**Topic: Result Analysis**

**Group Members:**

Hrishikesh M. Vichore 116A3062

Yash Kokane 217A3068

**Guided By:**

Dr. Vijay Katkar

Prof. Varsha Mali

Prof. Bushra Shaikh

**Software Used:**

Liclipse, Anaconda Python, Xampp, MySQL Database

**Code:**

import pandas as pd

import tkinter as tk

import openpyxl as xl

import os

from shutil import copyfile as cp

from tkinter import filedialog

from tkinter import Menu

from tkinter import messagebox

import re

import MySQLdb as ms

class Analyser:

def \_\_init\_\_(*self*,parent):

*self*.con = ms.connect('localhost','root','','miniproject')

*self*.root = parent

*self*.root.title("Main frame")

*self*.frame = tk.Frame(parent)

*self*.frame.grid()

menubar = Menu(*self*.frame)

menubar1 = Menu(menubar,tearoff = 0)

menubar1.add\_command(label = "New",command = *self*.new)

menubar1.add\_command(label = "Open Existing",command = '')

menubar1.add\_separator()

menubar1.add\_command(label = "Quit",command = *self*.root.quit)

menubar1.add\_command(label = "Restart",command = *self*.restart)

menubar.add\_cascade(label = "File", menu = menubar1)

*self*.root.config(menu = menubar)

def new(*self*):

file = filedialog.askopenfile(parent = *self*.frame,title = "Choose a file",mode = "rb")

if file != None:

*self*.filename = *self*.src = os.path.split(file.name)

*self*.filename = *self*.filename[1]

*self*.filename = *self*.filename

*self*.dst1 = *self*.src[0]

*self*.src = *self*.src[0]+"/"+*self*.src[1]

tk.Label(*self*.frame,text="File is {filename}".format(filename=*self*.filename),font=('Helvetica','15')).grid(row=0,column=1)

if not os.path.isfile(*self*.filename):

dst = "C:/Users/Admin/My Documents/LiClipse Workspace/Result\_Analysis3/{filename}".format(filename=*self*.filename)

cp(*self*.src,dst)

*self*.frame\_grid()

else:

*self*.frame\_grid()

def frame\_grid(*self*):

branch = tk.Button(*self*.frame,text = "Branch Analysis")

tk.Label(*self*.frame,text = 'Enter Branch Name :').grid(row = 1 , column = 0)

bname=tk.Entry(*self*.frame)

bname.grid(row = 1 , column = 1)

bname.focus()

branch.grid(row = 2 , column = 0)

ClassName = tk.Button(*self*.frame, text = "Class Analysis")

ClassName.bind("<Button-1>",lambda event , bname=bname : *self*.ClassWise(event,bname))

ClassName.grid(row = 2, column = 1)

bname.bind("<Return>",lambda event , bname=bname : *self*.BranchWise(event,bname))

branch.bind("<Button-1>", lambda event , bname=bname : *self*.BranchWise(event,bname))

TopperName = tk.Button(*self*.frame, text = "Topper's List")

TopperName.grid(row = 2, column = 2)

TopperName.bind("<Button-1>", lambda event , bname=bname : *self*.TopperList(event,bname))

def TopperList(*self*,event,bname):

filename = *self*.filename

bname = bname.get()

bname=bname.upper()

sep = ','

pd.options.mode.chained\_assignment = None

df = pd.read\_csv(filename,sep=sep)

if bname:

OverAllTopper = df.sort\_values("CGPA",ascending = False)

OverAllTopper = OverAllTopper.head(10)

df = df.query("Branch == @bname")

df = df.fillna(-2)

df = df.set\_index('Sr\_No')

s=df.shape

s=s[0] #Appeared

if not s == -1:

BranchTopper = df.sort\_values("CGPA",ascending = False)

BranchTopper = BranchTopper.head(10)

writer = pd.ExcelWriter("Toppers.xlsx",engine = 'xlsxwriter')

BranchTopper.to\_excel(writer,sheet\_name = 'Branch Topper')

OverAllTopper.to\_excel(writer,sheet\_name = 'OverAll Topper')

messagebox.showinfo("File Saved", "File Saved")

cp('Toppers.xlsx',"g:/Toppers.xlsx")

def ClassWise(*self*,event, bname):

destFile = 'g:/Result analysis\_Class.xlsx'

cp('ClassWise Analysis.xlsx', destFile)

workbook = xl.load\_workbook(destFile)

sheet = workbook.active

filename = *self*.filename

bname = bname.get()

bname=bname.upper()

sep = ','

P\_F\_Columns = ['R\_AM', 'R\_AP', 'R\_AC', 'R\_ED', 'R\_SPA', 'R\_CS']

Sub\_Columns = ['AM', 'AP', 'AC', 'ED', 'SPA', 'CS']

T\_Columns = ['T\_AM', 'T\_AP', 'T\_AC', 'T\_ED', 'T\_SPA', 'T\_CS']

pd.options.mode.chained\_assignment = None

df = pd.read\_csv(filename,sep=sep)

if bname:

df = df.query("Branch == @bname")

df = df.fillna(-2)

df = df.set\_index('Sr\_No')

s=df.shape

s=s[0] #Appeared

if not s == -1:

Absent = []

for col in P\_F\_Columns:

df[col].replace('P', 1, inplace = True)

df[col].replace('F', -1, inplace = True)

for col in T\_Columns:

df[col].replace('#VALUE!', -5, inplace = True)

for col in Sub\_Columns :

df[col].replace("AB", -3, inplace = True)

for col in Sub\_Columns :

int\_df = df[col].astype(int)

Absent.append(sum(map(lambda x: x == -3,int\_df)))

PassedSubject = []

for col in P\_F\_Columns :

int\_df = df[col].astype(int)

PassedSubject.append(sum(map(lambda x: x == 1,int\_df)))

for k in range(len(Sub\_Columns)):

sheet.cell(row = 2+k, column = 1).value = (Sub\_Columns[k])

sheet.cell(row = 2+k, column = 2).value = int(s - Absent[k])

sheet.cell(row = 2+k, column = 3).value = int(PassedSubject[k] - Absent[k])

sheet.cell(row = 2+k, column = 4).value = int(s - PassedSubject[k])

sheet.cell(row = 2+k, column = 5).value = int(Absent[k])

workbook.save(destFile)

messagebox.showinfo("Saved", "File Succesfully Saved")

a = re.findall(r"[\w']+",filename)

*self*.TableName = ''

for i in range(len(a)):

a[i] = a[i] + '\_'

*self*.TableName += a[i]

try:

df.to\_sql(name = *self*.TableName, con = *self*.con, flavor = "mysql", if\_exists = "replace",index = False)

*self*.con.commit()

*self*.con.close()

except :

pass

def BranchWise(*self*,event,bname):

destFile = 'g:/Result analysis\_Branch.xlsx'

cp('Result analysis.xlsx', destFile)

wb = xl.load\_workbook(destFile)

sheet = wb.active

filename = *self*.filename

bname = bname.get()

bname=bname.upper()

sep = ','

P\_F\_Columns = ['R\_AM', 'R\_AP', 'R\_AC', 'R\_ED', 'R\_SPA', 'R\_CS']

INT\_Columns = ['INT\_AM', 'INT\_AP', 'INT\_AC', 'INT\_ED', 'INT\_SPA', 'INT\_CS']

Sub\_Columns = ['AM', 'AP', 'AC', 'ED', 'SPA', 'CS']

T\_Columns = ['T\_AM', 'T\_AP', 'T\_AC', 'T\_ED', 'T\_SPA', 'T\_CS']

pd.options.mode.chained\_assignment = None

df = pd.read\_csv(filename,sep=sep)

if bname:

df = df.query("Branch == @bname")

df = df.fillna(-2)

df = df.set\_index('Sr\_No')

s=df.shape

s=s[0] #Appeared

if not s == -1:

dict1={}

dict2={}

dict3={}

for col in P\_F\_Columns:

df[col].replace('P', 1, inplace = True)

df[col].replace('F', -1, inplace = True)

for col in T\_Columns:

df[col].replace('#VALUE!', -5, inplace = True)

for col in T\_Columns:

int\_df = df[col].astype(int)

below\_lvl = map(lambda x: x < 50, int\_df)

below\_lvl = (sum(below\_lvl)/s)\*100

between\_lvl = map(lambda x: 50 < x < 60, int\_df)

between\_lvl = (sum(between\_lvl)/s)\*100

above\_lvl = map(lambda x: x > 60, int\_df)

above\_lvl = (sum(above\_lvl)/s)\*100

dict1[col] = below\_lvl

dict2[col] = between\_lvl

dict3[col] = above\_lvl

for col in Sub\_Columns :

df[col].replace("AB", -3, inplace = True)

df["External\_KT"] = 0

df["Internal\_KT"] = 0

for row in range(df.shape[0]):

counter = 0

counter1 = 0

for col in range (len(P\_F\_Columns)):

TheoryMarks = int(df.iloc[row][Sub\_Columns[col]])

TotalMarks = int(df.iloc[row][T\_Columns[col]])

InternalMarks = int(df.iloc[row][INT\_Columns[col]])

if df.iloc[row][P\_F\_Columns[col]] == -1 :

if((TheoryMarks >= 32) and (TotalMarks < 40) ):

counter += 1

df.Internal\_KT.iloc[row] = counter

if((TheoryMarks < 32) and (TotalMarks < 40) and (InternalMarks < 8)):

counter += 1

df.Internal\_KT.iloc[row] = counter

if (TheoryMarks < 32) :

counter1 += 1

df.External\_KT.iloc[row] = counter1

int\_df = df["GT"].astype(int)

failed = sum(map(lambda x : x == -2, int\_df))

passed = s-failed

#passpercent = (passed/s)\*100

failed = int(failed)

passed = int(passed)

ExtKT = []

IntKT = []

Cursor = *self*.con.cursor()

for i in range (6):

int\_df = df["External\_KT"].astype(int)

int\_df1 = df["Internal\_KT"].astype(int)

ExtKT.append(sum(map(lambda x : x == i+1,int\_df)))

IntKT.append(sum(map(lambda x : x == i+1,int\_df1)))

j = list(dict1.values())

k = list(dict1.keys())

for i in range(len(k)):

SubjectName = k[i].split("\_")

SubjectName = SubjectName[1]

sheet.cell(row = 6+i, column = 1).value = SubjectName

sheet.cell(row = 6+i, column = 3).value = j[i]

sql = "SELECT Faculty FROM subject\_faculty WHERE Subject = %s "

data = [k[i]]

Cursor.execute(sql,data)

TeacherName = Cursor.fetchone()

TeacherName = TeacherName[0]

sheet.cell(row = 6+i, column = 2).value = TeacherName

j = list(dict2.values())

k = list(dict1.keys())

for i in range(len(k)):

sheet.cell(row = 6+i, column = 4).value = j[i]

j = list(dict3.values())

k = list(dict1.keys())

for i in range(len(k)):

sheet.cell(row = 6+i, column = 5).value = j[i]

sheet.cell(row = 15, column = 1).value = s

sheet.cell(row = 15, column = 3).value = failed

sheet.cell(row = 15, column = 4).value = passed

for i in range (6):

sheet.cell(row = 16, column = 12-i).value = int(ExtKT[i])

sheet.cell(row = 15, column = 12-i).value = int(IntKT[i])

wb.save(destFile)

messagebox.showinfo("Saved", "File Succesfully Saved")

a = re.findall(r"[\w']+",filename)

*self*.TableName =''

for i in range(len(a)):

a[i] = a[i] + '\_'

*self*.TableName += a[i]

try:

df.to\_sql(name = *self*.TableName, con = *self*.con, flavor = "mysql", if\_exists = "replace",index = False)

*self*.con.commit()

*self*.con.close()

except:

pass

elif s == -1:

messagebox.showerror("Invalid Branch", "No such Branch exists")

elif not bname:

messagebox.showerror("Empty", "No Value Provided")

def restart(*self*):

*self*.root.destroy()

Analyser(root)

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

#root.geometry("800x300")

app = Analyser(root)

root.mainloop()

**ScreenShots:**

