**EMPLOYEE MANAGEMENT SYSTEM USING PYTHON PROJECT**

This is employee management system in which you can Add, Update , Delete and View employee details as well as you can create a bar Graph of first 5 highest salary employee. I have used various python libraries for creating Graph and adding , deleting, updating etc..Backend I used sqlite3 for storing the whole employee data .Also I have added Quotes which will change frequently .For this I have uses various libraries like bs4 , BeautifulSoup , requests , random etc….

**S1: open command prompt**

**S2: In command prompt create database and table**

C:\DEMO\Python\L16>**sqlite3 abc.db**

SQLite version 3.37.0 2021-11-27 14:13:22

Enter ".help" for usage hints.

**sqlite>** **create table employee(id int primary key, name text ,salary float);**

sqlite> select \* from employee;

**S3: Create file p1.py**

from tkinter import \*

from tkinter import messagebox

from tkinter.messagebox import \*

from tkinter.scrolledtext import \*

from sqlite3 import \*

import matplotlib.pyplot as plt

import random

import requests

from bs4 import BeautifulSoup

s = requests.Session()

url = 'https://randomwordgenerator.com/json/inspirational-quote.json'

response = s.get(url)

# print(response)

data = response.json()['data']

while True:

random\_quote = random.sample(data, 1)

quote = random\_quote[0]['inspirational\_quote']

quote = BeautifulSoup(quote, 'lxml').text

if len(quote)<=60:

break

#print(quote)

def f1():

mw.withdraw()

aw.deiconify()

def f2():

aw.withdraw()

mw.deiconify()

def f3():

mw.withdraw()

vw.deiconify()

vw\_emp\_data.delete(1.0, END)

info = ""

con = None

try:

con = connect("abc.db")

cursor = con.cursor()

sql = "select \* from employee"

cursor.execute(sql)

data = cursor.fetchall()

for d in data:

info = info + " id = " + str(d[0]) + ": name= " + str(d[1]) + ": salary= " + str(d[2]) +"\n"

vw\_emp\_data.insert(INSERT, info)

except Exception as e:

showerror("Mistake",e)

finally:

if con is not None:

con.close()

def f4():

vw.withdraw()

mw.deiconify()

def f5():

mw.withdraw()

uw.deiconify()

def f6():

uw.withdraw()

mw.deiconify()

def f7():

mw.withdraw()

dw.deiconify()

def f8():

dw.withdraw()

mw.deiconify()

def f9():

con = None

try:

con = connect("abc.db")

id = aw\_ent\_id.get()

name = aw\_ent\_name.get()

salary = aw\_ent\_salary.get()

if (len(id)==0 and len(name)==0 and len(salary)==0):

messagebox.showerror("Error","Id, Name and Salary are not entered\nPlease fill the details properly")

else:

id = aw\_ent\_id.get()

if len(id) == 0:

messagebox.showerror("Mistake","Id can't be empty")

aw\_ent\_id.delete(0, END)

aw\_ent\_id.focus()

return

elif not id.isdigit():

messagebox.showerror("Mistake","Id should have only Positive integers")

aw\_ent\_id.delete(0, END)

aw\_ent\_id.focus()

return

elif int(id)<=0 :

messagebox.showerror("Mistake","Id should have only Positive integers")

aw\_ent\_id.delete(0, END)

aw\_ent\_id.focus()

return

else:

id=int(id)

name = aw\_ent\_name.get()

if len(name)==0:

messagebox.showerror("Mistake","Name can't be empty")

aw\_ent\_name.delete(0, END)

aw\_ent\_name.focus()

return

elif not str(name).isalpha():

messagebox.showerror("Mistake","Name should be alphabets")

aw\_ent\_name.delete(0, END)

aw\_ent\_name.focus()

return

elif len(name)<=1:

messagebox.showerror("Mistake","Name should be more than 2 alphabets,eg.om")

aw\_ent\_name.delete(0, END)

aw\_ent\_name.focus()

return

else:

name=name

salary = aw\_ent\_salary.get()

if len(salary) == 0:

messagebox.showerror("Mistake","Salary can't be empty")

aw\_ent\_salary.delete(0, END)

aw\_ent\_salary.focus()

return

elif not salary.isdigit():

messagebox.showerror("Mistake","Salary should have only Positive integers")

aw\_ent\_salary.delete(0, END)

aw\_ent\_salary.focus()

return

elif(float(salary) < 8000) :

messagebox.showerror("Mistake","Salary should be more than 8k")

aw\_ent\_salary.delete(0, END)

aw\_ent\_salary.focus()

return

else:

salary=float(salary)

cursor = con.cursor()

sql = "insert into employee values('%s', '%s' ,'%s')"

cursor.execute(sql % (id, name,salary))

con.commit()

showinfo("Success", "record saved")

aw\_ent\_id.delete(0, END)

aw\_ent\_name.delete(0, END)

aw\_ent\_salary.delete(0, END)

aw\_ent\_id.focus()

except DatabaseError as e:

messagebox.showerror("Error","Id already exists")

aw\_ent\_id.delete(0, END)

aw\_ent\_id.focus()

con.rollback()

finally:

if con is not None:

con.close()

def f10():

con = None

try:

con = connect("abc.db")

id = uw\_ent\_id.get()

name = uw\_ent\_name.get()

salary = uw\_ent\_salary.get()

if (len(id)==0 and len(name)==0 and len(salary)==0):

messagebox.showerror("Error","Id, Name and Salary are not entered\nPlease fill the details properly")

else:

id = uw\_ent\_id.get()

if len(id) == 0:

messagebox.showerror("Mistake","Id can't be empty")

uw\_ent\_id.delete(0, END)

uw\_ent\_id.focus()

return

elif not id.isdigit():

messagebox.showerror("Mistake","Id should have only Positive integers")

uw\_ent\_id.delete(0, END)

uw\_ent\_id.focus()

return

elif int(id)<=0 :

messagebox.showerror("Mistake","Id should have only Positive integers")

uw\_ent\_id.delete(0, END)

uw\_ent\_id.focus()

return

else:

id=int(id)

name = uw\_ent\_name.get()

if len(name)==0:

messagebox.showerror("Mistake","Name can't be empty")

uw\_ent\_name.delete(0, END)

uw\_ent\_name.focus()

return

elif not str(name).isalpha():

messagebox.showerror("Mistake","Name should be alphabets")

uw\_ent\_name.delete(0, END)

uw\_ent\_name.focus()

return

elif len(name)<=1:

messagebox.showerror("Mistake","Name should be more than 2 alphabets,eg.om")

uw\_ent\_name.delete(0, END)

uw\_ent\_name.focus()

return

else:

name=name

salary = uw\_ent\_salary.get()

if len(salary) == 0:

messagebox.showerror("Mistake","Salary can't be empty")

uw\_ent\_salary.delete(0, END)

uw\_ent\_salary.focus()

return

elif not salary.isdigit():

messagebox.showerror("Mistake","Salary should have only Positive integers")

uw\_ent\_salary.delete(0, END)

uw\_ent\_salary.focus()

return

elif(float(salary) < 8000) :

messagebox.showerror("Mistake","Salary should be more than 8k")

uw\_ent\_salary.delete(0, END)

uw\_ent\_salary.focus()

return

else:

salary=float(salary)

cursor = con.cursor()

sql = "update employee set name='%s', salary='%s' where id='%s'"

cursor.execute(sql % (name,salary,id))

if cursor.rowcount==1:

con.commit()

showinfo("Success", "record updated")

uw\_ent\_id.delete(0, END)

uw\_ent\_name.delete(0, END)

uw\_ent\_salary.delete(0, END)

uw\_ent\_id.focus()

else:

raise Exception("record dose not exists")

uw\_ent\_id.delete(0, END)

uw\_ent\_name.delete(0, END)

uw\_ent\_salary.delete(0, END)

uw\_ent\_id.focus()

except Exception as e:

showerror("Mistake", e)

uw\_ent\_id.delete(0, END)

uw\_ent\_id.focus()

finally:

if con is not None:

con.close()

def f11():

id = dw\_ent\_id.get()

con = None

try:

if len(id) == 0:

raise Exception("id can't be empty")

if not str(id).isnumeric():

raise Exception("id should have only Positive integers")

else:

id=int(id)

if id <=0 :

raise Exception("id should have only Positive integers")

con = connect("abc.db")

cursor = con.cursor()

sql = "delete from employee where id='%s'"

cursor.execute(sql%(id))

if cursor.rowcount==1:

con.commit()

showinfo("Success", "record deleted")

else:

raise Exception("record dose not exists")

dw\_ent\_id.delete(0, END)

dw\_ent\_id.focus()

except Exception as e:

showerror("Mistake", e)

dw\_ent\_id.delete(0, END)

dw\_ent\_id.focus()

finally:

if con is not None:

con.close()

def f12():

mydb =connect(database="abc.db")

mycursor = mydb.cursor()

mycursor.execute("select distinct salary,name from employee order by salary desc limit 5; ")

result = mycursor.fetchall

Names = []

Salary = []

for i in mycursor:

Salary.append(i[0])

Names.append(i[1])

print("Name of employee = ", Names)

print("Salary of employee = ", Salary)

plt.bar(Names, Salary)

plt.ylim(0, 500000)

plt.xlabel("Name of Employee")

plt.ylabel("Salary of Employee")

plt.title("Top 5 Salary employee")

plt.show()

mw=Tk()

mw.title("E.M.S")

mw.geometry("800x600+100+100")

f=("Times New Roman",20,"bold")

mw.resizable(False,False)

mw\_btn\_add=Button(mw,text="Add",font=f,width=15,command=f1)

mw\_btn\_view=Button(mw,text="View",font=f,width=15,command=f3)

mw\_btn\_update=Button(mw,text="Update",font=f,width=15,command=f5)

mw\_btn\_delete=Button(mw,text="Delete",font=f,width=15,command=f7)

mw\_btn\_chart=Button(mw,text="Chart",font=f,width=15,command=f12)

lblQuote=Label(mw,text="Quote:"+quote,font=("Times New Roman",20,'bold'),bg="light yellow")

mw.configure(bg="light yellow")

y=10

mw\_btn\_add.pack(pady=y)

mw\_btn\_view.pack(pady=y)

mw\_btn\_update.pack(pady=y)

mw\_btn\_delete.pack(pady=y)

mw\_btn\_chart.pack(pady=y)

lblQuote.pack(pady=y)

aw=Toplevel(mw)

aw.title("Add Emp")

aw.geometry("800x600+100+100")

aw.configure(bg="light gray")

aw.resizable(False,False)

aw\_lab\_id=Label(aw,text="enter id:",font=f,bg="light gray")

aw\_ent\_id=Entry(aw,font=f,bd=4)

aw\_lab\_name=Label(aw,text="enter name:",font=f,bg="light gray")

aw\_ent\_name=Entry(aw,font=f,bd=4)

aw\_lab\_salary=Label(aw,text="enter salary:",font=f,bg="light gray")

aw\_ent\_salary=Entry(aw,font=f,bd=4)

aw\_btn\_save=Button(aw,text="Save",font=f,command=f9)

aw\_btn\_back=Button(aw,text="Back",font=f,command=f2)

y=10

aw\_lab\_id.pack(pady=y)

aw\_ent\_id.pack(pady=y)

aw\_lab\_name.pack(pady=y)

aw\_ent\_name.pack(pady=y)

aw\_lab\_salary.pack(pady=y)

aw\_ent\_salary.pack(pady=y)

aw\_btn\_save.pack(pady=y)

aw\_btn\_back.pack(pady=y)

aw.withdraw()

vw = Toplevel(mw)

vw.title("View Emp")

vw.geometry("800x600+100+100")

vw.configure(bg="pink")

vw.resizable(False,False)

vw\_emp\_data = ScrolledText(vw, width=40, height=10, font=f,bg="pink")

vw\_btn\_back = Button(vw, text="Back", font=f,command=f4)

vw\_emp\_data.pack(pady=y)

vw\_btn\_back.pack(pady=y)

vw.withdraw()

uw = Toplevel(mw)

uw.title("Update Emp")

uw.geometry("800x600+100+100")

uw.configure(bg="peach puff")

uw.resizable(False,False)

uw\_lab\_id=Label(uw,text="enter id:",font=f,bg="peach puff")

uw\_ent\_id=Entry(uw,font=f,bd=4)

uw\_lab\_name=Label(uw,text="enter name:",font=f,bg="peach puff")

uw\_ent\_name=Entry(uw,font=f,bd=4)

uw\_lab\_salary=Label(uw,text="enter salary:",font=f,bg="peach puff")

uw\_ent\_salary=Entry(uw,font=f,bd=4)

uw\_btn\_save=Button(uw,text="Save",font=f,command=f10)

uw\_btn\_back=Button(uw,text="Back",font=f,command=f6)

y=10

uw\_lab\_id.pack(pady=y)

uw\_ent\_id.pack(pady=y)

uw\_lab\_name.pack(pady=y)

uw\_ent\_name.pack(pady=y)

uw\_lab\_salary.pack(pady=y)

uw\_ent\_salary.pack(pady=y)

uw\_btn\_save.pack(pady=y)

uw\_btn\_back.pack(pady=y)

uw.withdraw()

dw = Toplevel(mw)

dw.title("Delete Emp")

dw.geometry("800x600+100+100")

dw.configure(bg="light blue")

dw.resizable(False,False)

dw\_lab\_id=Label(dw,text="enter id:",font=f,bg="light blue")

dw\_ent\_id=Entry(dw,font=f,bd=4)

dw\_btn\_save=Button(dw,text="Save",font=f,command=f11)

dw\_btn\_back=Button(dw,text="Back",font=f,command=f6)

y=10

dw\_lab\_id.pack(pady=y)

dw\_ent\_id.pack(pady=y)

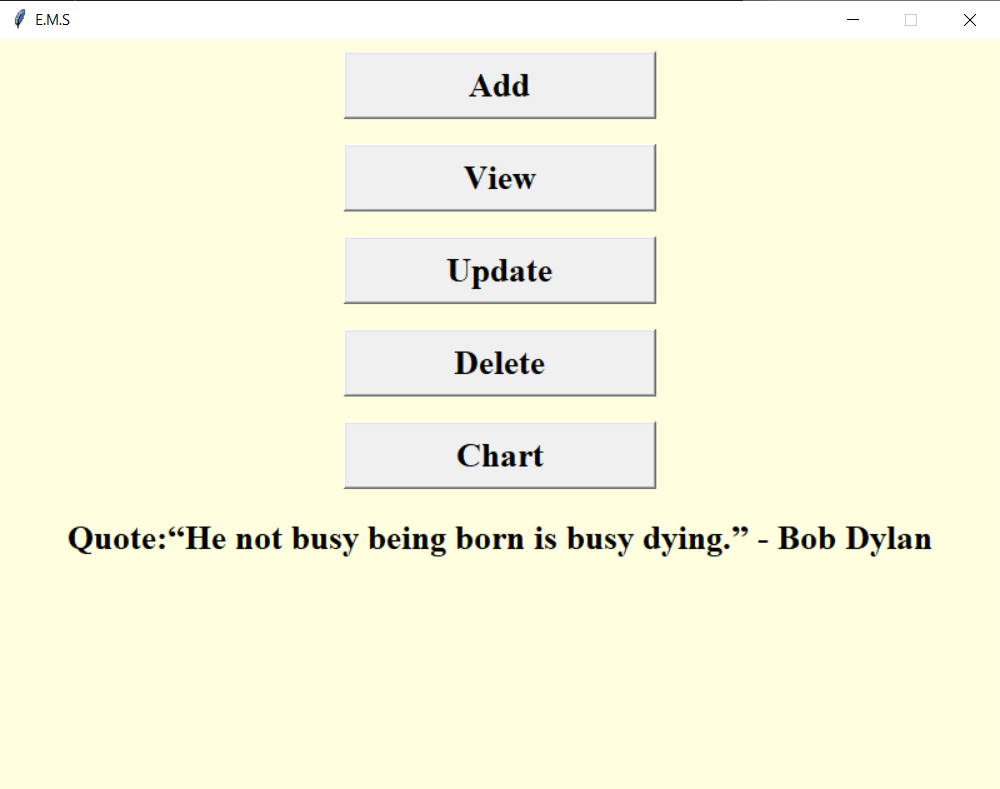
dw\_btn\_save.pack(pady=y)

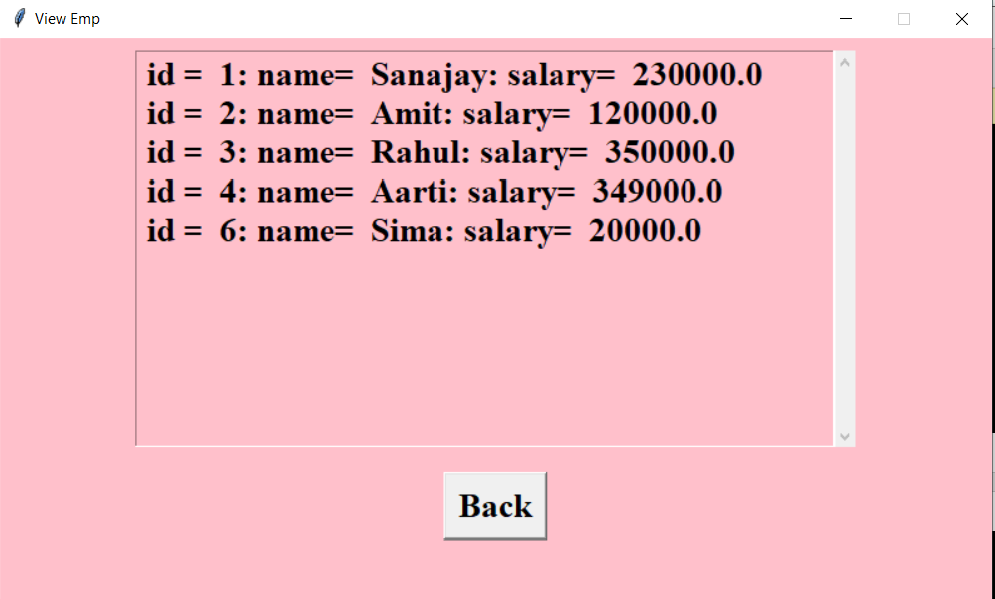
dw\_btn\_back.pack(pady=y)

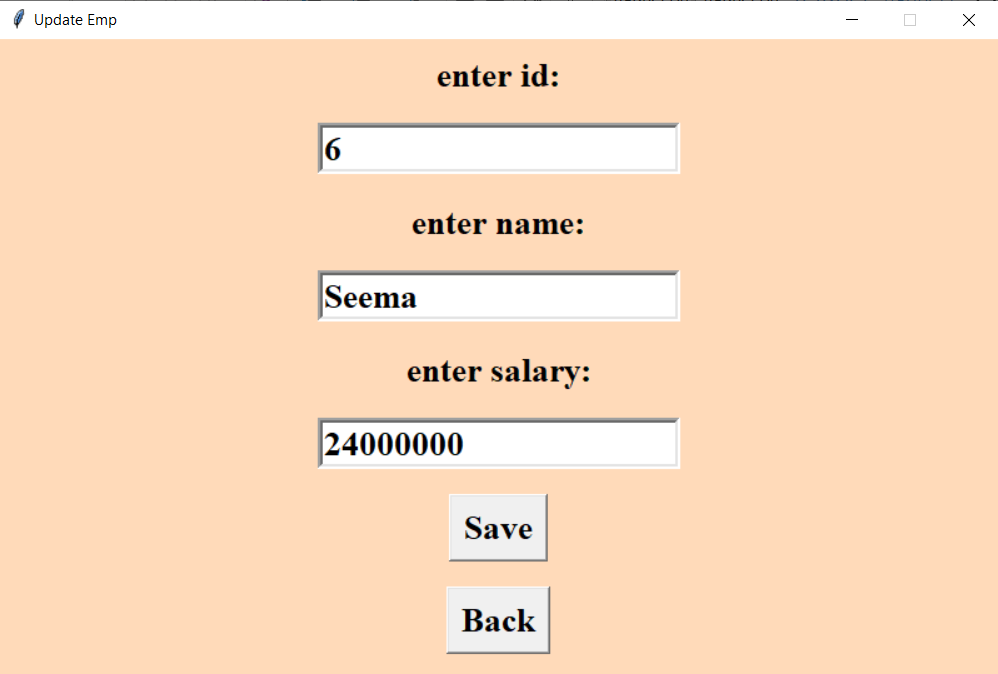
dw.withdraw()

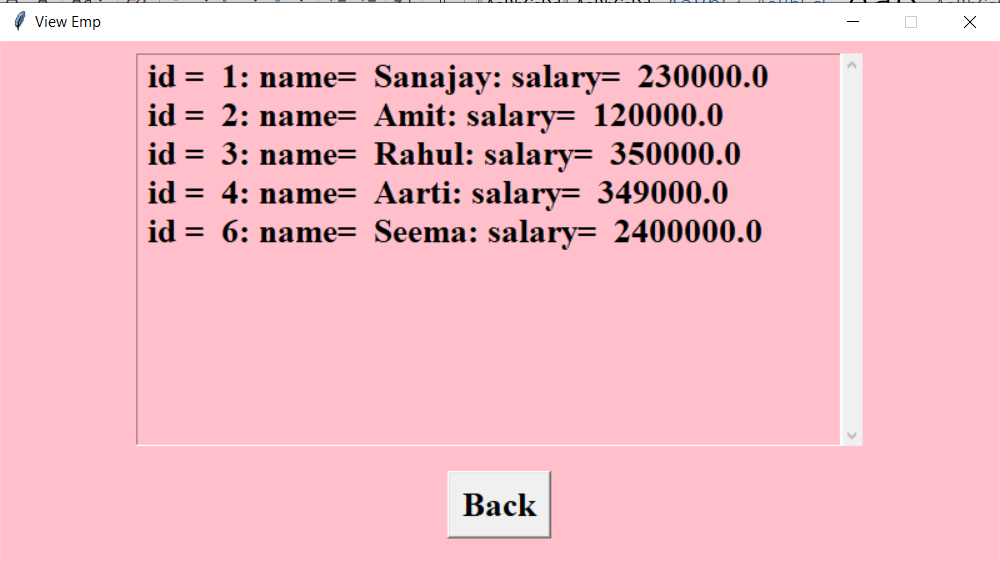
mw.mainloop()

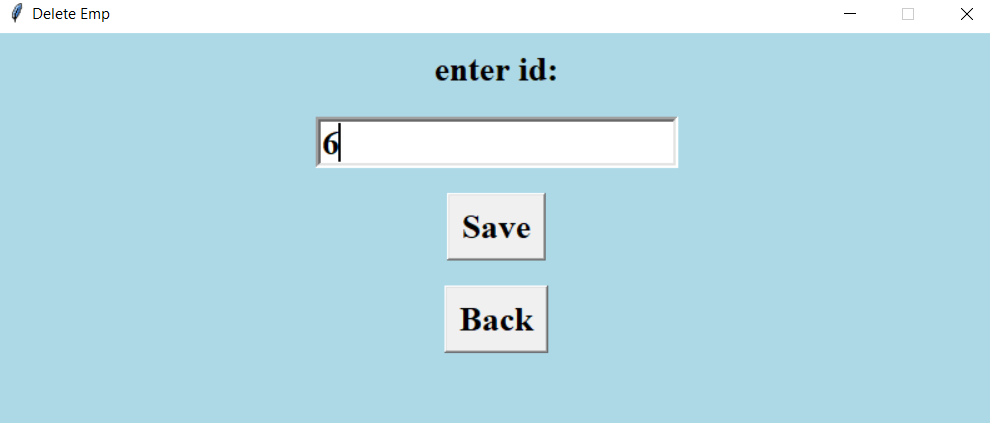
**S4: run p1.py**

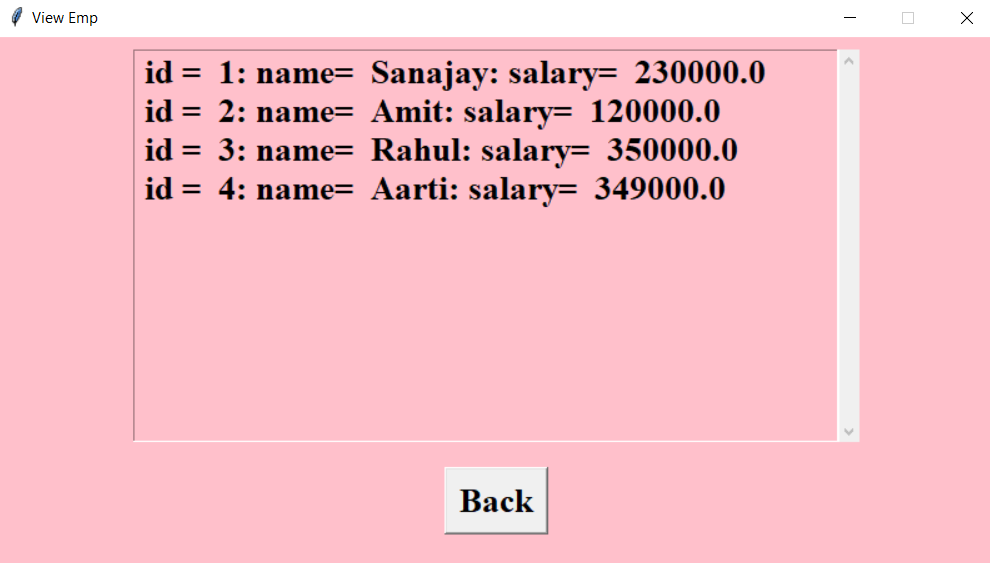


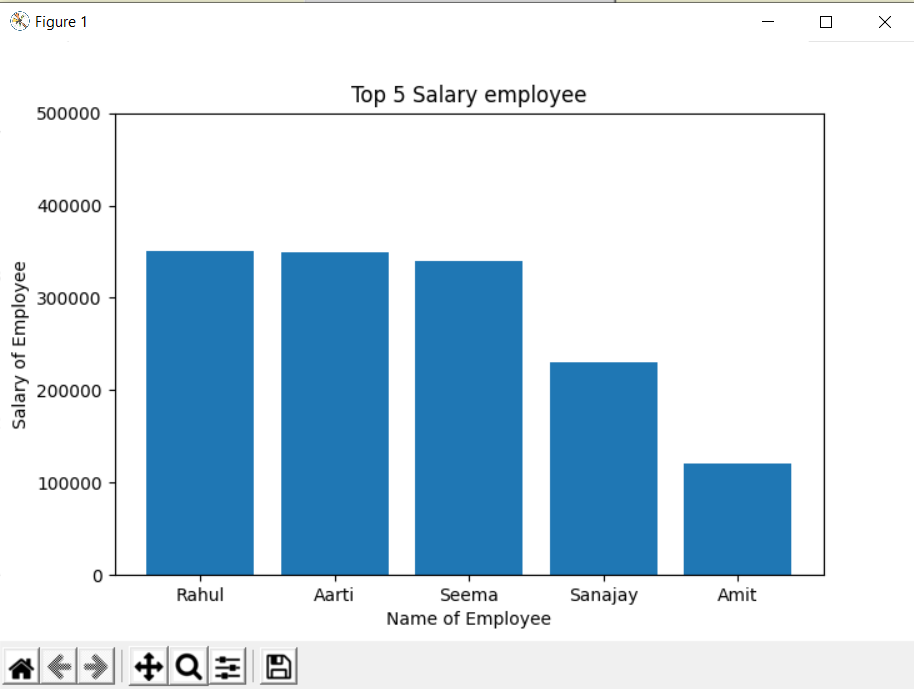












**S5:Validations:**

1. **Id should not be empty ,id should have only positive integers.**
2. **If id is ok ,the check validation of name should not be empty and name should contain only alphabets with min. length 2 eg. Om is ok**
3. **If name is ok, then check validation for salary ,salary should not be empty and should be min. 8k.**

**This all validations are covered in this project.**