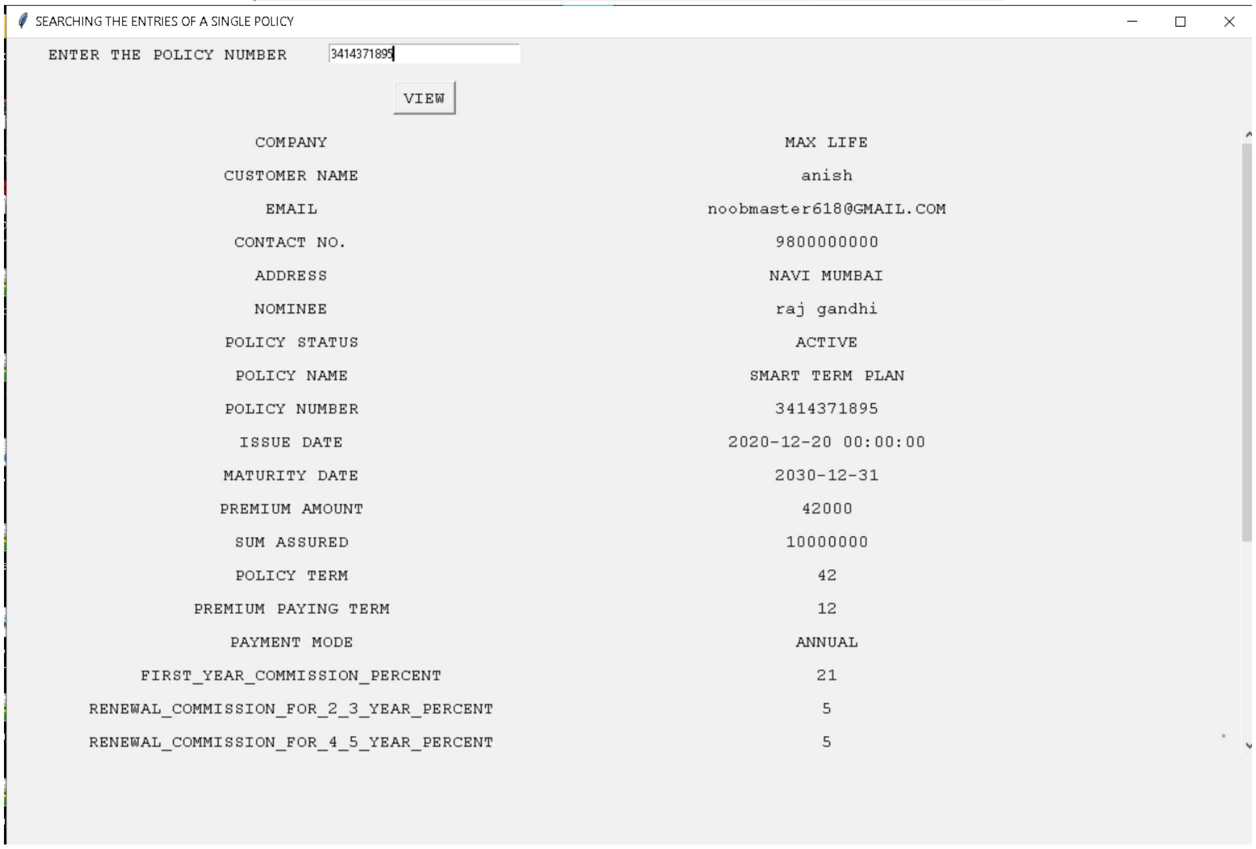
ENTER THE POLICY NUMBER 3414371895


SUCCESSFULL!!

DETAILS HAVE BEEN UPDATED

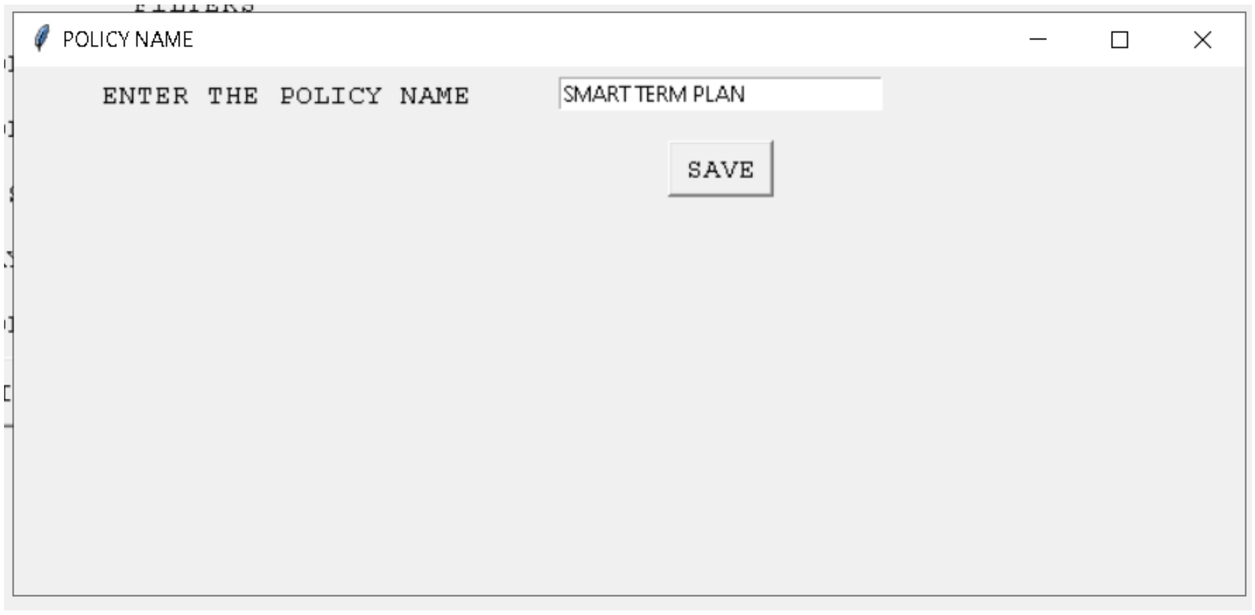
- You can search the database with various attributes such as policy number, policy name ,issue date ,payment mode, policy status.





ENTER THE POLICY NUMBER 3414371895

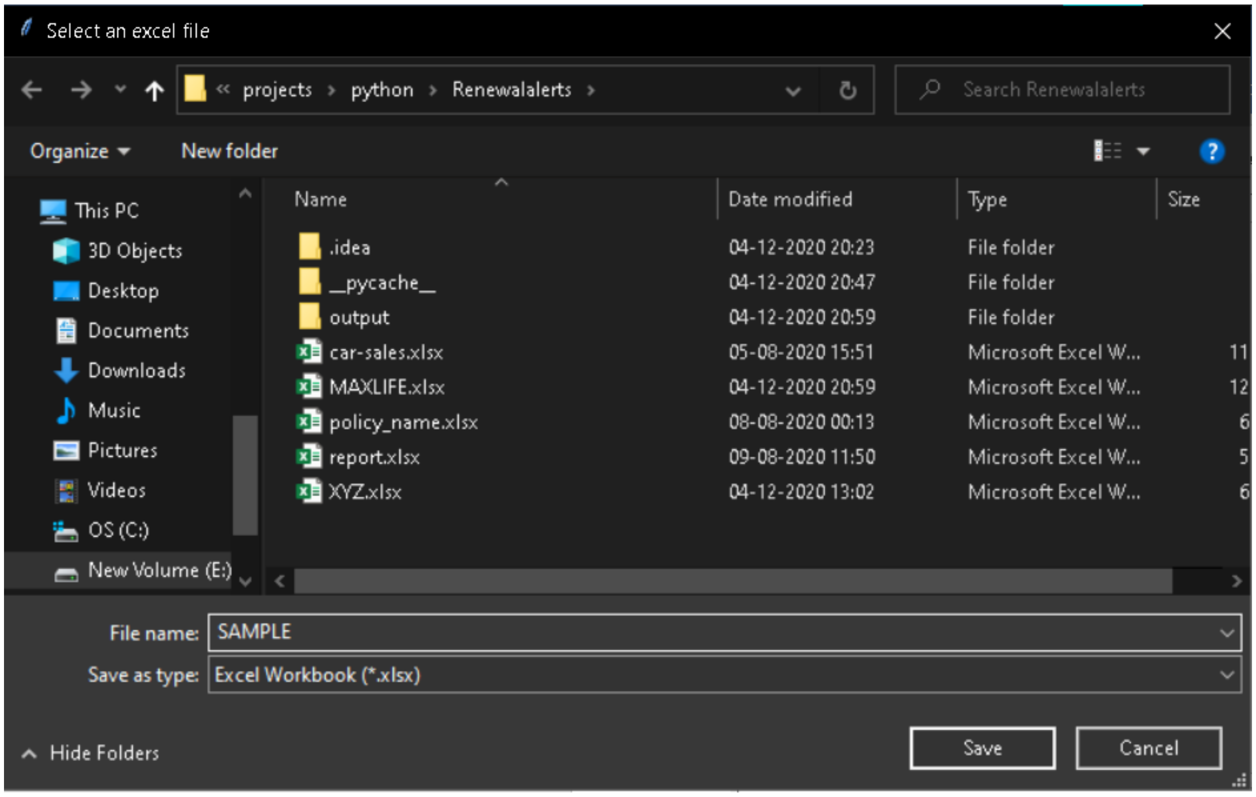
COMPANY	MAX LIFE
CUSTOMER NAME	anish
EMAIL	noobmaster618@GMAIL.COM
CONTACT NO.	9800000000
ADDRESS	NAVI MUMBAI
NOMINEE	raj gandhi
POLICY STATUS	ACTIVE
POLICY NAME	SMART TERM PLAN
POLICY NUMBER	3414371895
ISSUE DATE	2020-12-20 00:00:00
MATURITY DATE	2030-12-31
PREMIUM AMOUNT	42000
SUM ASSURED	10000000
POLICY TERM	42
PREMIUM PAYING TERM	12
PAYMENT MODE	ANNUAL
FIRST_YEAR_COMMISSION_PERCENT	21
RENEWAL_COMMISSION_FOR_2_3_YEAR_PERCENT	5
RENEWAL_COMMISSION_FOR_4_5_YEAR_PERCENT	5



POLICY NAME

ENTER THE POLICY NAME SMART TERM PLAN

SAVE



Select an excel file

« projects › python › Renewalalerts ›

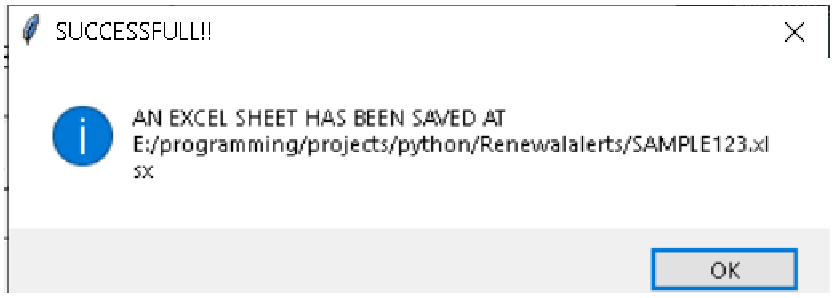
Search Renewalalerts

Organize New folder

Name	Date modified	Type	Size
.idea	04-12-2020 20:23	File folder	
__pycache__	04-12-2020 20:47	File folder	
output	04-12-2020 20:59	File folder	
car-sales.xlsx	05-08-2020 15:51	Microsoft Excel W...	11
MAXLIFE.xlsx	04-12-2020 20:59	Microsoft Excel W...	12
policy_name.xlsx	08-08-2020 00:13	Microsoft Excel W...	6
report.xlsx	09-08-2020 11:50	Microsoft Excel W...	5
XYZ.xlsx	04-12-2020 13:02	Microsoft Excel W...	6

File name: SAMPLE

Save as type: Excel Workbook (*.xlsx)

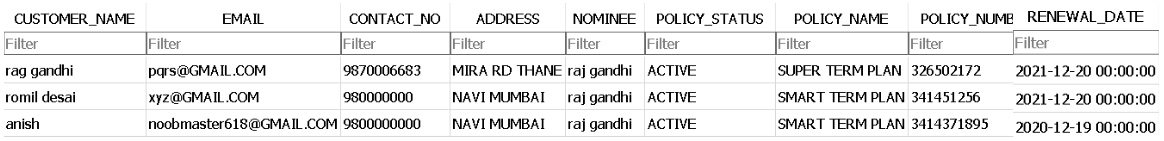


SUCCESSFULL!!

AN EXCEL SHEET HAS BEEN SAVED AT
E:/programming/projects/python/Renewalalerts/SAMPLE123.xl
sx

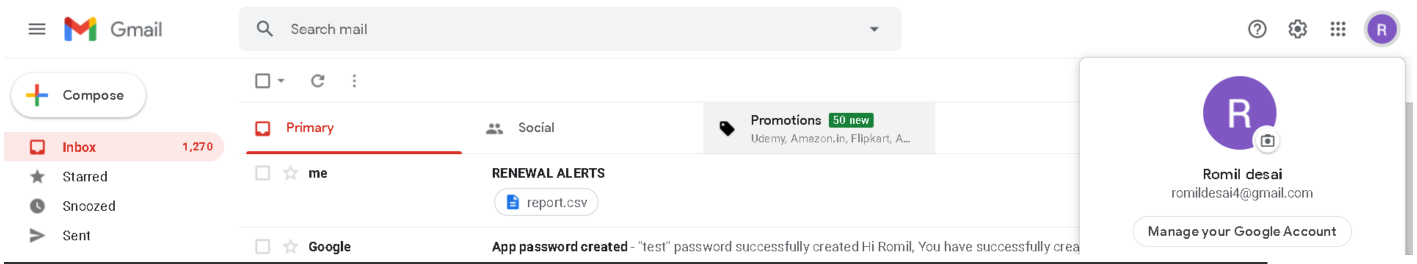
OK

- The sheet contains all the policy with name equal to the specified one.
- You can search similarly for all other options.
- You can get an autogenerated mail to your account regarding the policies whose renewal date is 15 days away from the current date .



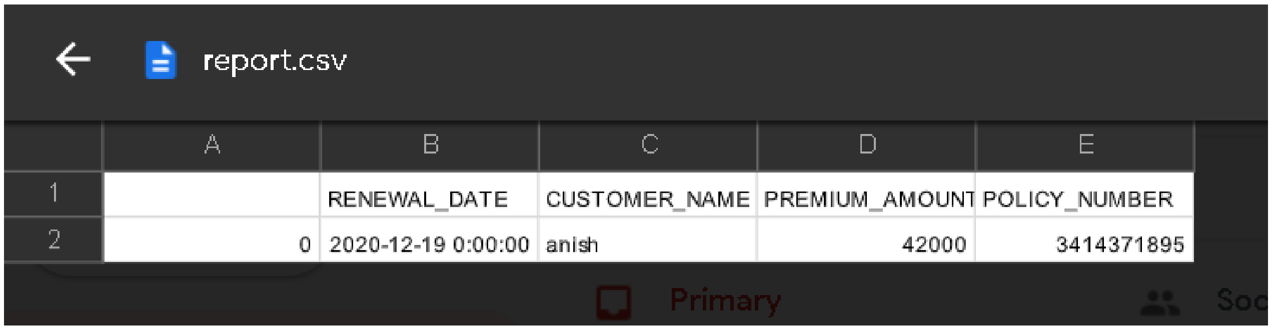
CUSTOMER_NAME	EMAIL	CONTACT_NO	ADDRESS	NOMINEE	POLICY_STATUS	POLICY_NAME	POLICY_NUMB	RENEWAL_DATE
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
rag gandhi	pqrs@GMAIL.COM	9870006683	MIRA RD THANE	raj gandhi	ACTIVE	SUPER TERM PLAN	326502172	2021-12-20 00:00:00
romil desai	xyz@GMAIL.COM	980000000	NAVI MUMBAI	raj gandhi	ACTIVE	SMART TERM PLAN	341451256	2021-12-20 00:00:00
anish	noobmaster618@GMAIL.COM	9800000000	NAVI MUMBAI	raj gandhi	ACTIVE	SMART TERM PLAN	3414371895	2020-12-19 00:00:00

- As you can see that renewal date of Anish policy is after 15 days from the current date (04-12-2020) hence an autogenerated mail is sent to the registered email.



me — RENEWAL ALERTS — report.csv

Google — App password created - "test" password successfully created Hi Romil, You have successfully crea



report.csv

	A	B	C	D	E
1		RENEWAL_DATE	CUSTOMER_NAME	PREMIUM_AMOUNT	POLICY_NUMBER
2	0	2020-12-19 0:00:00	anish	42000	3414371895

- You can cross check the policy number given in the database and the mail.