INTERNSHIP DOCUMENTATION

SSN COLLEGE OF ENGINEERING

Company:PATTERNS COGNITIVE

INTERNSHIP DETAILS:

Name:Lokkshanaa.S
Position:Intern
Duration:01-07-2024 to 31-07-2024

Date: 01/07/2024 **Day: Monday**

Task 1: Write a simple calculator python program to perform addition, subtraction, multiplication and division.

Program code:

```
#calculator
n1=int(input("\nEnter the first number:"))
n2=int(input("\nEnter the second number:"))
op=int(input("\nEnter the operation to be done: \n1.Add \n2.Subtract \n3.Multiply \n4.Division \n"))
if(op==1):
    print("Addition operation is done")
    result=n1+n2
elif(op==2):
    print("Subtraction operation is done")
    result=n1-n2
elif(op==3):
    print("Multiplication operation is done")
    result=n1*n2
elif(op==4):
    print("Division operation is done")
    result=n1/n2
else:
    print("Invalid input given ")
print("Result is",result)
```

Input and output:

Case 1: Addition

```
Enter the first number:3468

Enter the second number:892

Enter the operation to be done:
1.Add
2.Subtract
3.Multiply
4.Division
1
Addition operation is done
Result is 4360
```

Case 2: Subtraction

```
Enter the first number:89754

Enter the second number:7354

Enter the operation to be done:
1.Add
2.Subtract
3.Multiply
4.Division
2
Subtraction operation is done
Result is 82400
```

Case 3: Multiplication

```
Enter the first number:8794

Enter the second number:123

Enter the operation to be done:
1.Add
2.Subtract
3.Multiply
4.Division
3
Multiplication operation is done
Result is 1081662
```

Case 4: Division

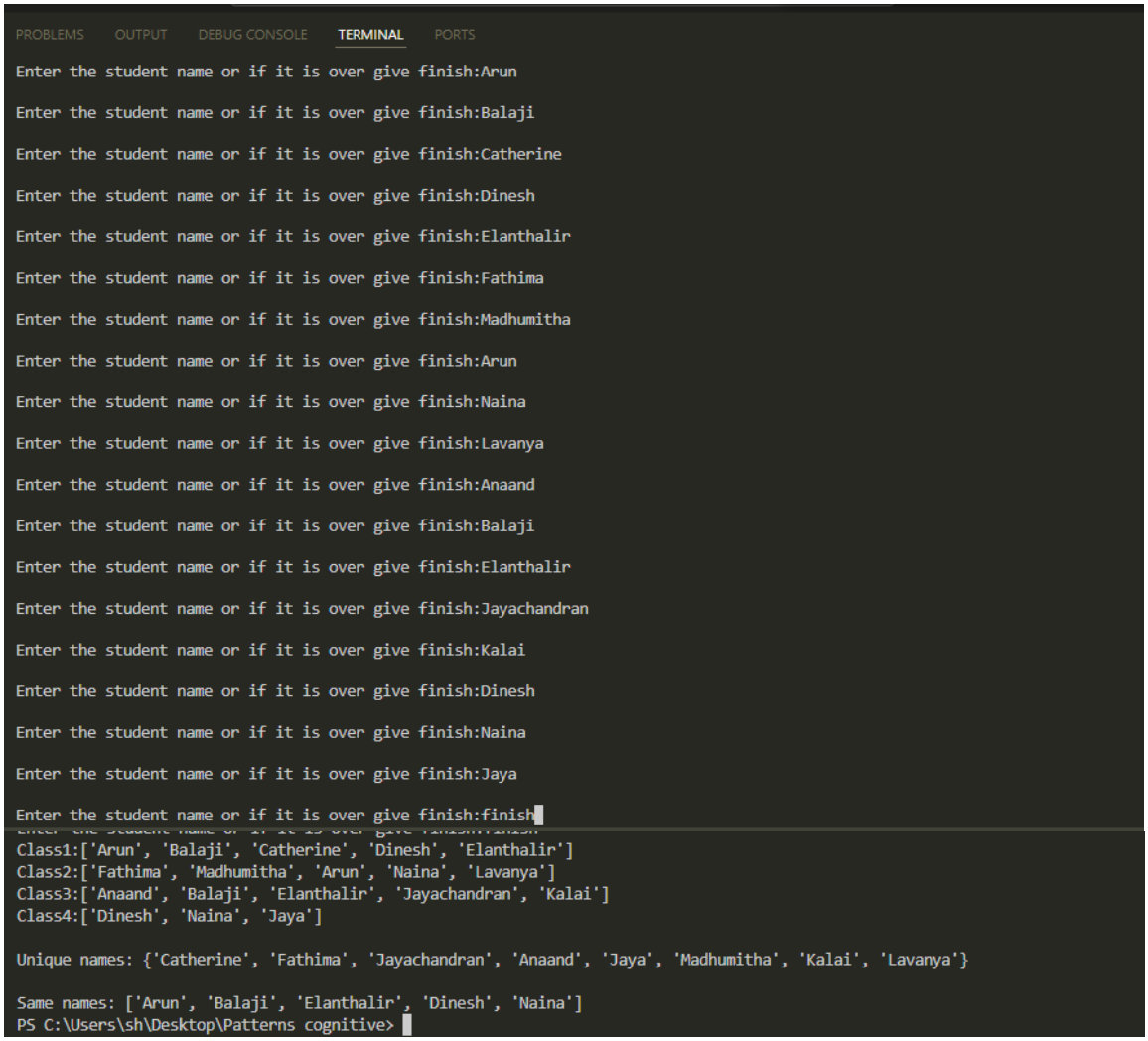
```
Enter the first number:1083

Enter the second number:18

Enter the operation to be done:
1.Add
2.Subtract
3.Multiply
4.Division
4
Division operation is done
Result is 60.166666666666664
```

Task 2: Write a Python program to create classes of students where each class contains a maximum of 5 students. Continuously accept student names as input until "finish" is entered. If a student's name appears more than once, it should be added to a list of repeated names and excluded from further classes. Display the grouped classes, unique student names, and repeated names.

Program code:

```python
stud_list=[]
current=[]
same_name=[]
unique_name=set()
while True:
    stud_name=input("\nEnter the student name or if it is over give finish:")
    if stud_name=="finish":
        break
    if stud_name in same_name:
        continue
    else:
        if stud_name in unique_name:
            unique_name.remove(stud_name)
            same_name.append(stud_name)
        else:
            unique_name.add(stud_name)
    current.append(stud_name)
    if len(current)==5:
        stud_list.append(current)
        current=[]
if current:
    stud_list.append(current)
for i,stud_list in enumerate(stud_list):
    print(f"Class{i+1}:{stud_list}")
print("\nUnique names:", unique_name)
print("\nSame names:", same_name)
```

Input and output:



Date: 02/07/2024 **Day: Tuesday**

Task : Write a Python program to collect and display student details. The program should prompt the user to enter basic information, including the student's name, register number, date of birth, gender, and marks for five subjects. It should then calculate the total marks, percentage, and determine if the student has passed or failed, displaying all the information in a formatted result.

Program code:

```python
# Collect student details
def details():
    s_name = input("Enter the student name: ")
    s_regno = input("Enter the student register number: ")
    stud_dob = input("Enter the date of birth (DD-MM-YYYY): ")

    lang1_mark = int(input("Enter the language 1 mark: "))
    lang2_mark = int(input("Enter the language 2 mark: "))
    maths_mark = int(input("Enter the maths mark: "))
    sci_mark = int(input("Enter the science mark: "))
    socsci_mark = int(input("Enter the social science mark: "))

    gender = input("Enter the gender (Male/Female): ")

    # Calculate total and percentage
    stud_total = lang1_mark + lang2_mark + maths_mark + sci_mark + socsci_mark
    stud_percent = (stud_total / 500) * 100

    # Determine pass or fail
    if lang1_mark < 35 or lang2_mark < 35 or maths_mark < 35 or sci_mark < 35 or socsci_mark < 35:
        final_result = "Fail"
    else:
        final_result = "Pass"

    # Display student details
    print("\nSSLC EXAMINATION RESULTS")
    print("STUDENT DETAILS:")
    print(f"Student Name:    {s_name}")
    print(f"Register Number: {s_regno}")
    print(f"Date of birth:   {stud_dob}")
    print(f"Language 1:      {lang1_mark}")
    print(f"Language 2:      {lang2_mark}")
    print(f"Maths:           {maths_mark}")
    print(f"Science:         {sci_mark}")
    print(f"Social Science:  {socsci_mark}")
    print(f"Total:           {stud_total}")
    print(f"Percentage:      {stud_percent:.2f}%")
    print(f"Final result:    {final_result}\n")

# Starting of the program
def start():
    while True:
        y = input("Do you want to enter student details? (yes/no): ")
        if y.lower() == "yes":
            details()
        else:
            break

print("STUDENT DETAILS:")
start()
```

Input and output:

```
STUDENT DETAILS:
Do you want to enter student details? (yes/no): yes
Enter the student name: Anand
Enter the student register number: 70332
Enter the date of birth (DD-MM-YYYY): 09-09-2003
Enter the language 1 mark: 87
Enter the language 2 mark: 76
Enter the maths mark: 65
Enter the science mark: 54
Enter the social science mark: 87
Enter the gender (Male/Female): Male

SSLC EXAMINATION RESULTS
STUDENT DETAILS:
Student Name:    Anand
Register Number: 70332
Date of birth:   09-09-2003
Language 1:      87
Language 2:      76
Maths:           65
Science:         54
Social Science:  87
Total:           369
Percentage:      73.80%
Final result:    Pass

Do you want to enter student details? (yes/no): yes
Enter the student name:
```

```
Do you want to enter student details? (yes/no): yes
Enter the student name: Priya789
Enter the student register number: 19701%^
Enter the date of birth (DD-MM-YYYY): 09-09-2003
Enter the language 1 mark: 76
Enter the language 2 mark: 44
Enter the maths mark: 23
Enter the science mark: 45
Enter the social science mark: 77
Enter the gender (Male/Female): Female

SSLC EXAMINATION RESULTS
STUDENT DETAILS:
Student Name:    Priya789
Register Number: 19701%^
Date of birth:   09-09-2003
Language 1:      76
Language 2:      44
Maths:           23
Science:         45
Social Science:  77
Total:           265
Percentage:      53.00%
Final result:    Fail

Do you want to enter student details? (yes/no): no
PS C:\Users\sh\Desktop\Patterns cognitive>
```

In order to avoid the numbers and special characters in the respective input, adding validations to avoid inappropriate input.

Date: 03/07/2024 ,04/07/2024 and 05/07/2024

Day: Wednesday , Thursday and Friday

Task: Learn about string functions and regular expressions in Python and modify the given Python program to include input validations. Develop a Python program to capture and validate student details, including name, register number, date of birth, gender, and marks for various subjects. The program should ensure the validity of each input, calculate the total and percentage of marks, and determine the final result (pass/fail) based on individual subject marks. The program should display the student's examination results and prompt the user to enter details for multiple students until the user decides to stop.

Program code:

```
from datetime import datetime
import re
def details():

    #student name
    def name():
        stud_name=input("Enter the name again:")
        return stud_name
    s_name=input("Enter the student name:")
    while not re.match(r'^[a-zA-Z\s]+$', s_name):
            if not s_name.isalnum():
                print("You have entered some special characters. please provide name in alphabets")
                s_name=name()
            else:
                if not s_name.replace(' ', '').isalpha():
                    print("You have entered numbers.please provide name in alphabet")
                s_name=name()
    #print("The student name is",s_name)

    #student register no
    def regno():
        stud_regno=input("Enter the register no(only 6 digits):")
        return stud_regno
    s_regno=input("Enter the student register no:")
    while not s_regno.isdigit():
        if s_regno.isalpha():
            print("You have entered alphabets.Enter digits" )
            s_regno=regno()
        elif s_regno.isalnum():
            print("You have entered combination of alphabets and numbers.Enter digits")
            s_regno=regno()
        else:
            print("You have entered special characters.Enter digits")
            s_regno=regno()
    while (len(s_regno)!=6):
            print("The register number should contains 6 digits")
            s_regno=regno()
    #print("The student register number:",s_regno)

    #student date of birth
    while True:
        stud_dob=input("Enter the date of birth:")
        format_of_dob=r'^\d{2}-\d{2}-\d{4}$'
        if re.match(format_of_dob,stud_dob):
            #print("Date of birth:",stud_dob)
            break
        else:
            print("Invalid format of date of birth")
```

```
    #language 1 marks
    def l1_mark():
        l1=int(input("Enter the language 1 mark (0-100):"))
        return l1
    lang1_mark=int(input("Enter the language 1 mark:"))
    while lang1_mark<0 or lang1_mark>100:
        print("Invalid input of marks")
        lang1_mark=l1_mark()

    #language 2 marks
    def l2_mark():
        l2=int(input("Enter the language 2 mark (0-100):"))
        return l2
    lang2_mark=int(input("Enter the language 2 mark:"))
    while lang2_mark<0 or lang2_mark>100:
        print("Invalid input of marks")
        lang2_mark=l2_mark()

    #maths mark
    def m_mark():
        m=int(input("Enter the maths mark (0-100):"))
        return m
    maths_mark=int(input("Enter the maths mark:"))
    while maths_mark<0 or maths_mark>100:
        print("Invalid input of marks")
        maths_mark=m_mark()

    #science mark
    def s_mark():
        s=int(input("Enter the science mark (0-100):"))
        return s
    sci_mark=int(input("Enter the science mark:"))
    while sci_mark<0 or sci_mark>100:
        print("Invalid input of marks")
        sci_mark=s_mark()

    #social science mark
    def ss_mark():
        ss=int(input("Enter the social science mark (0-100):"))
        return ss
    socsci_mark=int(input("Enter the social science mark:"))
    while socsci_mark<0 or socsci_mark>100:
        print("Invalid input of marks")
        socsci_mark=ss_mark()

    #student gender
    gender=input("Enter the gender (Male/Female):")
    #print("The gender of the student is:",gender)
```

```
    stud_total=0
    stud_total=lang1_mark+lang2_mark+maths_mark+sci_mark+socsci_mark

    stud_percent=(stud_total/500)*100
    #pass or fail
    final_result=" "

    if lang1_mark<35 or lang2_mark<35 or maths_mark<35 or sci_mark<35 or socsci_mark<35:
        final_result="Fail"
    else:
        final_result="Pass"

    print("SSLC EXAMINATION RESULTS")
    print("STUDENT DETAILS:")
    print("Student Name:    ",s_name)
    print("Register Number:",s_regno)
    print("Date of birth:   ",stud_dob)
    print("Language 1:    ",lang1_mark)
    print("Language 2:      ",lang2_mark)
    print("Maths:           ",maths_mark)
    print("Science:         ",sci_mark)
    print("Social Science:  ",socsci_mark)
    print("Total:           ",stud_total)
    print("Percentage:      ",stud_percent)
    print("Final result:    ",final_result)

#starting of the program
def start():
    while True:
        y = input("Do you want to enter student details? (yes/no): ")
        if y == "yes":
            details()
        else:
            break

print("STUDENT DETAILS:")
start()
```

Input and output:

```
STUDENT DETAILS:
Do you want to enter student details? (yes/no): yes
Enter the student name:Arun
Enter the student register no:123001
Enter the date of birth:09/09/2003
Invalid format of date of birth
Enter the date of birth:09-09-2003
Enter the language 1 mark:-70
Invalid input of marks
Enter the language 1 mark (0-100):90
Enter the language 2 mark:89
Enter the maths mark:78
Enter the science mark:67
Enter the social science mark:89
Enter the gender (Male/Female):Male
SSLC EXAMINATION RESULTS
STUDENT DETAILS:
Student Name:    Arun
Register Number: 123001
Date of birth:   09-09-2003
Language 1:    90
Language 2:      89
Maths:           78
Science:         67
Social Science:  89
Total:           413
Percentage:      82.6
Final result:    Pass
Do you want to enter student details? (yes/no): yes
Enter the student name:
```

```
Final result:    Pass
Do you want to enter student details? (yes/no): yes
Enter the student name:Balaji371-
You have entered some special characters. please provide name in alphabets
Enter the name again:#$%^&*
You have entered some special characters. please provide name in alphabets
Enter the name again:234567890
You have entered numbers.please provide name in alphabet
Enter the name again:Bala Sundar
Enter the student register no:109174
Enter the date of birth:18-09-2003
Enter the language 1 mark:67
Enter the language 2 mark:34
Enter the maths mark:56
Enter the science mark:78
Enter the social science mark:23
Enter the gender (Male/Female):Male
SSLC EXAMINATION RESULTS
STUDENT DETAILS:
Student Name:    Bala Sundar
Register Number: 109174
Date of birth:   18-09-2003
Language 1:    67
Language 2:      34
Maths:           56
Science:         78
Social Science:  23
Total:           258
Percentage:      51.6
Final result:    Fail
Do you want to enter student details? (yes/no):
```

```
Do you want to enter student details? (yes/no): yes
Enter the student name:Priya
Enter the student register no:jwsp
You have entered alphabets.Enter digits
Enter the register no(only 6 digits):kjw 81984
You have entered special characters.Enter digits
Enter the register no(only 6 digits):8209874630
The register number should contains 6 digits
Enter the register no(only 6 digits):123456
Enter the date of birth:04-05-2003
Enter the language 1 mark:23
Enter the language 2 mark:45
Enter the maths mark:67
Enter the science mark:78
Enter the social science mark:89
Enter the gender (Male/Female):Female
SSLC EXAMINATION RESULTS
STUDENT DETAILS:
Student Name:    Priya
Register Number: 123456
Date of birth:   04-05-2003
Language 1:    23
Language 2:      45
Maths:           67
Science:         78
Social Science:  89
Total:           302
Percentage:      60.4
Final result:    Fail
Do you want to enter student details? (yes/no): no
PS C:\Users\sh\Desktop\Patterns cognitive>
```

Date: 08/07/2024 **Day: Monday**

Task : Learn about exception handling in Python using the try and except methods. Modify the given Python program to include input validations using only try and except blocks. The program should prompt the user to enter student details, including the student's name, register number, date of birth, gender, and marks for five subjects. It should validate the inputs to ensure they meet specified criteria, handle any invalid inputs or errors using try and except, and then calculate the total marks, percentage, and determine if the student has passed or failed. Display all the information in a formatted result.

Program code:

```
import re
def details():

    def name():
        while True:
            try:
                s_name = input("Enter the student name: ")
                if not re.match(r'^[a-zA-Z\s]+$', s_name):
                    if not s_name.isalnum():
                        raise ValueError("You have entered some special characters. Please provide the
name in alphabets.")
                    elif not s_name.replace(' ', '').isalpha():
                        raise ValueError("You have entered numbers. Please provide the name in
alphabets.")
                return s_name
            except ValueError as ve:
                print(ve)
                #s_name = name()

    stud_name =name()
    #print(student_name)

    def regno():
        while True:
            try:
                s_regno=input("Enter the student register no:")
                while not s_regno.isdigit():
                    if s_regno.isalpha():
                        raise ValueError("You have entered alphabets.Enter digits" )
                    elif s_regno.isalnum():
                        raise ValueError("You have entered combination of alphabets and numbers.Enter
digits")
                    else:
                        raise ValueError("You have entered special characters.Enter digits")
                if (len(s_regno)!=6):
                    print("The register number should contains 6 digits")
                    s_regno=regno()
                return s_regno
            except ValueError as ve:
                print(ve)
    stud_regno=regno()
    #print(stud_regno)


    while True:
        try:
            stud_dob=input("Enter the date of birth:")
            format_of_dob=r'^\d{2}-\d{2}-\d{4}$'
            if re.match(format_of_dob,stud_dob):
                #print("Date of birth:",stud_dob)
                break
            else:
                raise ValueError("Invalid format of date of birth")
        except ValueError as ve:
            print(ve)
    #print(stud_dob)

    while True:
        try:
            lang1_mark=int(input("Enter the language 1 mark:"))
            if lang1_mark<0 or lang1_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)

    #print(lang1_mark)
    while True:
        try:
            lang2_mark=int(input("Enter the language 2 mark:"))
            if lang2_mark<0 or lang2_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)
```

```
    while True:
        try:
            maths_mark=int(input("Enter the maths mark:"))
            if maths_mark<0 or maths_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)

    while True:
        try:
            sci_mark=int(input("Enter the science mark:"))
            if sci_mark<0 or sci_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)

    while True:
        try:
            socsci_mark=int(input("Enter the social science mark:"))
            if socsci_mark<0 or socsci_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)

    gender=input("Enter the gender (Male/Female):")
    #print("The gender of the student is:",gender)

    stud_total=0
    stud_total=lang1_mark+lang2_mark+maths_mark+sci_mark+socsci_mark


    stud_percent=(stud_total/500)*100
    #pass or fail
    final_result=" "

    try:
        if lang1_mark<35 or lang2_mark<35 or maths_mark<35 or sci_mark<35 or socsci_mark<35:
            final_result="Fail"
        else:
            final_result="Pass"
    except ValueError as ve:
        print(ve)

    print("SSLC EXAMINATION RESULTS")
    print("STUDENT DETAILS:")
    print("Student Name:   ",stud_name)
    print("Register Number:",stud_regno)
    print("Date of birth:  ",stud_dob)
    print("\nLanguage 1:   ",lang1_mark)
    print("Language 2:     ",lang2_mark)
    print("Maths:          ",maths_mark)
    print("Science:        ",sci_mark)
    print("Social Science: ",socsci_mark)
    print("Total:          ",stud_total)
    print("Percentage:     ",stud_percent)
    print("Final result:   ",final_result)


while True:
    y=input("Do you want to enter the student details (yes/no)?")
    if y.lower()=="yes":
        details()
    elif y.lower()=="no":
        break
    else:
        print("Invalid input")
```

Input and output:

```
py
Do you want to enter the student details (yes/no)?yes
Enter the student name: Hema123
You have entered numbers. Please provide the name in alphabets.
Enter the student name: Hema@#$%^&
You have entered some special characters. Please provide the name in alphabets.
Enter the student name: Hema
Enter the student register no:123001
Enter the date of birth:08-07-2003
Enter the language 1 mark:87
Enter the language 2 mark:76
Enter the maths mark:93
Enter the science mark:98
Enter the social science mark:88
Enter the gender (Male/Female):Female
SSLC EXAMINATION RESULTS
STUDENT DETAILS:
Student Name:    Hema
Register Number: 123001
Date of birth:   08-07-2003

Language 1:    87
Language 2:      76
Maths:           93
Science:         98
Social Science:  88
Total:           442
Percentage:      88.4
Final result:    Pass
Do you want to enter the student details (yes/no)?yes
```

```
Do you want to enter the student details (yes/no)?yes
Enter the student name: Girish
Enter the student register no:onouw9170
You have entered combination of alphabets and numbers.Enter digits
Enter the student register no:98#$%
You have entered special characters.Enter digits
Enter the student register no:93013714
The register number should contains 6 digits
Enter the student register no:123002
Enter the date of birth:18/09/2003
Invalid format of date of birth
Enter the date of birth:07-03-2003
Enter the language 1 mark:-90
Marks should be within 0-100
Enter the language 1 mark:90
Enter the language 2 mark:67
Enter the maths mark:32
Enter the science mark:65
Enter the social science mark:89
Enter the gender (Male/Female):Male
SSLC EXAMINATION RESULTS
STUDENT DETAILS:
Student Name:    Girish
Register Number: 123002
Date of birth:   07-03-2003

Language 1:    90
Language 2:      67
Maths:           32
Science:         65
Social Science:  89
Total:           343
Percentage:      68.60000000000001
Final result:    Fail
Do you want to enter the student details (yes/no)?no
PS C:\Users\sh\Desktop\Patterns cognitive>
```

Date: 09/07/2024 **Day:Tuesday**

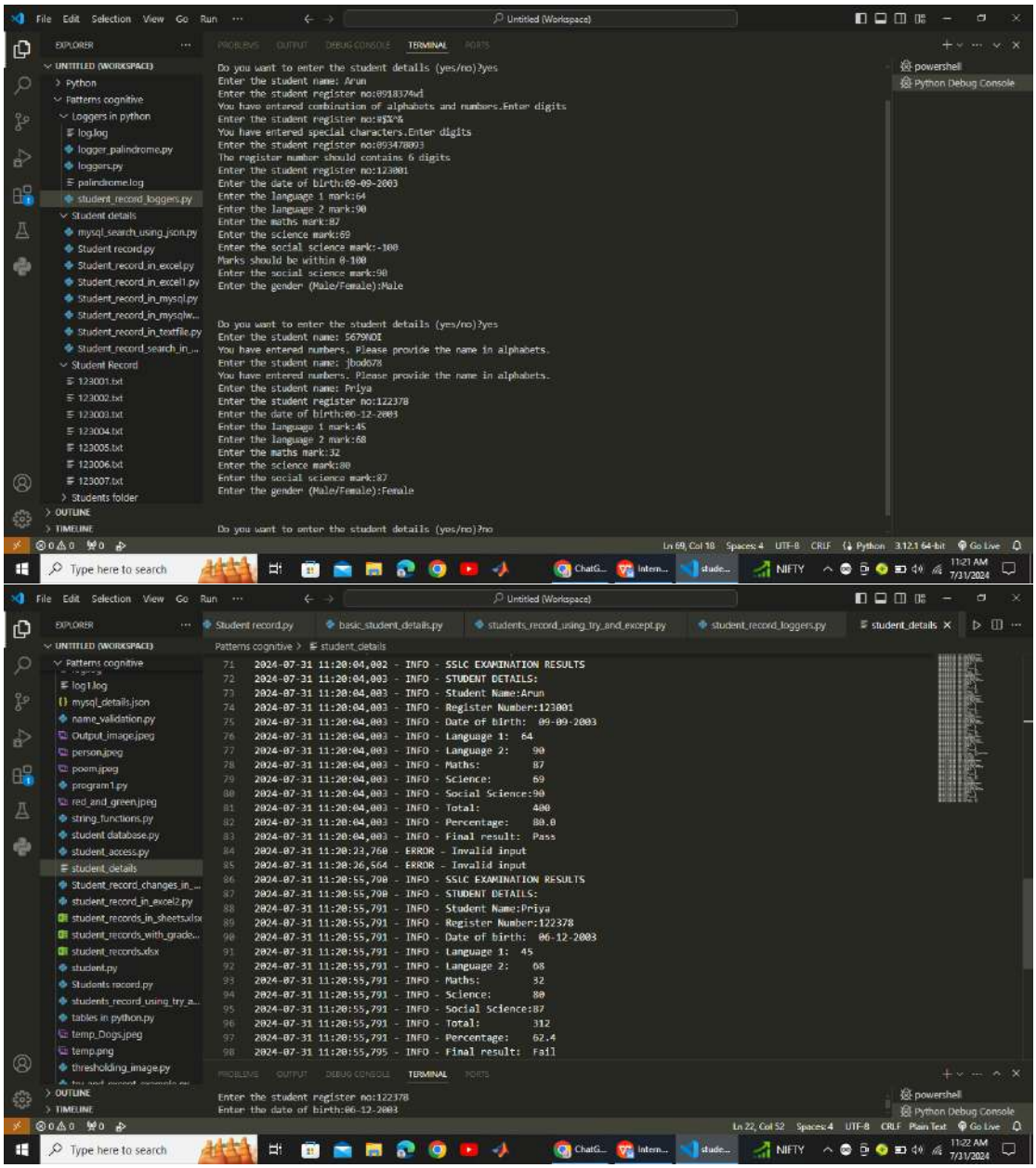
Task: Develop a Python program to capture and validate student details, including name, register number, date of birth, gender, and marks in various subjects. The program should ensure the inputs are correctly formatted and within acceptable ranges, compute the total and percentage marks, determine the final result (pass or fail), and log all details, including errors, in a log file using the logging module. The program should repeatedly prompt the user to enter details for multiple students until the user chooses to stop.

Program code:

```
import re
import logging


logging.basicConfig(level=logging.INFO ,filename="student_details",filemode="a",
                    format=("%(asctime)s - %(levelname)s - %(message)s"))

def details():

    def name():
        while True:
            try:
                s_name = input("Enter the student name: ")
                if not re.match(r'^[a-zA-Z\s]+$', s_name):
                    if not s_name.isalnum():
                        raise ValueError("You have entered some special characters. Please provide the
name in alphabets.")
                    elif not s_name.replace(' ', '').isalpha():
                        raise ValueError("You have entered numbers. Please provide the name in
alphabets.")
                return s_name
            except ValueError as ve:
                print(ve)
                logging.error("Invalid input")
                #s_name = name()

    stud_name =name()
    #print(student_name)

    def regno():
        while True:
            try:
                s_regno=input("Enter the student register no:")
                while not s_regno.isdigit():
                    if s_regno.isalpha():
                        raise ValueError("You have entered alphabets.Enter digits" )
                    elif s_regno.isalnum():
                        raise ValueError("You have entered combination of alphabets and numbers.Enter
digits")
                    else:
                        raise ValueError("You have entered special characters.Enter digits")
                if (len(s_regno)!=6):
                    print("The register number should contains 6 digits")
                    s_regno=regno()
                return s_regno
            except ValueError as ve:
                print(ve)
                logging.error("Invalid input")
    stud_regno=regno()
    #print(stud_regno)


    while True:
        try:
            stud_dob=input("Enter the date of birth:")
            format_of_dob=r'^\d{2}-\d{2}-\d{4}$'
            if re.match(format_of_dob,stud_dob):
                #print("Date of birth:",stud_dob)
                break
            else:
                raise ValueError("Invalid format of date of birth")
        except ValueError as ve:
            print(ve)
            logging.error("Invalid input")
    #print(stud_dob)
```

```
    while True:
        try:
            lang1_mark=int(input("Enter the language 1 mark:"))
            if lang1_mark<0 or lang1_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)

    #print(lang1_mark)
    while True:
        try:
            lang2_mark=int(input("Enter the language 2 mark:"))
            if lang2_mark<0 or lang2_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)
            logging.error("Invalid input")

    while True:
        try:
            maths_mark=int(input("Enter the maths mark:"))
            if maths_mark<0 or maths_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)
            logging.error("Invalid input")

    while True:
        try:
            sci_mark=int(input("Enter the science mark:"))
            if sci_mark<0 or sci_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)
            logging.error("Invalid input")

    while True:
        try:
            socsci_mark=int(input("Enter the social science mark:"))
            if socsci_mark<0 or socsci_mark>100:
                raise ValueError("Marks should be within 0-100")
            else:
                break
        except ValueError as ve:
            print(ve)
            logging.error("Invalid input")

    gender=input("Enter the gender (Male/Female):")
    #print("The gender of the student is:",gender)

    stud_total=0
    stud_total=lang1_mark+lang2_mark+maths_mark+sci_mark+socsci_mark


    stud_percent=(stud_total/500)*100
    #pass or fail
    final_result=" "

    try:
        if lang1_mark<35 or lang2_mark<35 or maths_mark<35 or sci_mark<35 or socsci_mark<35:
            final_result="Fail"
        else:
            final_result="Pass"
    except ValueError as ve:
        print(ve)
        logging.warning("Something went wrong")
    '''
    print("SSLC EXAMINATION RESULTS")
    print("STUDENT DETAILS:")
    print("Student Name:    ",stud_name)
    print("Register Number:",stud_regno)
    print("Date of birth:   ",stud_dob)
    print("\nLanguage 1:    ",lang1_mark)
    print("Language 2:      ",lang2_mark)
    print("Maths:           ",maths_mark)
    print("Science:         ",sci_mark)
    print("Social Science: ",socsci_mark)
    print("Total:           ",stud_total)
    print("Percentage:      ",stud_percent)
    print("Final result:    ",final_result)
    '''
    print("\n")
    logging.info("SSLC EXAMINATION RESULTS")
    logging.info("STUDENT DETAILS:")
    logging.info("Student Name:%s    ",stud_name)
    logging.info("Register Number:%s",stud_regno)
    logging.info("Date of birth:  %s",stud_dob)
    logging.info("Language 1:   %s ",lang1_mark)
    logging.info("Language 2:     %s ",lang2_mark)
    logging.info("Maths:          %s ",maths_mark)
    logging.info("Science:        %s ",sci_mark)
    logging.info("Social Science:%s ",socsci_mark)
    logging.info("Total:          %s ",stud_total)
    logging.info("Percentage:     %s ",stud_percent)
    logging.info("Final result:   %s ",final_result)


while True:
    y=input("Do you want to enter the student details (yes/no)?")
    if y.lower()=="yes":
        details()
    elif y.lower()=="no":
        break
    else:
        logging.error("Invalid input")
```

Input and output:



Date:10/07/2024 and 11/07/2024
Day:Wednesday and Thursday

Task: Develop a Python program to capture and validate student details, including their name, register number, date of birth, marks in five subjects, and gender. The program should ensure all inputs meet specific criteria, calculate the total marks and percentage, determine the pass or fail status, and optionally save the results in a text file named with the student's register number, opening it in Notepad on a Windows system.

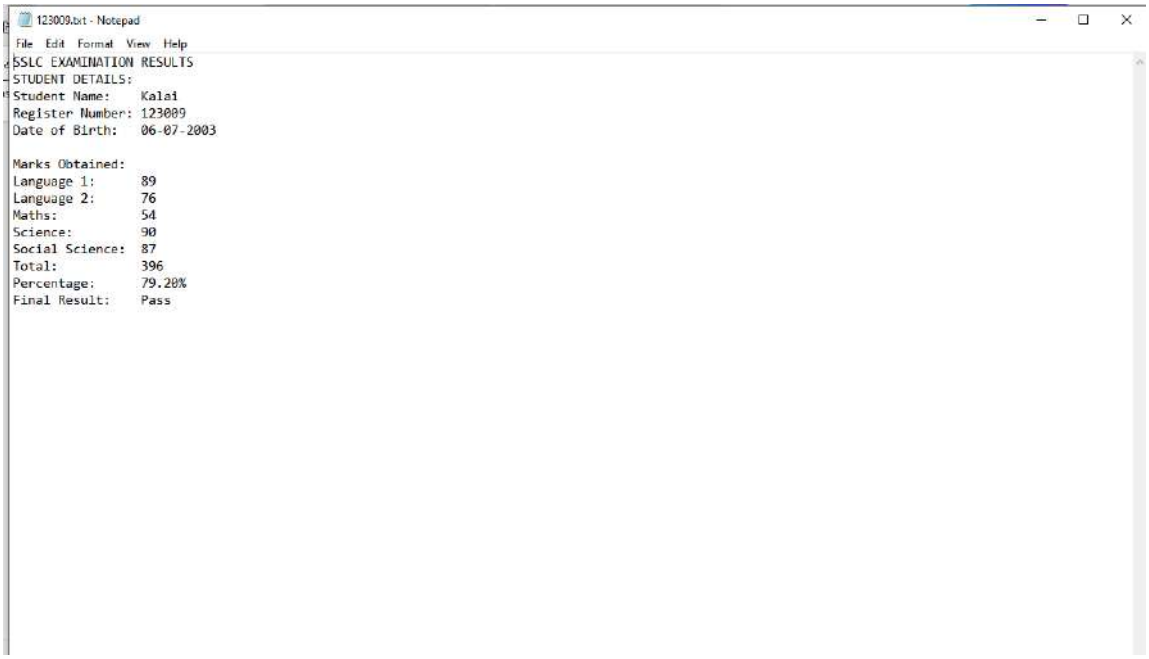
Program code:

```
from datetime import datetime
import re
import os
def details():

    #student name
    def name():
        stud_name=input("Enter the name again:")
        return stud_name
    s_name=input("Enter the student name:")
    while not re.match(r'^[a-zA-Z\s]+$', s_name):
            if not s_name.isalnum():
                print("You have entered some special characters. please provide name in alphabets")
                s_name=name()
            else:
                if not s_name.replace(' ', '').isalpha():
                    print("You have entered numbers.please provide name in alphabet")
                s_name=name()
    #print("The student name is",s_name)

    #student register no
    def regno():
        stud_regno=input("Enter the register no(only 6 digits):")
        return stud_regno
    s_regno=input("Enter the student register no:")
    while not s_regno.isdigit():
        if s_regno.isalpha():
            print("You have entered alphabets.Enter digits" )
            s_regno=regno()
        elif s_regno.isalnum():
            print("You have entered combination of alphabets and numbers.Enter digits")
            s_regno=regno()
        else:
            print("You have entered special characters.Enter digits")
            s_regno=regno()
    while (len(s_regno)!=6):
            print("The register number should contains 6 digits")
            s_regno=regno()
    #print("The student register number:",s_regno)

    #student date of birth
    while True:
        stud_dob=input("Enter the date of birth:")
        format_of_dob=r'^\d{2}-\d{2}-\d{4}$'
        if re.match(format_of_dob,stud_dob):
            #print("Date of birth:",stud_dob)
            break
        else:
            print("Invalid format of date of birth")

    #language 1 marks
    def l1_mark():
        l1=int(input("Enter the language 1 mark (0-100):"))
        return l1
    lang1_mark=int(input("Enter the language 1 mark:"))
    while lang1_mark<0 or lang1_mark>100:
        print("Invalid input of marks")
        lang1_mark=l1_mark()
    #print("Language 1:",lang1_mark)

    #language 2 marks
    def l2_mark():
        l2=int(input("Enter the language 2 mark (0-100):"))
        return l2
    lang2_mark=int(input("Enter the language 2 mark:"))
    while lang2_mark<0 or lang2_mark>100:
        print("Invalid input of marks")
        lang2_mark=l2_mark()
    #print("Language 2:",lang2_mark)

    #maths mark
    def m_mark():
        m=int(input("Enter the maths mark (0-100):"))
        return m
    maths_mark=int(input("Enter the maths mark:"))
    while maths_mark<0 or maths_mark>100:
        print("Invalid input of marks")
        maths_mark=m_mark()
    #print("Maths:",maths_mark)
```

```
    #science mark
    def s_mark():
        s=int(input("Enter the science mark (0-100):"))
        return s
    sci_mark=int(input("Enter the science mark:"))
    while sci_mark<0 or sci_mark>100:
        print("Invalid input of marks")
        sci_mark=s_mark()
    #print("Science:",sci_mark)

    #social science mark
    def ss_mark():
        ss=int(input("Enter the social science mark (0-100):"))
        return ss
    socsci_mark=int(input("Enter the social science mark:"))
    while socsci_mark<0 or socsci_mark>100:
        print("Invalid input of marks")
        socsci_mark=ss_mark()
    #print("Social Science:",socsci_mark)

    #student gender
    gender=input("Enter the gender (Male/Female):")
    #print("The gender of the student is:",gender)

    stud_total=0
    stud_total=lang1_mark+lang2_mark+maths_mark+sci_mark+socsci_mark


    stud_percent=(stud_total/500)*100
    #pass or fail
    final_result=" "

    if lang1_mark<35 or lang2_mark<35 or maths_mark<35 or sci_mark<35 or socsci_mark<35:
        final_result="Fail"
    else:
        final_result="Pass"

    print("SSLC EXAMINATION RESULTS")
    print("STUDENT DETAILS:")
    print("Student Name:    ",s_name)
    print("Register Number:",s_regno)
    print("Date of birth:   ",stud_dob)
    print("\nLanguage 1:    ",lang1_mark)
    print("Language 2:      ",lang2_mark)
    print("Maths:           ",maths_mark)
    print("Science:         ",sci_mark)
    print("Social Science: ",socsci_mark)
    print("Total:           ",stud_total)
    print("Percentage:      ",stud_percent)
    print("Final result:    ",final_result)
    p=input("Do you want to print? (Yes/No)")
    if (p=="Yes"):
        if not os.path.exists("Student Record"):
            os.mkdir("Student Record")
        file=os.path.join("Student Record",f"{s_regno}.txt")
        with open (file,"w") as f:
            f.write("SSLC EXAMINATION RESULTS\n")
            f.write("STUDENT DETAILS:\n")
            f.write(f"Student Name:    {s_name}\n")
            f.write(f"Register Number: {s_regno}\n")
            f.write(f"Date of Birth:   {stud_dob}\n")
            f.write("\nMarks Obtained:\n")
            f.write(f"Language 1:      {lang1_mark}\n")
            f.write(f"Language 2:      {lang2_mark}\n")
            f.write(f"Maths:           {maths_mark}\n")
            f.write(f"Science:         {sci_mark}\n")
            f.write(f"Social Science:  {socsci_mark}\n")
            f.write(f"Total:           {stud_total}\n")
            f.write(f"Percentage:      {stud_percent:.2f}%\n")
            f.write(f"Final Result:    {final_result}\n")
        os.system(f"notepad.exe {file}")

#starting of the program
def start():
    while True:
        y = input("Do you want to enter student details? (yes/no): ")
        if y == "yes":
            details()
        else:
            break

print("STUDENT DETAILS:")
start()
```

Input and output:

```
STUDENT DETAILS:
Do you want to enter student details? (yes/no): yes
Enter the student name:Kalai
Enter the student register no:123009
Enter the date of birth:06-07-2003
Enter the language 1 mark:89
Enter the language 2 mark:76
Enter the maths mark:54
Enter the science mark:90
Enter the social science mark:87
Enter the gender (Male/Female):Female
SSLC EXAMINATION RESULTS
STUDENT DETAILS:
Student Name:    Kalai
Register Number: 123009
Date of birth:   06-07-2003

Language 1:    89
Language 2:      76
Maths:           54
Science:         90
Social Science:  87
Total:           396
Percentage:      79.2
Final result:    Pass
Do you want to print? (Yes/No)yes
```



Date:12/07/2024 and 15/07/2024
Day:Friday and Monday

Task: Create a Python program to manage student records, including entering new details and searching for existing records stored in an Excel file. The program should validate inputs for student information such as name, register number, date of birth, gender, and marks in five subjects. It should calculate the total marks, percentage, and determine the final result (pass or fail), assigning grades based on the marks. The records should be saved to an Excel file with bolded column headers, and the final result should be highlighted if it is "Fail". The program should also

provide an option to display and print student details, formatted in a tabular format and also to search for student details in an Excel file using their register number and date of birth. The program should display the student's information if found, including name, gender, and marks in five subjects, and calculate the total marks and percentage. It should also save these details in a text file named with the student's register number, open it in Notepad on a Windows system

Program code:

```
from datetime import datetime
import re
import os
from openpyxl import Workbook, load_workbook
from openpyxl.styles import Font,PatternFill
from tabulate import tabulate

def details():
    # Function to get and validate student name
    #student name
    try:
        s_name=input("Enter the student name:")
        while not re.match(r'^[a-zA-Z\s]+$', s_name):
                if not s_name.isalnum():
                    print("You have entered some special characters. please provide name in alphabets")
                    s_name=input("Enter the student name:")
                else:
                    if not s_name.replace(' ', '').isalpha():
                        print("You have entered numbers.please provide name in alphabet")
                    s_name=input("Enter the student name:")
    except Exception as e:
        print(f"An error occured {e}")

    #student register no
    try:
        s_regno=input("\nEnter your register number:")
        while not s_regno.isdigit():
            if s_regno.isalpha():
                print("You have entered alphabets.Please enter only digits")
                s_regno=input("\nEnter your register number:")
            elif s_regno.isalnum():
                print("You have entered the combination of letters and digits.Please enter only
digits")
                s_regno=input("\nEnter your register number:")
            else:
                print("You have entered special characters.Please enter only digits")
                s_regno=input("\nEnter your register number:")
        while len(s_regno)!=6:
            print("Reguster number should contain 6 digits only.")
            s_regno=input("\nEnter your register number:")
    except Exception as e:
        print(f"An error occured {e}")

    #student date of birth
    try:
        while True:
            stud_dob=input("Enter the date of birth:")
            format_of_dob=r'^\d{2}-\d{2}-\d{4}$'
            if re.match(format_of_dob,stud_dob):
                #print("Date of birth:",stud_dob)
                break
            else:
                print("Invalid format of date of birth")
    except Exception as e:
        print(f"An error occured {e}")

    gender=input("Enter the gender (Male/Female):")
    #print("The gender of the student is:",gender)
    # marks array
    stud_marks=[]
    for i in range(5):
        if i==0:
            #language 1 mark
            while True:
                try:
                    lang1_mark=int(input("Enter the language 1 mark:"))
                    if lang1_mark<0 or lang1_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(lang1_mark)
                        break
                except ValueError as ve:
                    print(ve)
        elif i==1:
            #language 2 mark
            while True:
                try:
                    lang2_mark=int(input("Enter the language 2 mark:"))
                    if lang2_mark<0 or lang2_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(lang2_mark)
                        break
                except ValueError as ve:
                    print(ve)
        elif i==2:
            #maths mark
            while True:
                try:
                    maths_mark=int(input("Enter the maths mark:"))
                    if maths_mark<0 or maths_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(maths_mark)
                        break
                except ValueError as ve:
                    print(ve)
```

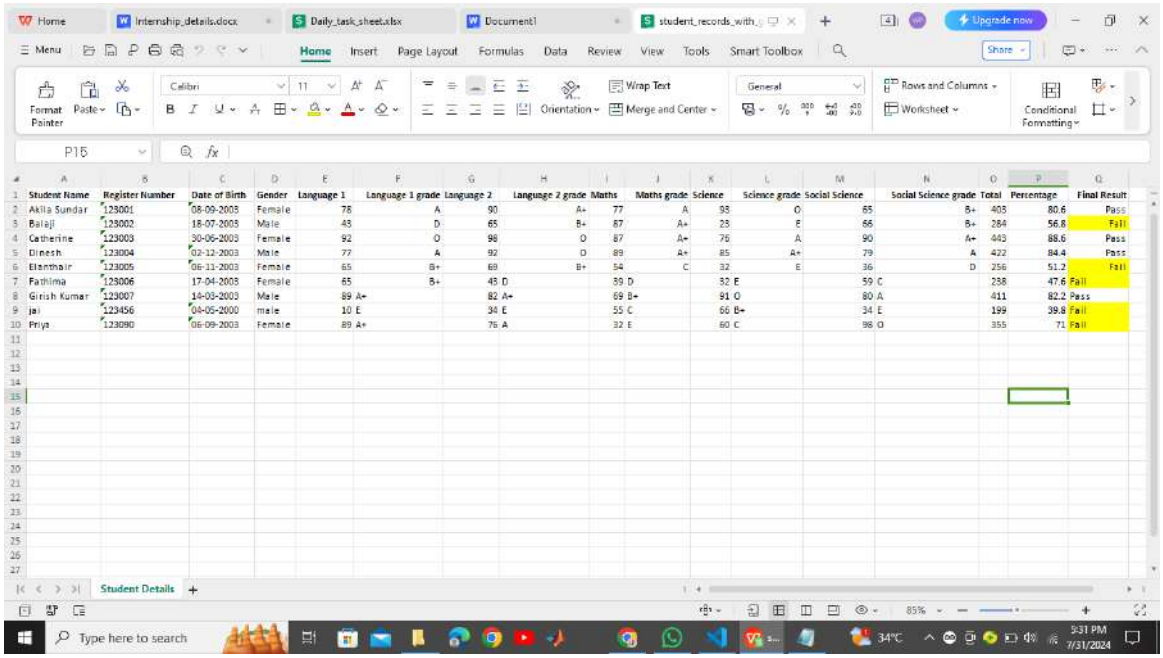
```
        elif i==3:
            #science mark
            while True:
                try:
                    sci_mark=int(input("Enter the science mark:"))
                    if sci_mark<0 or sci_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(sci_mark)
                        break
                except ValueError as ve:
                    print(ve)
        elif i==4:
            #social science mark
            while True:
                try:
                    socsci_mark=int(input("Enter the social science mark:"))
                    if socsci_mark<0 or socsci_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(socsci_mark)
                        break
                except ValueError as ve:
                    print(ve)
    #print(stud_marks)

    grades=[]
    for i in stud_marks:
        if 90<i<=100:
            grades.append("O")
        elif 80<i<=90:
            grades.append("A+")
        elif 70<i<=80:
            grades.append("A")
        elif 60<i<=70:
            grades.append("B+")
        elif 50<i<=60:
            grades.append("C")
        elif 35<i<=50:
            grades.append("D")
        else:
            grades.append("E")

    #print(grades)

    stud_total=0
    stud_total=lang1_mark+lang2_mark+maths_mark+sci_mark+socsci_mark

    stud_percent=(stud_total/500)*100
    #pass or fail

    try:
        if lang1_mark<35 or lang2_mark<35 or maths_mark<35 or sci_mark<35 or socsci_mark<35:
            final_result="Fail"
        else:
            final_result="Pass"
    except ValueError as ve:
        print(ve)
    save_to_excel(s_name, s_regno, stud_dob, gender, stud_marks,grades, stud_total, stud_percent,
final_result)

def save_to_excel(s_name, s_regno, stud_dob, gender, stud_marks, grades, stud_total, stud_percent,
final_result):


    file_path = "student_records_with_grades.xlsx"
    if os.path.exists(file_path):
        wb = load_workbook(file_path)
        sheet = wb.active
    else:
        wb = Workbook()
        sheet = wb.active
        sheet.title = "Student Details"
        sheet.append(["Student Name", "Register Number", "Date of Birth", "Gender", "Language 1",
"Language 1 grade", "Language 2", "Language 2 grade", "Maths", "Maths grade" ,"Science" , "Science
grade" , "Social Science" , "Social Science grade" , "Total", "Percentage", "Final Result"])
        #bold the column headexit
        #for cell in sheet["1:1"]:
            #cell.font=Font(bold=True)
    for cell in sheet["1:1"]:
        cell.font=Font(bold=True)


    sheet.append([s_name, s_regno, stud_dob, gender, stud_marks[0], grades[0], stud_marks[1],
grades[1], stud_marks[2], grades[2], stud_marks[3], grades[3], stud_marks[4], grades[4], stud_total,
stud_percent, final_result])
    for row in sheet.iter_rows(min_row=2,max_row=sheet.max_row,min_col=17,max_col=17):
        for cell in row:
            if cell.value=="Fail":
                cell.fill=PatternFill(start_color="FFFF00",end_color="FFFF00",fill_type="solid")

    wb.save(file_path)
    print(f"Details have been saved to {file_path}")

def get_details():

    try:
        s_regno=input("\nEnter your register number:")
        while not s_regno.isdigit():
            if s_regno.isalpha():
                print("You have entered alphabets.Please enter only digits")
                s_regno=input("\nEnter your register number:")
            elif s_regno.isalnum():
                print("You have entered the combination of letters and digits.Please enter only
digits")
                s_regno=input("\nEnter your register number:")
            else:
                print("You have entered special characters.Please enter only digits")
                s_regno=input("\nEnter your register number:")
        while len(s_regno)!=6:
            print("Reguster number should contain 6 digits only.")
            s_regno=input("\nEnter your register number:")
    except Exception as e:
        print(f"An error occured {e}")
```

```
    try:
        while True:
            stud_dob=input("Enter the date of birth:")
            format_of_dob=r'^\d{2}-\d{2}-\d{4}$'
            if re.match(format_of_dob,stud_dob):
                #print("Date of birth:",stud_dob)
                break
            else:
                print("Invalid format of date of birth")
    except Exception as e:
        print(f"An error occured {e}")

    file = "student_records_with_grades.xlsx"
    student_found = False

    if os.path.exists(file):
        wb = load_workbook(file)
        sheet = wb.active
        for row in sheet.iter_rows(values_only=True):
            if row[1] == s_regno and row[2] == stud_dob :
                print("SSLC EXAMINATION:")
                print("Student details:")
                print("Student Name:    ", row[0])
                print("Register Number: ", row[1])
                print("Date of Birth:   ", row[2])
                print("Gender:          ", row[3])
                table_details=[
                    ["Language 1",row[4],row[5]],
                    ["Language 2",row[6],row[7]],
                    ["Maths",row[8],row[9]],
                    ["Science",row[10],row[11]],
                    ["Social Science",row[12],row[13]],
                    ["Total",row[14],""],
                    ["Percentage",row[15],""],
                    ["Result",row[16],""],
                ]
                print("\n Marks and grades")
                print(tabulate(table_details,headers=["Subjects","Marks","Grades"],tablefmt="grid"))

                student_found = True
                p=input("Do you want to print(yes/no)? ")
                if p=="yes":
                    print_details(row)
                break
            #print("The student details are not found")
    if not student_found:
        print("No student details are found.")


def print_details(row):
    filename="Students folder"
    if not os.path.exists(filename):
        os.makedirs(filename)

    file=f"{row[1]}.txt"
    filepath=os.path.join(filename,file)

    with open (filepath,"w") as f:
                f.write("SSLC EXAMINATION:")
                f.write("\nStudent details:")
                f.write(f"\nStudent Name:     {row[0]}")
                f.write(f"\nRegister Number:  {row[1]}")
                f.write(f"\nDate of Birth:    {row[2]}")
                f.write(f"\nGender:           {row[3]}")
                table_details=[
                    ["Language 1",row[4],row[5]],
                    ["Language 2",row[6],row[7]],
                    ["Maths",row[8],row[9]],
                    ["Science",row[10],row[11]],
                    ["Social Science",row[12],row[13]],
                    ["Total",row[14],""],
                    ["Percentage",row[15],""],
                    ["Result",row[16],""],
                ]
                f.write("\n Marks and grades")
                f.write(tabulate(table_details,headers=["Subjects","Marks","Grades"],tablefmt="grid"))

    os.startfile(filepath)

# starting of the program
def start():
    while True:
        y = input("Do you want to enter student details or search for the student details?
(enter/search/exit): ")
        if y == "enter":
            details()
        elif y=="search":
            get_details()
        elif y == "exit":
            break
        else:
            print("Invalid input. Please give input as 'enter', 'search', 'exit'")
start()
```

Input and output:

Case 1: Entering the details

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

ncher' '54344' '--' 'c:\Users\sh\Desktop\Patterns cognitive\Student details\Student_record_in_excel.py'
Do you want to enter student details or search for the student details? (enter/search/exit): enter
Enter the student name:Priya

Enter your register number:123090
Enter the date of birth:06-09-2003
Enter the gender (Male/Female):Female
Enter the language 1 mark:89
Enter the language 2 mark:76
Enter the maths mark:32
Enter the science mark:60
Enter the social science mark:98
Details have been saved to student_records_with_grades.xlsx
Do you want to enter student details or search for the student details? (enter/search/exit):
```



Case 2 : Searching for the student record using register number and date of birth and printing it.

```
Do you want to enter student details or search for the student details? (enter/search/exit): search

Enter your register number:123002
Enter the date of birth:18-07-2003
SSLC EXAMINATION:
Student details:
Student Name:     Balaji
Register Number:  123002
Date of Birth:    18-07-2003
Gender:           Male

 Marks and grades
+----------------+---------+----------+
| Subjects       | Marks   | Grades   |
+================+=========+==========+
| Language 1     | 43      | D        |
+----------------+---------+----------+
| Language 2     | 65      | B+       |
+----------------+---------+----------+
| Maths          | 87      | A+       |
+----------------+---------+----------+
| Science        | 23      | E        |
+----------------+---------+----------+
| Social Science | 66      | B+       |
+----------------+---------+----------+
| Total          | 284     |          |
+----------------+---------+----------+
| Percentage     | 56.8    |          |
+----------------+---------+----------+
| Result         | Fail    |          |
+----------------+---------+----------+
Do you want to print(yes/no)? yes
```

123002.txt - Notepad

File Edit Format View Help

```
SSLC EXAMINATION:
Student details:
Student Name:     Balaji
Register Number:  123002
Date of Birth:    18-07-2003
Gender:           Male
 Marks and grades+----------------+---------+----------+
| Subjects       | Marks   | Grades   |
+================+=========+==========+
| Language 1     | 43      | D        |
+----------------+---------+----------+
| Language 2     | 65      | B+       |
+----------------+---------+----------+
| Maths          | 87      | A+       |
+----------------+---------+----------+
| Science        | 23      | E        |
+----------------+---------+----------+
| Social Science | 66      | B+       |
+----------------+---------+----------+
| Total          | 284     |          |
+----------------+---------+----------+
| Percentage     | 56.8    |          |
+----------------+---------+----------+
| Result         | Fail    |          |
+----------------+---------+----------+
```

Date: 16/07/2024 **Day:Tuesday**

Task: Develop a Python program that extracts text from an image using Tesseract OCR and saves it to a text file. The program should then provide functionality to count words, lines, each character's occurrence, special characters, and specific words ("for", "the", "of", "it") in the text file. Users should be able to interactively choose and perform these operations, with the results displayed accordingly.

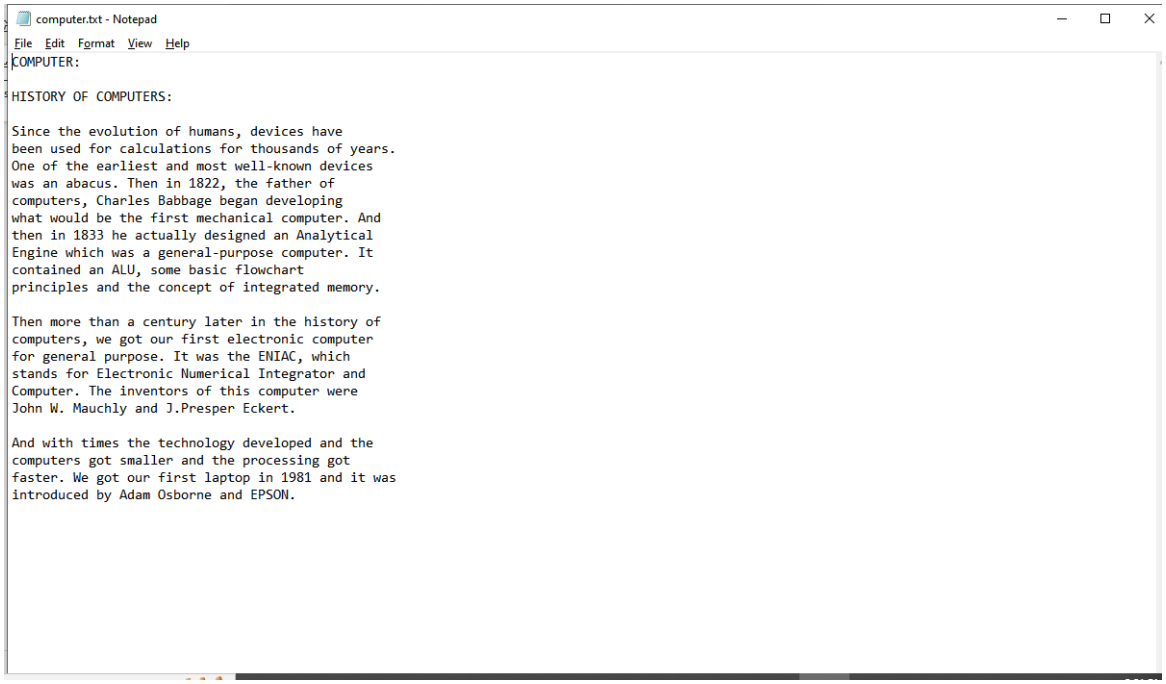
Program code:

```
import pytesseract
from PIL import Image
import os

pytesseract.pytesseract.tesseract_cmd = r'C:\Users\sh\Desktop\Pattens cognitive\Tesseract
python\tesseract.exe'

img_file="computer.jpg"

#method to count the number of words in the text file
def count_words(file):
    try:
        with open(file,"r") as f:
            content=f.read()
            word_count=len(content.split())
            print(f"No of words: {word_count}")
    except Exception as e:
        print(f"An error occured : {e}")

#method to count the number of lines in the text file
def count_lines(file):
    try:
        with open(file,"r") as f:
            content=f.read()
            line_count=content.count('\n')+1
            print(f"No of lines: {line_count}")
    except Exception as e:
        print(f"An error occured : {e}")

#method to count the occurence of each character in the text file
def count_each_character(file):
    try:
        with open (file,"r") as f:
            content=f.read()
            char={}
            for i in content:
                if i.isalnum():
                    if i in char:
                        char[i]+=1
                    else:
                        char[i]=1
        print("Count of the each character in the text file")
        for i,count in char.items():
            print(f"{i}:{count}")
    except Exception as e:
        print(f"An error occured : {e}")

#method to count the special characters in the text file
def count_special_character(file):
    try:
        with open(file,"r") as f:
            content=f.read()
            sc=0
            for i in content:
                if not i.isalnum() and not i.isspace():
                    sc+=1
            print(f"No of special characters: {sc}")
    except Exception as e:
        print(f"An error occured : {e}")
```

```python
#method to count the specific words in the textfile
def count_specific_word(file):
    try:
        with open(file,"r") as f:
            content=f.read()
            count={}
            specific_word=["for","the","of","it"]
            for word in specific_word:
                count[word]=content.count(word)
        for i,count in count.items():
            print(f"{i}:{count}")
    except Exception as e:
        print(f"An error occured : {e}")

#method to transfer the text from the image to the text file
def transfer(imag):
    try:
        img=Image.open(imag)
        ocr_result=pytesseract.image_to_string(img)
        file="computer.txt"
        with open (file,"w") as f:
            f.write(ocr_result)
    except Exception as e:
        print(f"An error occured: {e}")
    os.startfile(file)


    while True:
        op=input("Enter to perform the following operations \n1.count words \n2.count lines \n3.count
each character  \n4.count special character \n5.count specific word \n6.exit \n")
        if op.lower()=="count words":
            count_words(file)
        elif op.lower()=="count lines":
            count_lines(file)
        elif op.lower()=="count each character":
            count_each_character(file)
        elif op.lower()=="count special character":
            count_special_character(file)
        elif op.lower()=="count specific word":
            count_specific_word(file)
        elif op.lower()=="exit":
            break
        else:
            print("Invalid input")

#starting of the program
def start():
    while True:
        s=input("Enter \n1.transfer \n2.exit \n")
        if s.lower()=="transfer":
            transfer(img_file)
        elif s.lower()=="exit":
            break
        else:
            print("Invalid input")

start()
```

Input and output:

Transferring the text from image to the notepad file



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
1.transfer
2.exit
transfer
Enter to perform the following operations
1.count words
2.count lines
3.count each character
4.count special character
5.count specific word
6.exit
count words
No of words: 153
Enter to perform the following operations
1.count words
2.count lines
3.count each character
4.count special character
5.count specific word
6.exit
count lines
No of lines: 27
Enter to perform the following operations
1.count words
2.count lines
3.count each character
4.count special character
5.count specific word
6.exit
count each character
```

```
count each character
Count of the each character in the text file
C:5
O:7
M:3
P:4
U:3
T:6
E:7
R:3
H:1
I:5
S:4
Y:1
F:1
i:36
n:56
c:36
e:88
t:61
h:34
v:7
o:54
l:28
u:24
f:17
m:19
a:54
s:40
d:26
b:9
r:48
y:9
w:14
k:2
1:4
8:3
```

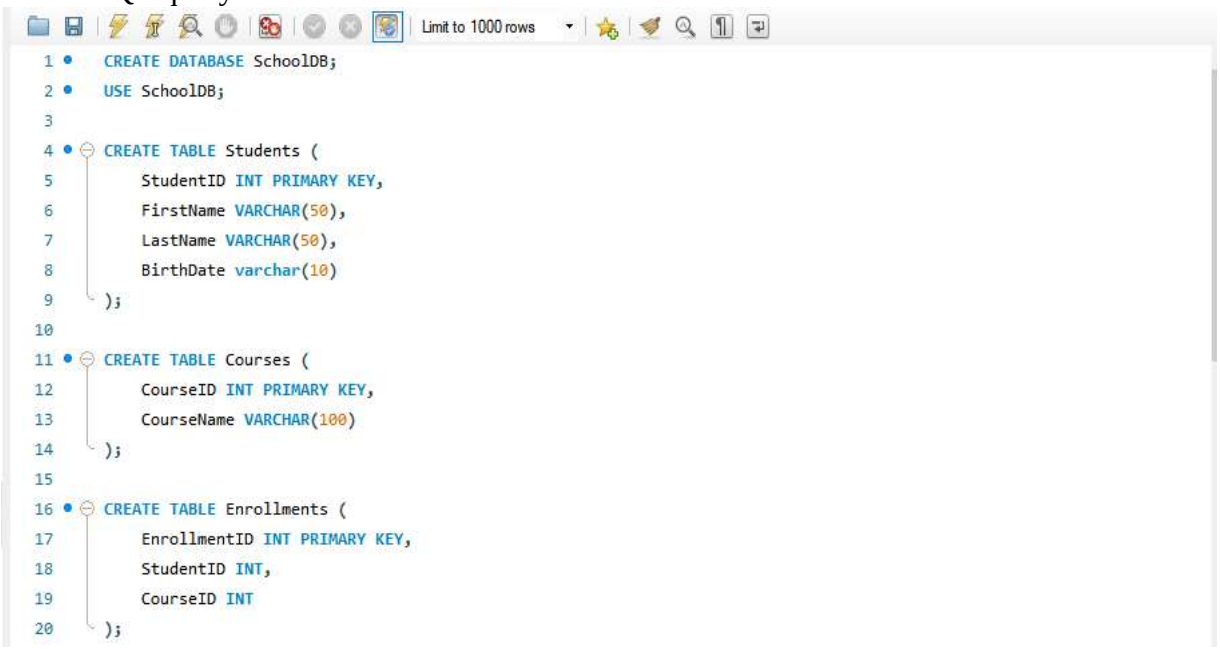
```
6.exit
count special character
No of special characters: 22
Enter to perform the following operations
1.count words
2.count lines
3.count each character
4.count special character
5.count specific word
6.exit
count specific word
Invalid input
Enter to perform the following operations
1.count words
2.count lines
3.count each character
4.count special character
5.count specific word
6.exit
count specific word
for:4
the:12
of:7
it:2
Enter to perform the following operations
1.count words
2.count lines
3.count each character
4.count special character
5.count specific word
6.exit
exit
Enter
1.transfer
2.exit
exit
PS C:\Users\sh\Desktop\Patterns cognitive>
```

Date: 17/07/2024 **Day:Wednesday**

Task: Create a database with at least two related tables. Populate the tables with sample data. Practice SQL queries such as selecting specific columns, filtering records, joining tables, updating records, and deleting records based on certain conditions.

SQL QUERIES:

Create SQL query



Insert details into tables:

```
21
22 • INSERT INTO Students (StudentID, FirstName, LastName, BirthDate) VALUES (1, 'Arun', 'Sundar', '09-09-2003');
23 • INSERT INTO Students (StudentID, FirstName, LastName, BirthDate) VALUES (2, 'Meena', 'Raj', '12-05-2002');
24 • INSERT INTO Students (StudentID, FirstName, LastName, BirthDate) VALUES (3, 'Ravi', 'Kumar', '03-11-2001');
25 • INSERT INTO Students (StudentID, FirstName, LastName, BirthDate) VALUES (4, 'Anita', 'Sharma', '21-02-2003');
26 • INSERT INTO Students (StudentID, FirstName, LastName, BirthDate) VALUES (5, 'Vijay', 'Patel', '15-08-2000');
27
28 • INSERT INTO Courses (CourseID, CourseName) VALUES (1, 'Mathematics');
29 • INSERT INTO Courses (CourseID, CourseName) VALUES (2, 'Physics');
30 • INSERT INTO Courses (CourseID, CourseName) VALUES (3, 'Chemistry');
31 • INSERT INTO Courses (CourseID, CourseName) VALUES (4, 'Biology');
32 • INSERT INTO Courses (CourseID, CourseName) VALUES (5, 'English');
33
34 • INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID) VALUES (1, 1, 1);
35 • INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID) VALUES (3, 3, 2);
36 • INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID) VALUES (4, 4, 3);
37 • INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID) VALUES (5, 5, 4);
38 • INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID) VALUES (6, 1, 2);
```

Select query using where condition:

```
39
40     -- Select all students born after 01-01-2002
41 •   SELECT * FROM Students WHERE BirthDate> '01-01-2002';
42     -- Select all enrollments for a specific student
43 •   SELECT * FROM Enrollments WHERE StudentID = 1;
44
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

StudentID	FirstName	LastName	BirthDate
1	Arun	Sundar	09-09-2003
2	Meena	Raj	12-05-2002
3	Ravi	Kumar	03-11-2001
4	Anita	Sharma	21-02-2003
5	Vijay	Patel	15-08-2000
NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

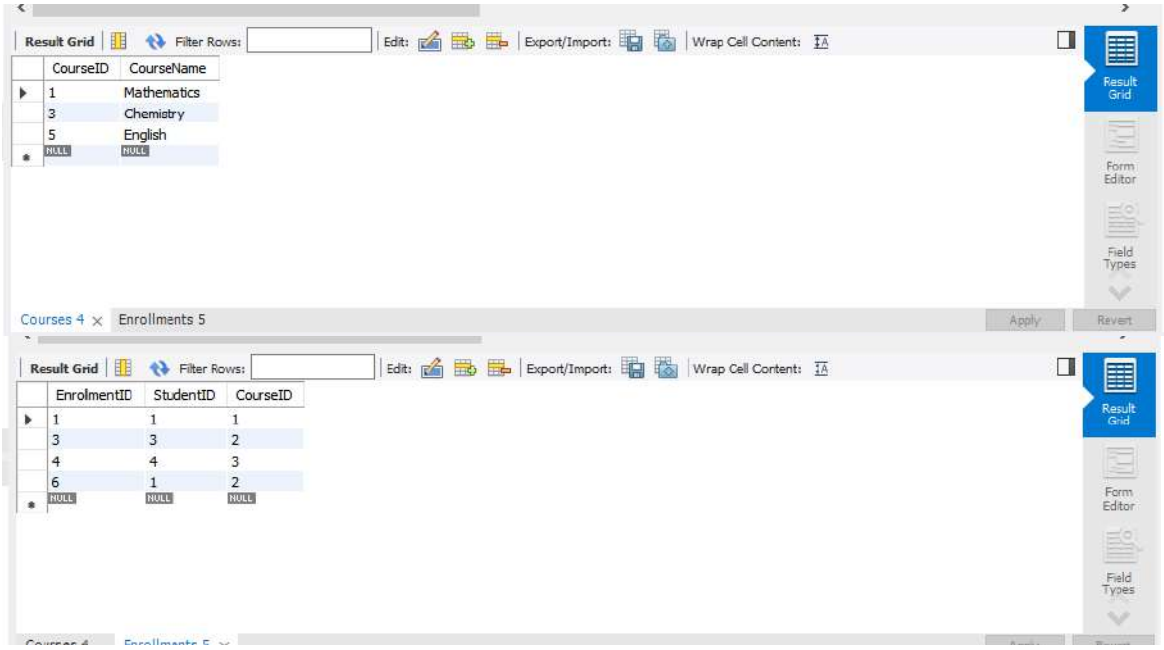
EnrollmentID	StudentID	CourseID
1	1	1
6	1	2
NULL	NULL	NULL

Result Grid, Form Editor, Field Types

Students 2 | Enrollments 3 | Apply | Revert

Select query using in clause:

```
45     -- Select all courses with CourseID in a specific list
46 •   SELECT * FROM Courses
47     WHERE CourseID IN (1, 3, 5);
48
49     -- Select all students enrolled in specific courses
50 •   SELECT * FROM Enrollments
51     WHERE CourseID IN (1, 2, 3);
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CourseID	CourseName
1	Mathematics
3	Chemistry
5	English
NULL	NULL

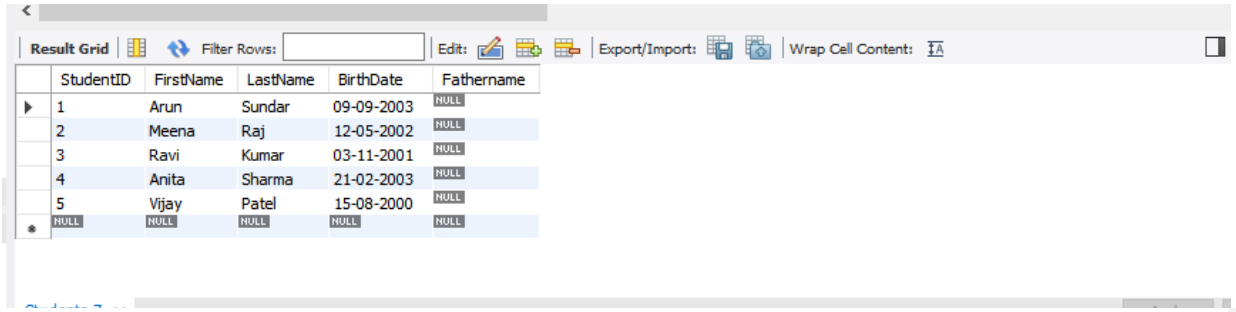
Courses 4 | Enrollments 5 | Apply | Revert

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EnrolmentID	StudentID	CourseID
1	1	1
3	3	2
4	4	3
6	1	2
NULL	NULL	NULL

Alter query to modify the table:

```
53 •  ALTER TABLE Students ADD Fathername VARCHAR(100);
54 •  select * from Students;
```



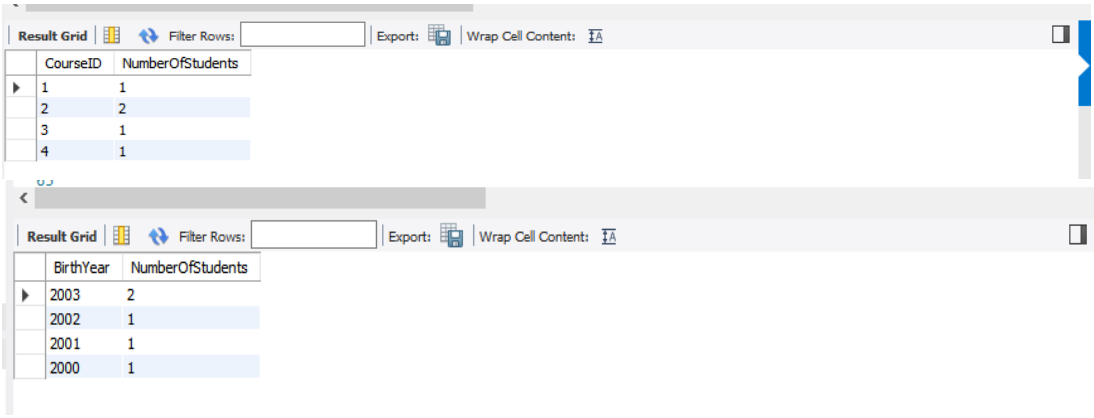
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

StudentID	FirstName	LastName	BirthDate	Fathername
1	Arun	Sundar	09-09-2003	NULL
2	Meena	Raj	12-05-2002	NULL
3	Ravi	Kumar	03-11-2001	NULL
4	Anita	Sharma	21-02-2003	NULL
5	Vijay	Patel	15-08-2000	NULL
NULL	NULL	NULL	NULL	NULL

```
55 •  ALTER TABLE Students MODIFY BirthDate DATE;
56 •  ALTER TABLE Students DROP COLUMN Fathername;
```

Group by clause:

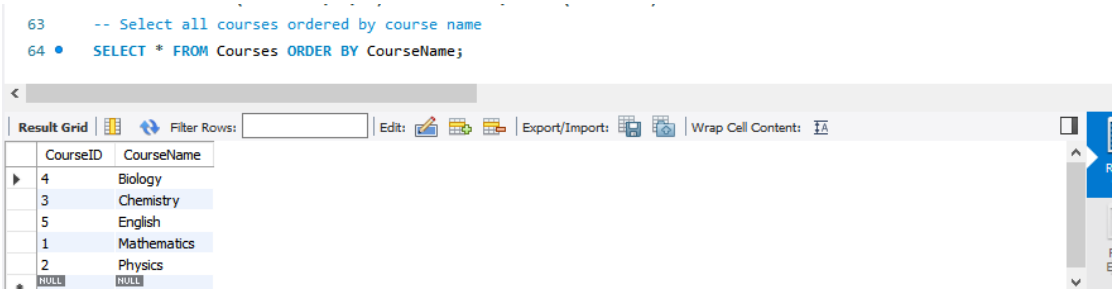
```
57
58      -- Count the number of students enrolled in each course
59 •  SELECT CourseID, COUNT(StudentID) AS NumberOfStudents FROM Enrollments GROUP BY CourseID;
60
61      -- Group students by birth year
62 •  SELECT SUBSTRING(BirthDate, 7, 4) AS BirthYear, COUNT(StudentID) AS NumberOfStudents FROM Students GROUP BY BirthYear;
63
```



CourseID	NumberOfStudents
1	1
2	2
3	1
4	1

BirthYear	NumberOfStudents
2003	2
2002	1
2001	1
2000	1

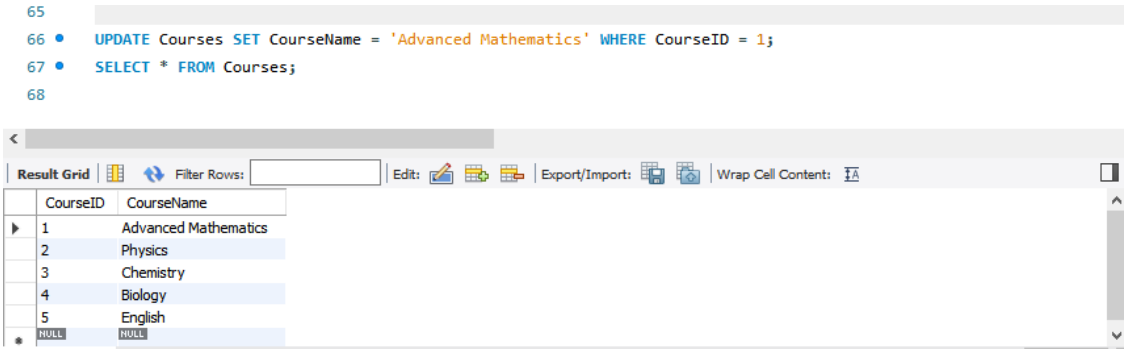
Order by:



```
-- Select all courses ordered by course name
SELECT * FROM Courses ORDER BY CourseName;
```

CourseID	CourseName
4	Biology
3	Chemistry
5	English
1	Mathematics
2	Physics

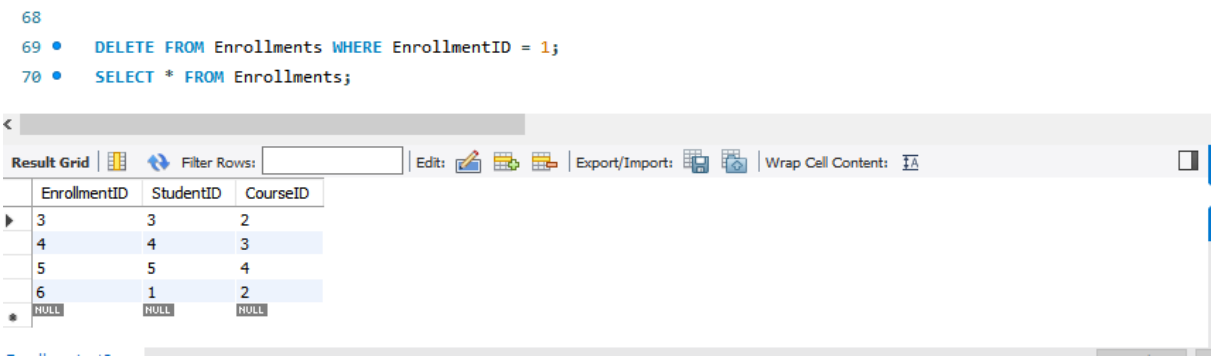
Update query:



```
UPDATE Courses SET CourseName = 'Advanced Mathematics' WHERE CourseID = 1;
SELECT * FROM Courses;
```

CourseID	CourseName
1	Advanced Mathematics
2	Physics
3	Chemistry
4	Biology
5	English

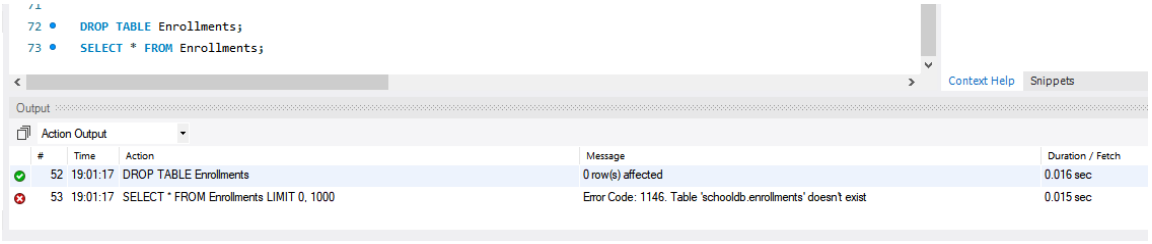
Delete query:



```
DELETE FROM Enrollments WHERE EnrollmentID = 1;
SELECT * FROM Enrollments;
```

EnrollmentID	StudentID	CourseID
3	3	2
4	4	3
5	5	4
6	1	2

Drop query:



Transaction Control Language (TCL):

1.) Begin Transaction:
START TRANSACTION;

2.)Commit Transaction:
COMMIT;

3.)Rollback Transaction:
ROLLBACK;

4.)Savepoint and Rollback to Savepoint:
SAVEPOINT sp1;
UPDATE Students SET Email = 'new.email@example.com' WHERE StudentID = 1; ROLLBACK TO sp1;

Data Control Language (DCL):

1.) Grant Privileges:
GRANT SELECT, INSERT ON Students TO 'root'@'localhost';

2.)Revoke Privileges:
REVOKE INSERT ON Students FROM 'root'@'localhost';

Date:18/07/2024 **Day:Thursday**

Task: Create a database and tables and perform sql queries

-- database that is being used has to be specified first
use school;

-- to create a table with column names and their data types
create table student_table (student_name varchar(20),

```
student_regno int,
student_dob varchar(10),
language1_mark int,
language2_mark int,
maths_mark int,
science_mark int,
social_mark int);

-- renaming the table
rename table student_table to student_record;

-- transaction that is started
start transaction;
-- inserting the values of the columns that is records
insert into student_record values("Akila Sundar",123001,"08-09-2003",67,78,89,90,99),
("Balaji",123002,"18-07-2003",34,56,88,72,81),
("Catherine",123003,"07-05-
2003",52,93,86,82,88),("Dinesh",123004,"09-03-2003",44,78,76,71,23),
("Elanthir",123005,"08-12-2003",92,54,32,66,73);
-- updating a record with specific where condition
update student_record set maths_mark=100 where student_name="Dinesh";
-- commit to save all the commands that has been done till this
commit;

-- to display all the records in the student_record table
select * from student_record;
start transaction;
-- adding an another column to the table
alter table student_record add column total varchar(10);
-- modifying the existing column
alter table student_record modify column total int;
-- the column of total marks is calculated for each record with the help of set command
update student_record set total=language1_mark+language2_mark+maths_mark+science_mark+social_mark;
commit;
select * from student_record;

-- trigger is to update the existing column if we are to insert a new record with the condition and
```

```
-- delimiter is removed and then changed to ; again at the end of the trigger
delimiter //
create trigger before_insert_student before insert on student_record for each row begin
set
new.total=new.language1_mark+new.language2_mark+new.maths_mark
+new.science_mark+new.social_mark;
end;
//
delimiter ;

-- inserting a new record but the user is no giving input to the total column instead the trigger that is created
-- will update the total column with the given values
insert into student_record (student_name,student_regno,student_dob,language1_mark,language2_mark,maths_mark,
science_mark,social_mark) values("Fathima",123006,"06-11-2003",87,54,39,42,31);
select * from student_record;
insert into student_record (student_name,student_regno,student_dob,language1_mark,language2_mark,maths_mark,
science_mark,social_mark) values("Girish Kumar",123007,"14-03-2003",97,58,89,46,81),("Harsha",123008,"30-03-2003",88,87,82,70,79),
("Iniya",123009,"04-01-2003",87,58,29,68,86),("Janani",123010,"14-09-2003",76,45,29,55,89);
alter table student_record add column percentage float;
update student_record set percentage=total/5;
commit;
alter table student_record add column result varchar(10);
-- select statement displays only the record tha satisfies the where condition
select
student_name,language1_mark,language2_mark,maths_mark,science_mark,social_mark from student_record
where student_regno=123007;
select * from student_record where total>400;
select * from student_record where student_regno>=123005;

-- updating the result column without the user input
update student_record set result=case when language1_mark<35
```

```
or language2_mark<35 or maths_mark<35 or science_mark<35 or social_mark<35 then "fail"
else "pass"
end;
select * from student_record;
alter table student_record add column gender varchar(10);
update student_record set gender="male";

-- delete a column in the table using drop column command in the alter table method
alter table student_record drop column gender;
-- deleting a record wih specific condition in the table
delete from student_record where student_regno=123010;

-- order by default=ascending otherwise desc is given
select * from student_record order by student_name desc;
-- to count the number of record in the student record table
select count(*) as total_students from student_record;
select result,count(*) as total_students from student_record group by result;
select result from student_record group by result;
select total , count(*) as total_students from student_record group by total;
select * from student_record;

-- to round of the average value to 2 decimal places
select round(avg(language1_mark),2) as language1_average ,
round(avg(language2_mark),2) as language2_average,
round(avg(maths_mark),2) as maths_average,
round(avg(science_mark),2) as science_average,
round(avg(social_mark),2) as social_average,
round(avg(total),2) as class_average,
round(avg(percentage),2) as class_percentage from student_record;

-- the command below is used to know the existing users in mysql
select user , host from mysql.user;

-- granting the permission to perform all operations on the table student record to the existing users
grant all privileges on student_record.* to 'root'@'localhost';
rollback;
-- revoking all privileges from the user
revoke all privileges, grant option from 'root'@'localhost';
rollback;
```

```
-- granting only these privileges to the user
grant select,update,insert on student_record.* to 'root'@'localhost';
-- shows the permission that are available for the speciied user
show grants for'root'@'localhost';
-- revoking the update permission from the user root
revoke update on student_record.* from 'root'@'localhost';
rollback ;

use school;
grant all privileges on school to 'root'@'localhost';
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'root';
SELECT user,authentication_string,plugin,host FROM mysql.user;


select * from student_record;
```

Date:22/07/2024 **Day:Monday**

Task: Create a Python program that manages student records by capturing and validating student details, saving them to an Excel file, and providing a search functionality to retrieve and display student details. The program should handle inputs for student name, register number, date of birth, gender, and marks for various subjects. It should calculate grades, total marks, and the final result (pass/fail). Additionally, the program should enable printing of student details to a text file if a matching record is found.

Program code:

```
from datetime import datetime
import re
import os
from openpyxl import Workbook, load_workbook
from openpyxl.styles import Font, PatternFill

def details():
    # Function to get and validate student name
    try:
        s_name = input("Enter the student name:")
        while not re.match(r'^[a-zA-Z\s]+$', s_name):
            if not s_name.isalnum():
                print("You have entered some special characters. Please provide name in alphabets.")
            else:
                print("You have entered numbers. Please provide name in alphabets.")
            s_name = input("Enter the student name:")
    except Exception as e:
        print(f"An error occurred: {e}")

    # Student register no
    try:
        s_regno = input("\nEnter your register number:")
        while not s_regno.isdigit() or len(s_regno) != 6:
            if s_regno.isalpha():
                print("You have entered alphabets. Please enter only digits.")
            elif s_regno.isalnum():
                print("You have entered a combination of letters and digits. Please enter only
digits.")
            else:
                print("You have entered special characters. Please enter only digits.")
            s_regno = input("\nEnter your register number:")
    except Exception as e:
        print(f"An error occurred: {e}")

    # Student date of birth
    try:
        while True:
            stud_dob = input("Enter the date of birth (DD-MM-YYYY):")
            format_of_dob = r'^\d{2}-\d{2}-\d{4}$'
            if re.match(format_of_dob, stud_dob):
                break
            else:
                print("Invalid format of date of birth.")
    except Exception as e:
        print(f"An error occurred: {e}")

    gender = input("Enter the gender (Male/Female):")

    stud_marks=[]
    for i in range(5):
        if i==0:
            #language 1 mark
            while True:
                try:
                    lang1_mark=int(input("Enter the language 1 mark:"))
                    if lang1_mark<0 or lang1_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(lang1_mark)
                        break
                except ValueError as ve:
                    print(ve)
        elif i==1:
            #language 2 mark
            while True:
                try:
                    lang2_mark=int(input("Enter the language 2 mark:"))
                    if lang2_mark<0 or lang2_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(lang2_mark)
                        break
                except ValueError as ve:
                    print(ve)
        elif i==2:
            #maths mark
            while True:
                try:
                    maths_mark=int(input("Enter the maths mark:"))
                    if maths_mark<0 or maths_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(maths_mark)
                        break
                except ValueError as ve:
                    print(ve)
        elif i==3:
            #science mark
            while True:
                try:
                    sci_mark=int(input("Enter the science mark:"))
                    if sci_mark<0 or sci_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(sci_mark)
                        break
                except ValueError as ve:
                    print(ve)
        elif i==4:
            #social science mark
            while True:
                try:
                    socsci_mark=int(input("Enter the social science mark:"))
                    if socsci_mark<0 or socsci_mark>100:
                        raise ValueError("Marks should be within 0-100")
                    else:
                        stud_marks.append(socsci_mark)
                        break
                except ValueError as ve:
                    print(ve)
    #print(stud_marks)
```

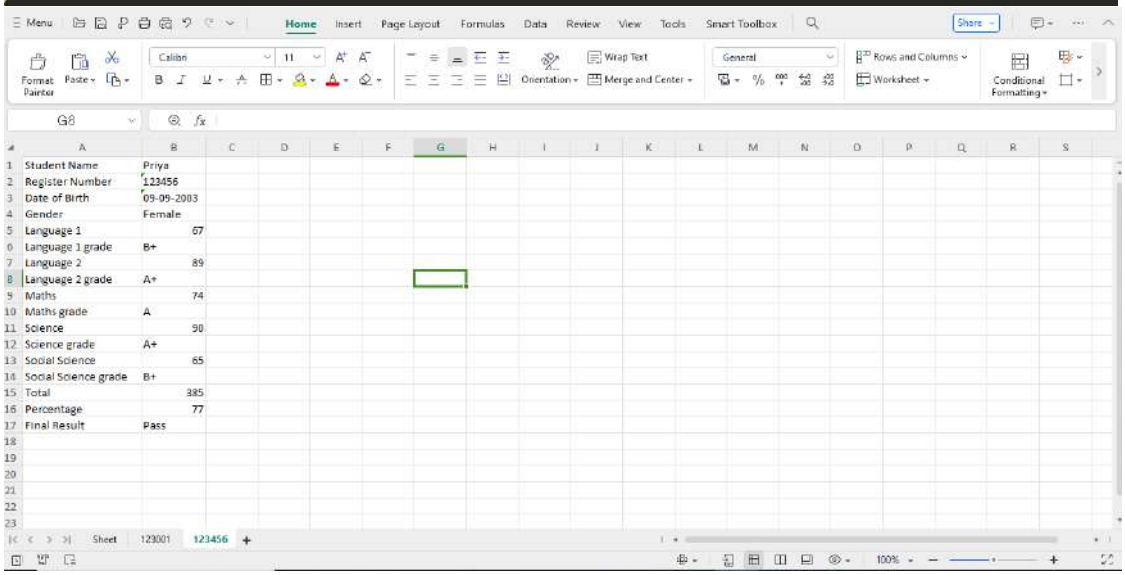
```
    grades=[]
    for i in stud_marks:
        if 90<i<=100:
            grades.append("O")
        elif 80<i<=90:
            grades.append("A+")
        elif 70<i<=80:
            grades.append("A")
        elif 60<i<=70:
            grades.append("B+")
        elif 50<i<=60:
            grades.append("C")
        elif 35<i<=50:
            grades.append("D")
        else:
            grades.append("E")

    #print(grades)

    stud_total=0
    stud_total=lang1_mark+lang2_mark+maths_mark+sci_mark+socsci_mark

    stud_percent=(stud_total/500)*100
    #pass or fail

    try:
        if lang1_mark<35 or lang2_mark<35 or maths_mark<35 or sci_mark<35 or socsci_mark<35:
            final_result="Fail"
        else:
            final_result="Pass"
    except ValueError as ve:
        print(ve)
    save_to_excel(s_name, s_regno, stud_dob, gender, stud_marks,grades, stud_total, stud_percent,
final_result)

def save_to_excel(s_name, s_regno, stud_dob, gender, stud_marks, grades, stud_total, stud_percent,
final_result):
    file_path = "student_records_in_sheets.xlsx"
    if os.path.exists(file_path):
        wb = load_workbook(file_path)
    else:
        wb = Workbook()

    if s_regno in wb.sheetnames:
        sheet = wb[s_regno]
    else:
        sheet = wb.create_sheet(title=s_regno)
        headers = [
            "Student Name", "Register Number", "Date of Birth", "Gender",
            "Language 1", "Language 1 grade", "Language 2", "Language 2 grade",
            "Maths", "Maths grade", "Science", "Science grade", "Social Science",
            "Social Science grade", "Total", "Percentage", "Final Result"
        ]
        values = [
            s_name, s_regno, stud_dob, gender, stud_marks[0], grades[0],
            stud_marks[1], grades[1], stud_marks[2], grades[2], stud_marks[3],
            grades[3], stud_marks[4], grades[4], stud_total, stud_percent, final_result
        ]
        for i in range(len(headers)):
            sheet.append([headers[i], values[i]])

    wb.save(file_path)
    print(f"Details have been saved to {file_path}")
```

```
def get_details():
    def regno():
        return input("Enter the register no (only 6 digits): ")

    stu_regno = regno()
    while not stu_regno.isdigit() or len(stu_regno) != 6:
        print("Invalid input. Please enter a 6-digit register number.")
        stu_regno = regno()

    while True:
        stu_dob = input("Enter the date of birth (DD-MM-YYYY):")
        format_of_dob = r'^\d{2}-\d{2}-\d{4}$'
        if re.match(format_of_dob, stu_dob):
            break
        else:
            print("Invalid format of date of birth. Please use DD-MM-YYYY format.")

    file = "student_records_in_sheets.xlsx"
    student_found = False

    if os.path.exists(file):
        wb = load_workbook(file)
        if stu_regno in wb.sheetnames:
            ws = wb[stu_regno]
            details = {}
            for row in ws.iter_rows(values_only=True):
                details[row[0]] = row[1]
            if details["Register Number"] == stu_regno and details["Date of Birth"] == stu_dob:
                print("SSLC EXAMINATION:")
                print("Student details:")
                for key, value in details.items():
                    print(f"{key}: {value}")
                student_found = True
                p = input("Do you want to print (yes/no)? ")
                if p.lower() == "yes":
                    print_details(details)
        if not student_found:
            print("No student details are found.")
    else:
        print("No student records found.")

def print_details(details):
    filename = "Students folder"
    if not os.path.exists(filename):
        os.makedirs(filename)

    file = f"{details['Register Number']}.txt"
    filepath = os.path.join(filename, file)

    with open(filepath, "w") as f:
        f.write("SSLC EXAMINATION:\n")
        f.write("Student details:\n")
        for key, value in details.items():
            f.write(f"{key}: {value}\n")

    os.startfile(filepath)
# Starting of the program
def start():
    while True:
        y = input("Do you want to enter student details or search for the student details?
(enter/search/exit): ")
        if y == "enter":
            details()
        elif y == "search":
            get_details()
        elif y == "exit":
            break
        else:
            print("Invalid input. Please give input as 'enter', 'search', 'exit'.")

start()
```

Input and output:

Entering the student details:

```
Do you want to enter student details or search for the student details? (enter/search/exit): enter
Enter the student name:Priya

Enter your register number:123456
Enter the date of birth (DD-MM-YYYY):09-09-2003
Enter the gender (Male/Female):Female
Enter the language 1 mark:67
Enter the language 2 mark:89
Enter the maths mark:74
Enter the science mark:90
Enter the social science mark:65
Details have been saved to student_records_in_sheets.xlsx
Do you want to enter student details or search for the student details? (enter/search/exit):
```



	A	B
1	Student Name	Priya
2	Register Number	123456
3	Date of Birth	09-09-2003
4	Gender	Female
5	Language 1	67
6	Language 1 grade	B+
7	Language 2	89
8	Language 2 grade	A+
9	Maths	74
10	Maths grade	A
11	Science	90
12	Science grade	A+
13	Social Science	65
14	Social Science grade	B+
15	Total	385
16	Percentage	77
17	Final Result	Pass

Searching for a student record:

```
Do you want to enter student details or search for the student details? (enter/search/exit): search
Enter the register no (only 6 digits): 123001
Enter the date of birth (DD-MM-YYYY):08-09-2003
SSLC EXAMINATION:
Student details:
Student Name: Akila Sundar
Register Number: 123001
Date of Birth: 08-09-2003
Gender: Female
Language 1: 67
Language 1 grade: B+
Language 2: 89
Language 2 grade: A+
Maths: 76
Maths grade: A
Science: 90
Science grade: A+
Social Science: 85
Social Science grade: A+
Total: 407
Percentage: 81.4
Final Result: Pass
Do you want to print (yes/no)? yes
```

Date:23/07/2024 and 24/07/2024
Day:Tuesday and Wednesday

Task: Design a Python program to manage student records in a MySQL database. The program should:

1.) Allow users to create a new database and tables with customizable column names.
2.) Insert, update, search, delete, and display student records with validations for various input fields such as name, register number, date of birth, and marks.
3.) Include functionality to alter tables by adding or deleting columns.

Program code:

```
import mysql.connector
import re

#connection to mysql
def connect_to_mysql():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="Satram2003!123"
    )

#creating the database
def create_database(cursor):
    try:
        db_name = input("Enter the database name to be created: ")
        cursor.execute(f"SHOW DATABASES LIKE '{db_name}'")
        result = cursor.fetchone()
        if result:
            print(f"Database '{db_name}' already exists.")
        else:
            cursor.execute(f"CREATE DATABASE `{db_name}`")
            print(f"Database '{db_name}' created successfully.")
        return db_name
    except Exception as e:
        print(f"An error occurred: {e}")

#creating the tabele and passing the cursor as the parameter
def create_table(cursor):
    try:
        while True:
            table_name = input("Enter the table name to be created (or type 'exit' to stop): ")
            if table_name.lower() == 'exit':
                break

            cursor.execute(f"SHOW TABLES LIKE '{table_name}'")
            result = cursor.fetchone()
            if result:
                print(f"Table '{table_name}' already exists.")
                continue

            columns = []
            print("Enter column details (type 'done' to finish):")
            while True:
                column = input("Column name (or type 'done'): ")
                if column.lower() == 'done':
                    break
                col_type = input(f"Data type for column '{column}': ")
                columns.append(f"`{column}` {col_type}")

            columns_str = ", ".join(columns)
            create_table_query = f"CREATE TABLE `{table_name}` ({columns_str});"
            cursor.execute(create_table_query)
            print(f"Table '{table_name}' created successfully.")
    except Exception as e:
        print(f"An error occurred: {e}")

#inserting record in the specified table in the database
def insert_student_record(cursor, table_name):
    try:
        # Student name
        while True:
            s_name = input("Enter the student name: ")
            if re.match(r'^[a-zA-Z\s]+$', s_name):
                break
            elif not s_name.isalnum():
                print("You have entered some special characters. Please provide name in alphabets")
            else:
                print("You have entered numbers. Please provide name in alphabet")

        # Student registor number
        while True:
            s_regno = input("Enter the student register number: ")
            if s_regno.isdigit() and len(s_regno) == 6:
                break
            elif s_regno.isalpha():
                print("You have entered alphabets. Please enter only digits")
            elif s_regno.isalnum():
                print("You have entered a combination of letters and digits. Please enter only digits")
            else:
                print("You have entered special characters. Please enter only digits")
            if len(s_regno) != 6:
                print("Register number should contain 6 digits only.")

        # Student date of birth
        while True:
            stud_dob = input("Enter the date of birth (DD-MM-YYYY): ")
            if re.match(r'^\d{2}-\d{2}-\d{4}$', stud_dob):
                break
            else:
                print("Invalid format of date of birth")

        #language 1 mark
        while True:
            try:
                lang1_mark=int(input("Enter the language 1 mark:"))
                if lang1_mark<0 or lang1_mark>100:
                    raise ValueError("Marks should be within 0-100")
                else:
                    break
            except ValueError as ve:
                print(ve)
        #language2 mark
        while True:
            try:
                lang2_mark=int(input("Enter the language 2 mark:"))
                if lang2_mark<0 or lang2_mark>100:
                    raise ValueError("Marks should be within 0-100")
                else:
                    break
            except ValueError as ve:
                print(ve)
        #maths mark
        while True:
            try:
                maths_mark=int(input("Enter the maths mark:"))
                if maths_mark<0 or maths_mark>100:
                            raise ValueError("Marks should be within 0-100")
                else:
                    break
            except ValueError as ve:
                        print(ve)
        #science mark
```

```
        while True:
            try:
                sci_mark=int(input("Enter the science mark:"))
                if sci_mark<0 or sci_mark>100:
                    raise ValueError("Marks should be within 0-100")
                else:
                    break
            except ValueError as ve:
                        print(ve)
        #social science mark
        while True:
            try:
                socsci_mark=int(input("Enter the social science mark:"))
                if socsci_mark<0 or socsci_mark>100:
                    raise ValueError("Marks should be within 0-100")
                else:
                    break
            except ValueError as ve:
                print(ve)


        stud_total = lang1_mark + lang2_mark + maths_mark + sci_mark + socsci_mark
        stud_percent = (stud_total / 500) * 100

        if lang1_mark<35 or lang2_mark<35 or maths_mark<35 or sci_mark<35 or socsci_mark<35:
            result="Fail"
        else:
            result="Pass"

        insert_query = f"INSERT INTO {table_name}  VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
        cursor.execute(insert_query, (s_name, s_regno, stud_dob, lang1_mark, lang2_mark, maths_mark,
sci_mark, socsci_mark, stud_total, stud_percent, result))
        print(f"Record inserted successfully into table '{table_name}'")

    except Exception as e:
        print(f"An error occurred: {e}")

def alter_table(cursor,table_name):
    action=input("Do you want to add a column or delete a column?(add/delete):")
    if action.lower() == "add":
        new_col_name = input("Enter the column name to be added:")
        #data_type = input(f"Enter the data type of the {new_col_name}: ")

        # Adding backticks around table and column names to handle special characters
        add_column_query = f"ALTER TABLE {table_name} ADD COLUMN {new_col_name} VARCHAR(10) ;"
        #add_column_query=f"alter table Student_record add column gender VARCHAR(10);"
        try:
            cursor.execute(add_column_query)
            #print("added")
            print(f"The column '{new_col_name}' has been added to the table '{table_name}'.")
        except mysql.connector.Error as e:
            print(f"Error: {e}")
    elif action.lower()=="delete":
        cursor.execute(f"SHOW COLUMNS FROM {table_name};")
        records=cursor.fetchall()
        for col in records:
            print(col)
        col_name=input("Enter the column name to be deleted:")
        cursor.execute(f"ALTER TABLE {table_name} DROP COLUMN {col_name};")
        print(f"The {col_name} column is deleted successfully from {table_name}")
    else:
        print("Invalid action to perform in the table")
```

```python
#updating the existing record in the specified table in the database
def update_student_record(cursor,table_name):
    try:
        while True:
            s_regno = input("Enter the student register number that has to be modified: ")
            if s_regno.isdigit() and len(s_regno) == 6:
                break
            elif s_regno.isalpha():
                print("You have entered alphabets. Please enter only digits")
            elif s_regno.isalnum():
                print("You have entered a combination of letters and digits. Please enter only digits")
            else:
                print("You have entered special characters. Please enter only digits")
            if len(s_regno) != 6:
                print("Register number should contain 6 digits only.")
        cursor.execute(f"SELECT * FROM {table_name} WHERE Reg_no=%s ",(s_regno,))
        result=cursor.fetchone()
        if result:
            print(f"Record exists with the register number {s_regno}")
            cursor.execute(f"SHOW COLUMNS FROM {table_name};")
            records=cursor.fetchall()
            for col in records:
                print(col)
            col_name=input("Enter the column name that has to be modified:")
            new_value=input("Enter the new value of that column:")
            cursor.execute(f"UPDATE {table_name} SET {col_name} =%s WHERE Reg_no=%s",
(new_value,s_regno))
            print("Updated the record successfully")

            if col_name in ["Lang1_mark","Lang2_mark","Maths_mark","Science_mark","Social_mark"]:
                cursor.execute(f"SELECT Lang1_mark,Lang2_mark,Maths_mark,Science_mark,Social_mark FROM
{table_name} WHERE Reg_no=%s",(s_regno,))
                marks=cursor.fetchone()
                lang1_mark , lang2_mark , maths_mark , sci_mark , socsci_mark=marks
                s_total = lang1_mark + lang2_mark + maths_mark + sci_mark + socsci_mark
                s_percent = (s_total / 500) * 100
                if lang1_mark<35 or lang2_mark<35 or maths_mark<35 or sci_mark<35 or socsci_mark<35:
                    res="Fail"
                else:
                    res="Pass"

                cursor.execute(f"UPDATE {table_name} SET Total=%s,Percentage=%s,Result=%s WHERE
Reg_no=%s",(s_total,s_percent,res,s_regno))
                print("Records are updated successfully")


        else:
            print(f"Record with register number {s_regno} does not exist.")

    except Exception as e:
        print(f"An error occured {e}")

#searching for the specific record in the table
def search_student_record(cursor, table_name):
    try:
        try:
            s_regno=input("\nEnter your register number:")
            while not s_regno.isdigit():
                if s_regno.isalpha():
                    print("You have entered alphabets.Please enter only digits")
                    s_regno=input("\nEnter your register number:")
                elif s_regno.isalnum():
                    print("You have entered the combination of letters and digits.Please enter only
digits")
                    s_regno=input("\nEnter your register number:")
                else:
                    print("You have entered special characters.Please enter only digits")
                    s_regno=input("\nEnter your register number:")
            while len(s_regno)!=6:
                print("Reguster number should contain 6 digits only.")
                s_regno=input("\nEnter your register number:")
        except Exception as e:
            print(f"An error occured {e}")

        #student date of birth
        try:
            while True:
                stud_dob=input("Enter the date of birth:")
                format_of_dob=r'^\d{2}-\d{2}-\d{4}$'
                if re.match(format_of_dob,stud_dob):
                    #print("Date of birth:",stud_dob)
                    break
                else:
                    print("Invalid format of date of birth")
        except Exception as e:
            print(f"An error occured {e}")

        select_query = f"SELECT * FROM {table_name} WHERE Reg_no = %s AND Date_of_birth = %s"
        cursor.execute(select_query, (s_regno, stud_dob))
        record = cursor.fetchone()

        if record:
            print("Record found:")
            print(f"\n Student details")
            print(f"Student Name:        {record[0]}")
            print(f"Student Register no: {record[1]}")
            print(f"Date of birth:       {record[2]}")
            print("\nMarks Obtained")
            print(f"Language 1       :  {record[3]}")
            print(f"Language 2       :  {record[4]}")
            print(f"Maths            :  {record[5]}")
            print(f"Science          :  {record[6]}")
            print(f"Social Science   :  {record[7]}")
            print(f"Total            :  {record[8]}")
            print(f"Percentage       :  {record[9]}")
            print(f"Final Result     :  {record[10]}")
        else:
            print("No matching record found.")
    except Exception as e:
        print(f"An error occurred: {e}")

#deleting the existing record in the table
```

```python
def display_student_record(cursor,table_name):
    try:
        query=f"SELECT * FROM {table_name} ORDER BY Reg_no;"
        cursor.execute(query)
        records=cursor.fetchall()
        print("Total number of records:",cursor.rowcount)
        for row in records:
            print(f"\nStudent Name:       {row[0]}")
            print(f"Student Register no: {row[1]}")
            print(f"Date of birth:       {row[2]}")
            print("\nMarks Obtained")
            print(f"Language 1       :  {row[3]}")
            print(f"Language 2       :  {row[4]}")
            print(f"Maths            :  {row[5]}")
            print(f"Science          :  {row[6]}")
            print(f"Social Science   :  {row[7]}")
            print(f"Total            :  {row[8]}")
            print(f"Percentage       :  {row[9]}")
            print(f"Final Result     :  {row[10]}")

    except Exception as e:
        print(f"An error occured {e}")
#main function
def main():
    mydb = connect_to_mysql()
    cursor = mydb.cursor()

    db_name = create_database(cursor)
    mydb.database = db_name

    create_table(cursor)

    while True:
        add_record = input("Do you want to add a record to a table? (yes/no): ")
        if add_record.lower() == 'no':
            break
        elif add_record.lower() == 'yes':
            table_name = input("Enter the table name to add records: ")
            cursor.execute(f"SHOW TABLES LIKE '{table_name}';")
            result=cursor.fetchone()
            if result:
                insert_student_record(cursor, table_name)
            else:
                print(f"The {table_name} table does not exist.Create the table to insert the details.")
                #create_table(cursor)
        else:
            print(f"Invalid input")


    while True:
        alter_table_input=input("Do you want alter the table?(yes/no):")
        if alter_table_input.lower()=="no":
            break
        elif alter_table_input.lower()=="yes":
            table_name=input("Enter the table name to be altered:")
            cursor.execute(f"SHOW TABLES LIKE '{table_name}';")
            result=cursor.fetchone()
            if result:
                alter_table(cursor,table_name)
            else:
                print(f"The {table_name} table does not exist.")
        else:
            print(f"Invalid input")

    while True:
        update_record=input("Do you want to update the record?(yes/no):")
        if update_record.lower()=="no":
            break
        elif update_record.lower()=="yes":
            table_name = input("Enter the table name to update records: ")
            cursor.execute(f"SHOW TABLES LIKE '{table_name}';")
            result=cursor.fetchone()
            if result:
                update_student_record(cursor, table_name)
            else:
                print(f"The {table_name} table does not exist.")
                #create_table(cursor)
        else:
            print(f"Invalid input")

    while True:
        display_record=input("Do you want to display the record?(yes/no):")
        if display_record.lower()=="no":
            break
        elif display_record.lower()=="yes":
            table_name = input("Enter the table name to update records: ")
            cursor.execute(f"SHOW TABLES LIKE '{table_name}';")
            result=cursor.fetchone()
            if result:
                display_student_record(cursor,table_name)
            else:
                print(f"The {table_name} table does not exist.")
                #create_table(cursor)
        else:
            print(f"Invalid input")



    while True:
        search_record = input("Do you want to search for a record? (yes/no): ")
        if search_record.lower() == 'no':
            break
        elif search_record.lower() == 'yes':
            table_name = input("Enter the table name to search records: ")
            cursor.execute(f"SHOW TABLES LIKE '{table_name}';")
            result=cursor.fetchone()
            if result:
                search_student_record(cursor, table_name)
            else:
                print(f"The {table_name} table does not exist.")
                #create_table(cursor)
        else:
            print(f"Invalid input")

    while True:
        delete_record=input("Do you want a delete the student_record?(yes/no):")
        if delete_record.lower()=="no":
            break
        elif delete_record.lower()=="yes":
            table_name=input("Enter the table name:")
            cursor.execute(f"SHOW TABLES LIKE '{table_name}';")
            result=cursor.fetchone()
            if result:
                delete_student_record(cursor, table_name)
            else:
                print(f"The {table_name} table does not exist.")
                #create_table(cursor)
        else:
            print(f"Invalid input")

    cursor.close()
    mydb.commit()
    mydb.close()

main()
```

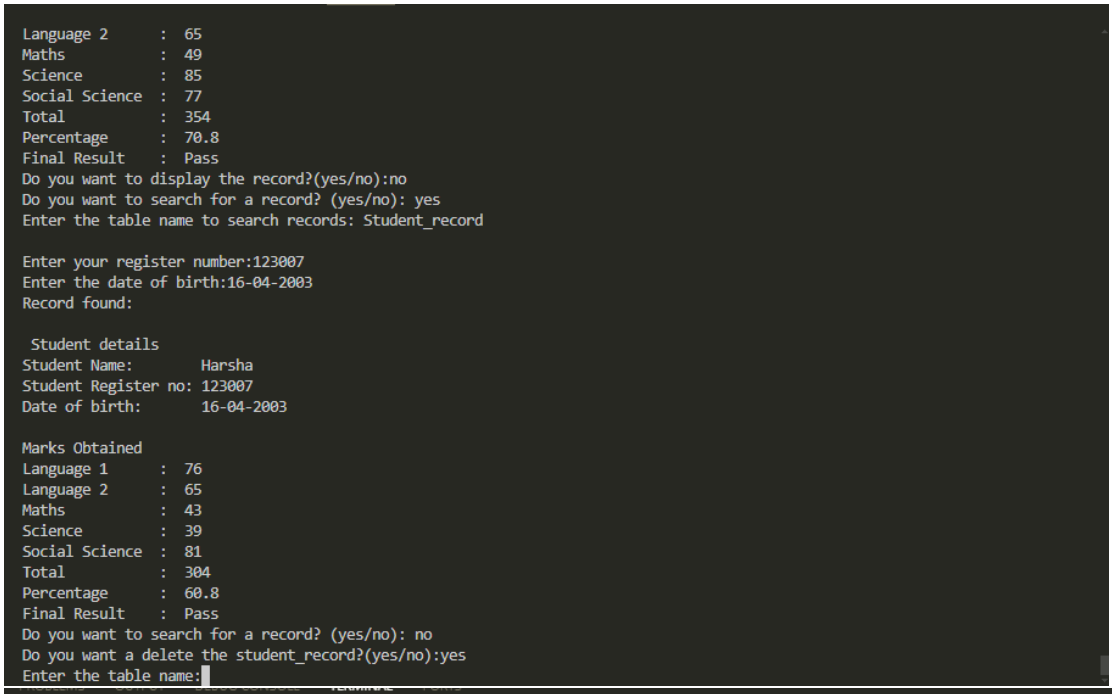
Input and Output:

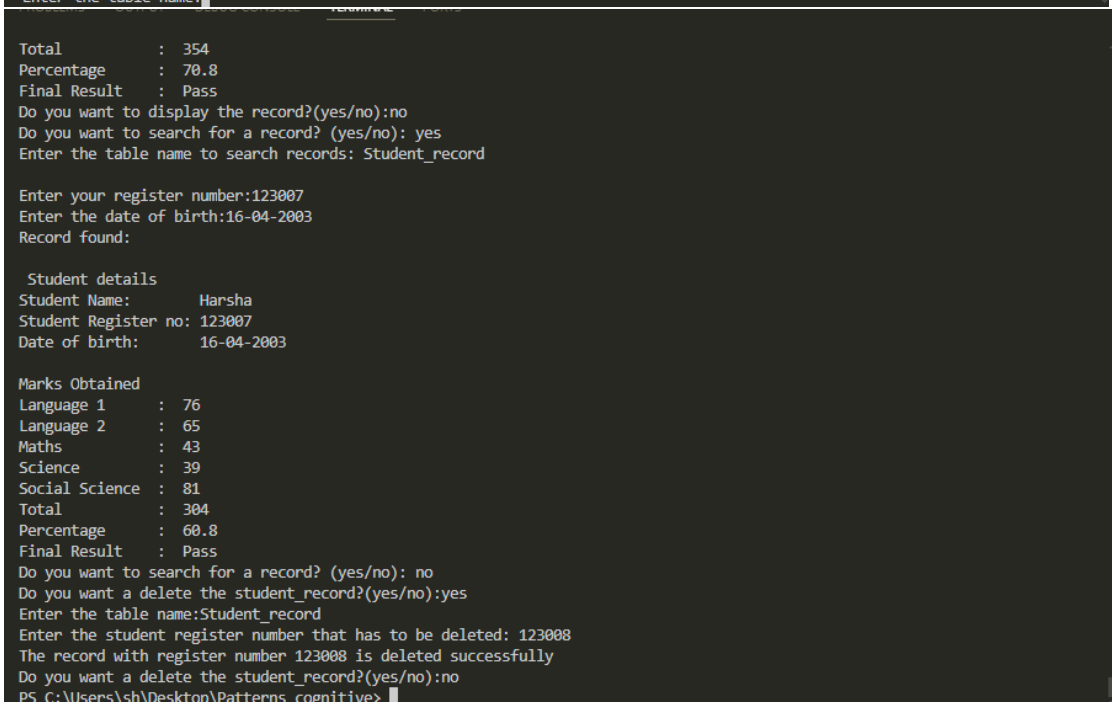
```
Enter the database name to be created: ABC_SCHOOL
Database 'ABC_SCHOOL' already exists.
Enter the table name to be created (or type 'exit' to stop): Student_record
Table 'Student_record' already exists.
Enter the table name to be created (or type 'exit' to stop): exit
Do you want to add a record to a table? (yes/no): yes
Enter the table name to add records: Student_record
Enter the student name: Priya
Enter the student register number: 123456
Enter the date of birth (DD-MM-YYYY): 07-09-2003
Enter the language 1 mark:98
Enter the language 2 mark:87
Enter the maths mark:-90
Marks should be within 0-100
Enter the maths mark:76
Enter the science mark:65
Enter the social science mark:32
```

```
Do you want alter the table?(yes/no):no
Do you want to update the record?(yes/no):yes
Enter the table name to update records: Student_record
Enter the student register number that has to be modified: 123003
Record exists with the register number 123003
('Name', 'varchar(25)', 'YES', '', None, '')
('Reg_no', 'varchar(10)', 'YES', '', None, '')
('Date_of_birth', 'varchar(10)', 'YES', '', None, '')
('Lang1_mark', 'int', 'YES', '', None, '')
('Lang2_mark', 'int', 'YES', '', None, '')
('Maths_mark', 'int', 'YES', '', None, '')
('Science_mark', 'int', 'YES', '', None, '')
('Social_mark', 'int', 'YES', '', None, '')
('Total', 'int', 'YES', '', None, '')
('Percentage', 'float', 'YES', '', None, '')
('Result', 'varchar(10)', 'YES', '', None, '')
('Gender', 'varchar(10)', 'YES', '', None, '')
Enter the column name that has to be modified:Name
Enter the new value of that column:Cathy
Updated the record successfully
Do you want to update the record?(yes/no):no
```

```
Do you want to display the record?(yes/no):yes
Enter the table name to update records: Student_record
Total number of records: 8

Student Name:      Bala sundar
Student Register no: 123001
Date of birth:       24-08-2003

Marks Obtained
Language 1      :  76
Language 2      :  65
Maths           :  54
Science         :  43
Social Science  :  69
Total           :  307
Percentage      :  61.4
Final Result    :  Pass

Student Name:      Brinda
Student Register no: 123002
Date of birth:       25-08-2003

Marks Obtained
Language 1      :  35
Language 2      :  76
Maths           :  87
Science         :  90
Social Science  :  82
Total           :  370
Percentage      :  74.0
Final Result    :  Pass
```

```
Student Name:      Cathy
Student Register no: 123003
Date of birth:       17-07-2003

Marks Obtained
Language 1      :  78
Language 2      :  85
Maths           :  37
Science         :  52
Social Science  :  30
Total           :  282
Percentage      :  56.4
Final Result    :  Fail

Student Name:      Dinesh
Student Register no: 123004
Date of birth:       09-09-2003

Marks Obtained
Language 1      :  43
Language 2      :  78
Maths           :  98
Science         :  76
Social Science  :  84
Total           :  379
Percentage      :  75.8
Final Result    :  Pass

Student Name:      Fathima
Student Register no: 123005
Date of birth:       30-12-2003

Student Name:      Fathima
Student Register no: 123005
Date of birth:       30-12-2003

Marks Obtained
Language 1      :  79
Language 2      :  65
Maths           :  54
Science         :  43
Social Science  :  31
Total           :  272
Percentage      :  54.4
Final Result    :  Fail

Student Name:      Girish
Student Register no: 123006
Date of birth:       12-09-2003

Marks Obtained
Language 1      :  84
Language 2      :  87
Maths           :  65
Science         :  66
Social Science  :  31
Total           :  333
Percentage      :  66.6
Final Result    :  Fail

Student Name:      Harsha
Student Register no: 123007
Date of birth:       16-04-2003

Percentage      :  66.6
Final Result    :  Fail

Student Name:      Harsha
Student Register no: 123007
Date of birth:       16-04-2003

Marks Obtained
Language 1      :  76
Language 2      :  65
Maths           :  43
Science         :  39
Social Science  :  81
Total           :  304
Percentage      :  60.8
Final Result    :  Pass

Student Name:      Iniya
Student Register no: 123008
Date of birth:       04-05-2003

Marks Obtained
Language 1      :  78
Language 2      :  65
Maths           :  49
Science         :  85
Social Science  :  77
Total           :  354
Percentage      :  70.8
Final Result    :  Pass
Do you want to display the record?(yes/no):no
Do you want to search for a record? (yes/no):
```





Date:25/07/2024 **Day: Thursday**

Task: Develop a Python script to manage JSON file operations. The script should allow users to read and print specific fields from a JSON file, handling errors like file not found. Additionally, it should offer an option to write data to the JSON file.

Program code:

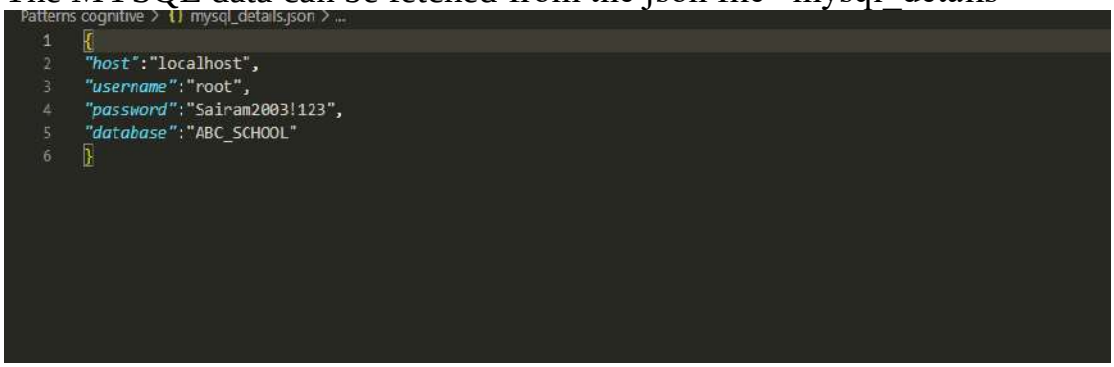
```python
import json
import os

#to know the current directory that is working
print(f"Current working directory: {os.getcwd()}")

def read_data():
    try:
        file_name=input("Enter the file name:")
        with open (file_name,"r") as file:

            #loading the contents of the file into data
            data=json.load(file)

            #to read the json file and print the contents of the file
            print(data)

            #accessing each elements of the json file
            print(f"Name : {data["name"]}")
            print(f"Age: {data["age"]}")
            print(f"Is student: {data["isstudent"]}")
            print(f"Courses: {data["courses"]}")
            print(f"City: {data["city"]}")

    except FileNotFoundError:
        print(f"An error occured ")

def main():
    while True:
        action=input("Do you want to read or write in the json file?(read/write/exit)")
        if action.lower()=="read":
            read_data()
        elif action.lower()=="write":
            write_data()

main()
```

Input and output:

```
Current working directory: C:\Users\sh\Desktop\Patterns cognitive
Do you want to read or write in the json file?(read/exit)read
Enter the file name:data.json
{'name': 'Lokkshanaa', 'age': 20, 'isstudent': 'True', 'courses': ['Maths', 'Science', 'Literature'], 'city': 'Chennai'}
Name : Lokkshanaa
Age: 20
Is student: True
Courses: ['Maths', 'Science', 'Literature']
City: Chennai
Do you want to read or write in the json file?(read/exit)exit
PS C:\Users\sh\Desktop\Patterns cognitive>
```

```python
#connection to mysql
try:
    with open ("mysql_details.json","r") as json_file:
        data=json.load(json_file)
        mydb=mysql.connector.connect(
            host=data["host"],
            username=data["username"],
            password=data["password"],
            database=data["database"]
    )
except mysql.connector.Error as e:
    print(f"An error occured {e}")
```

The MYSQL data can be fetched from the json file “mysql_details”



Date:26/07/2024 **Day: Friday**

Task: Create a Python script using the cv2 module to perform various image processing operations, including converting an image to grayscale, rotating it to the right or left, and resizing it based on user input. The script should handle errors gracefully and display the original and processed images. Implement a user-driven menu to select the desired operation and ensure proper handling of image loading and manipulation.

Program code:

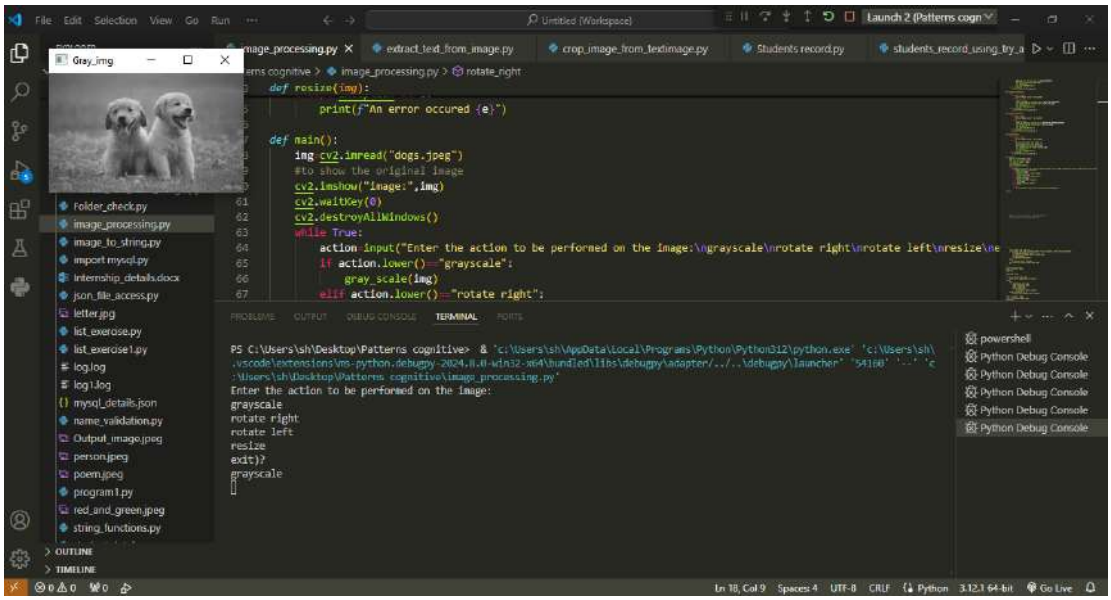
```python
#using cv2 module
import cv2

def gray_scale(img):
    try:
        if img is None:
            print("Image couldn't be loaded")
        else:
            #grayscaling an image
            gray_img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            cv2.imshow("Gray_img",gray_img)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
    except Exception as e:
        print(f"An error occured {e}")

def rotate_right(img):
    try:
        if img is None:
            print("Image couldn't be loaded")
        else:
            #rotate right
            right_rot_imag=cv2.rotate(img,cv2.ROTATE_90_CLOCKWISE)
            cv2.imshow("image rotated right",right_rot_imag)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
    except Exception as e:
        print(f"An error occured {e}")
```

```
def rotate_left(img):
    try:
        if img is None:
            print("Image couldn't be loaded")
        else:
            #rotate left
            left_rot_imag=cv2.rotate(img,cv2.ROTATE_90_COUNTERCLOCKWISE)
            cv2.imshow("image rotated left",left_rot_imag)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
    except Exception as e:
        print(f"An error occured {e}")

def resize(img):
    try:
        if img is None:
            print("Image couldn't be loaded")
        else:
            w=int(input("Enter the width:"))
            h=int(input("Enter the height:"))
            resize_imag=cv2.resize(img,(w,h))
            cv2.imshow("resized image",resize_imag)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
    except Exception as e:
        print(f"An error occured {e}")

def main():
    img=cv2.imread("dogs.jpeg")
    #to show the original image
    cv2.imshow("image:",img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    while True:
        action=input("Enter the action to be performed on the image:\ngrayscale\nrotate right\nrotate
left\nresize\nexit)?\n")
        if action.lower()=="grayscale":
            gray_scale(img)
        elif action.lower()=="rotate right":
            rotate_right(img)
        elif action.lower()=="rotate left":
            rotate_left(img)
        elif action.lower()=="resize":
            resize(img)
        elif action.lower()=="exit":
            break
        else:
            print("Invalid action to be performed on the image.")

main()
```

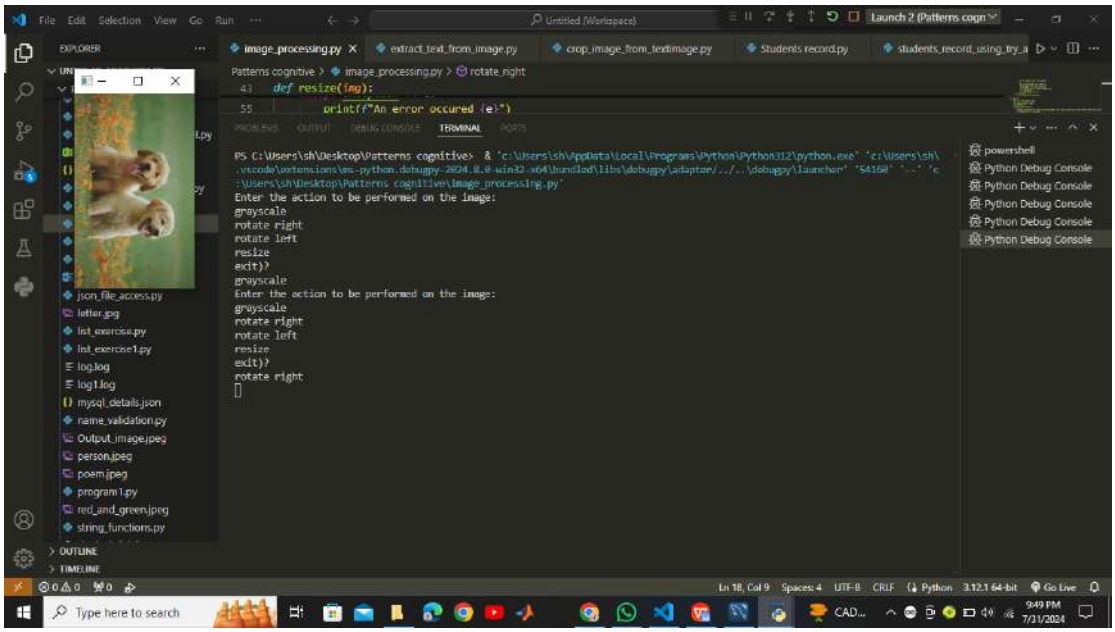
Input and output:

Displaying the original image

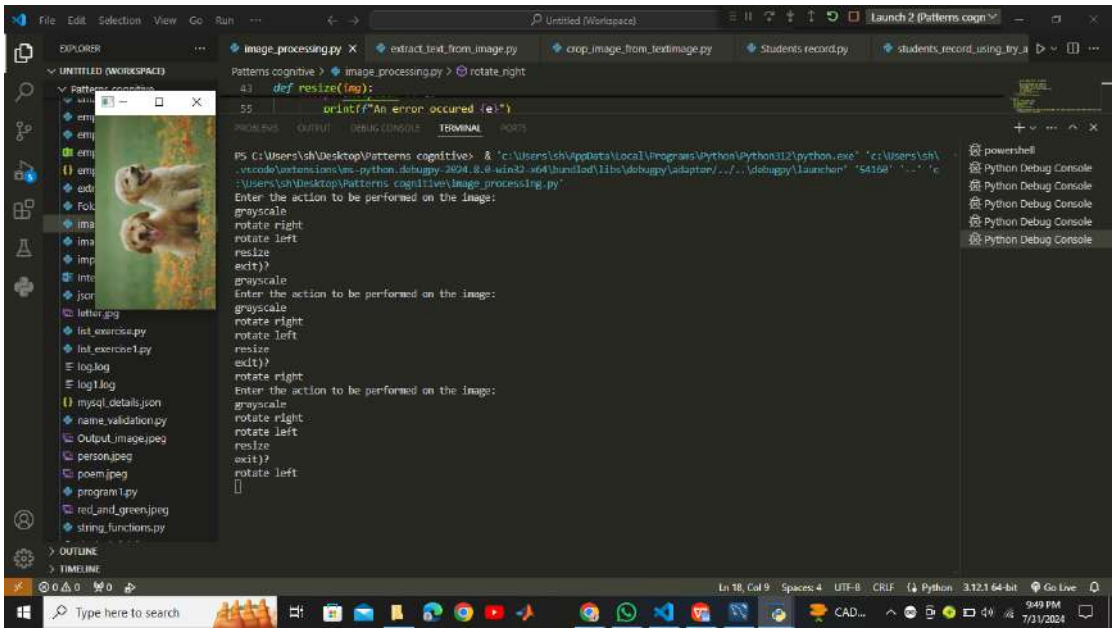
Grayscaling an image:



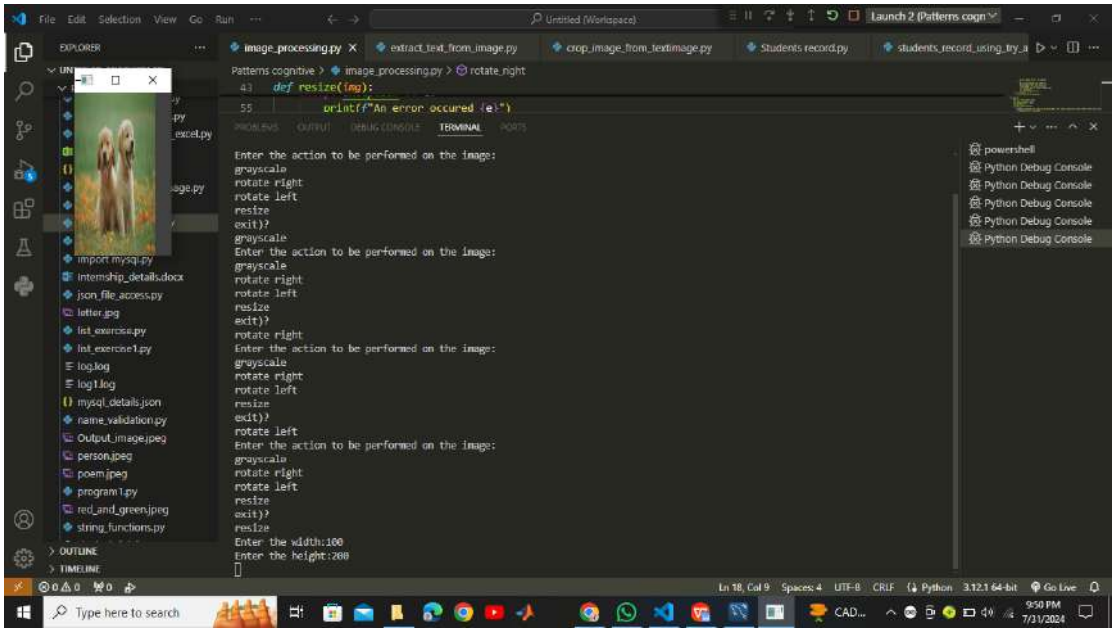
Rotating right the image:



Rotating left the image:



Resizing an image:



Task: Develop a Python program using the cv2 module to perform various image processing operations on an input image. The operations should include displaying a blurred version of the image, applying different thresholding techniques, calculating and displaying intensity statistics, generating and plotting a grayscale histogram, adding two images, and cropping a specified region of the image. Ensure each operation is correctly implemented and handles potential errors gracefully.

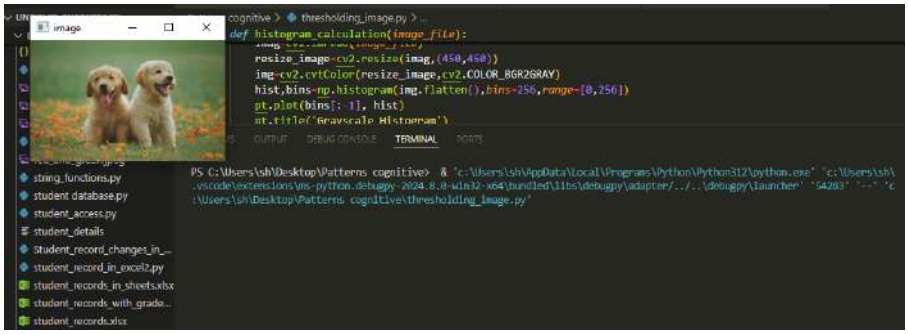
Program code:

```
import cv2
import numpy as np
from matplotlib import pyplot as pt

image_file="dogs.jpeg"
#imag=cv2.imread(image_file)

def show_image(image_file):
    img=cv2.imread(image_file)
    cv2.imshow("image",img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    blurred_img = cv2.GaussianBlur(img, (5, 5), 0)
    cv2.imshow("Blurred Image", blurred_img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


def threshing_image(image_file):

    img=cv2.imread(image_file,cv2.IMREAD_GRAYSCALE)
    ret,thresh_image=cv2.threshold(img,127,255,cv2.THRESH_BINARY)
    cv2.imshow("Threshed image",thresh_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    ret,thresh_image_inv=cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
    cv2.imshow("Threshed image",thresh_image_inv)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    ret,thresh_image_trunc=cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
    cv2.imshow("Threshed image",thresh_image_trunc)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    ret,thresh_image_tozero=cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
    cv2.imshow("Threshed image",thresh_image_tozero)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    ret,thresh_image_tozeroinv=cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)
    cv2.imshow("Threshed image",thresh_image_tozeroinv)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

```
def intensity_calculation(image_file):

    img=cv2.imread(image_file,cv2.IMREAD_GRAYSCALE)
    #calculaing the average using numpy module
    average_intensity=np.mean(img)

    #limiting decimals to 2 points using round function
    print(f"Average intensity: {round(average_intensity,2)}")
    #limiting decimals to 2 points using .2f
    print(f"Average intensity: {average_intensity:.2f}")

    #maximum intensity of the image
    max_intensity=np.max(img)
    print(f"Maximum intensity:{round(max_intensity,2)}")

    #minimum intensity of the image
    min_intensity=np.min(img)
    print(f"Minimum intensity:{round(min_intensity,2)}")

    #standard deviation of the image
    std=np.std(img)
    print(f"Standard deviation: {std}")

def histogram_calculation(image_file):

    imag=cv2.imread(image_file)
    resize_image=cv2.resize(imag,(450,450))
    img=cv2.cvtColor(resize_image,cv2.COLOR_BGR2GRAY)
    hist,bins=np.histogram(img.flatten(),bins=256,range=[0,256])
    pt.plot(bins[:-1], hist)
    pt.title('Grayscale Histogram')
    pt.xlabel('Pixel Intensity')
    pt.ylabel('Frequency')
    pt.show()

def add_images():
    image_file1="dogs.jpeg"
    image_file2="person.jpeg"
    image1=cv2.imread(image_file1)
    image2=cv2.imread(image_file2)
    res1=cv2.resize(image1,(300,300))
    res2=cv2.resize(image2,(300,300))

    added_image=cv2.add(res1,res2)
    cv2.imshow("Added",added_image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

def crop_image():
    image_file="dogs.jpeg"
    img=cv2.imread(image_file)
    print("shape of the image",img.shape)
    crop=img[50:100,80:130]
    cv2.imshow("cropped",crop)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

show_image(image_file)
threshing_image(image_file)
intensity_calculation(image_file)
histogram_calculation(image_file)
add_images()
crop_image()
```

Input and output:

Displaying an original image

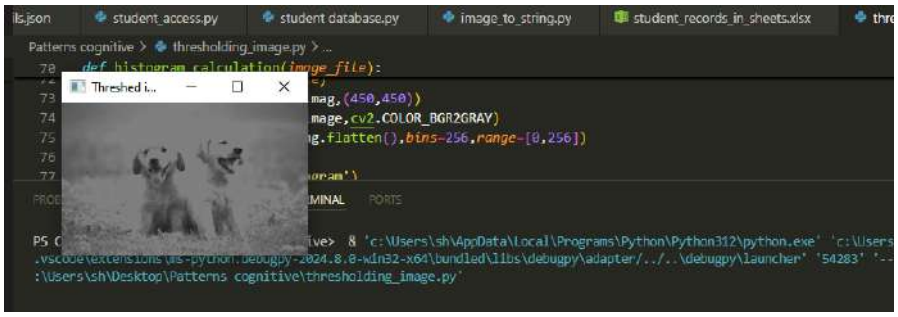


Blurring the image

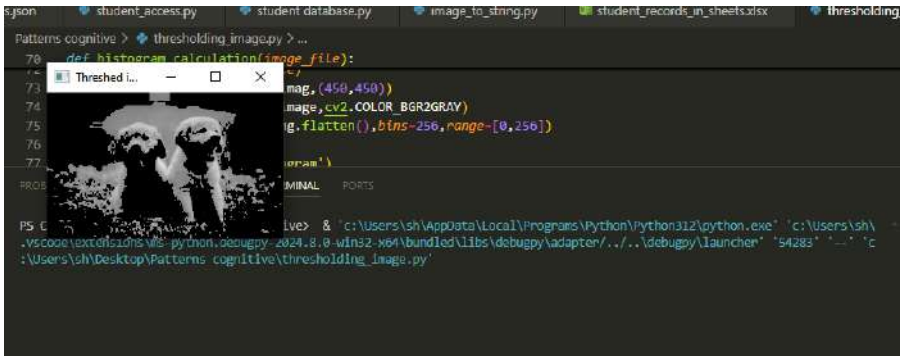
Thresholding an image using THRESH_BINARY

Thresholding an image using THRESH_BINARY_INV

Thresholding an image using THRESH_TRUNC



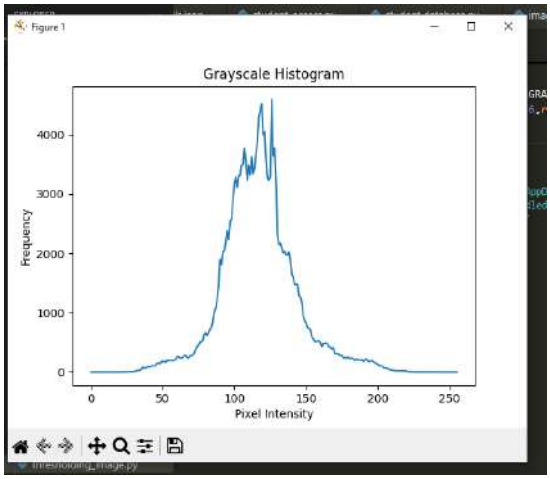
Thresholding an image using THRESH_TOZERO



Thresholding an image using THRESH_TOZERO_INV

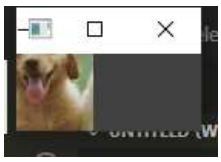
Intensity calculation of the image

Histogram calculation of the image



Adding two images together

Cropping the image



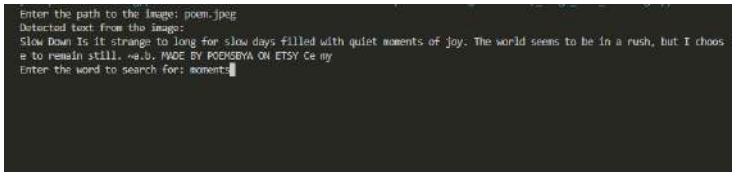
Date:29/07/2024 **Day: Monday**

Task: Create a Python program using the cv2 and pytesseract modules to extract text from an image and allow the user to crop a specific word from the image. The program should read an image file, detect and display all text found in the image, and enable the user to input a word to search for. If the word is found, the program should crop the region containing the word, display the cropped image, and save it to disk. Ensure the program handles potential errors such as missing images or words not found in the text.

Program code:

```
import cv2
import pytesseract
from pytesseract import Output
import os
import matplotlib.pyplot as plt

pytesseract.pytesseract.tesseract_cmd = r'C:\Users\sh\Desktop\Patterns cognitive\Tesseract
python\tesseract.exe'

def show_text_from_image(image_path):
    # Read the image
    image = cv2.imread(image_path)
    if image is None:
        print(f"Error: Could not open or find the image {image_path}")
        return None, None

    # Convert the image to grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Use pytesseract to get data from the image
    data = pytesseract.image_to_data(gray_image, output_type=Output.DICT)

    # Print all detected text
    print("Detected text from the image:")
    for i in range(len(data['text'])):
        if int(data['conf'][i]) > 0:  # Confidence level filter to ignore low-confidence words
            print(data['text'][i], end=' ')
    print()  # Newline for better readability

    return image, data

def crop_word_from_image(image, data, word):
    # Iterate through all detected text
    for i in range(len(data['text'])):
        if word.lower() in data['text'][i].lower():
            x, y, w, h = data['left'][i], data['top'][i], data['width'][i], data['height'][i]
            cropped_image = image[y:y+h, x:x+w]

            # Display the cropped image
            plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
            plt.title(f'Cropped image containing: {word}')
            plt.axis('off')
            plt.show()

            # Save the cropped image
            cropped_image_path = f"cropped_{word}.png"
            cv2.imwrite(cropped_image_path, cropped_image)

            print(f"Cropped image saved as {cropped_image_path}")
            return

    print(f"Word '{word}' not found in the image.")

def main():
    image_path = input("Enter the path to the image: ")
    image, data = show_text_from_image(image_path)

    if image is not None and data is not None:
        word = input("Enter the word to search for: ")
        crop_word_from_image(image, data, word)
main()
```

Input and output:



Date:30/07/2024 and 31/07/2024
Day:Tuesday and Wednesday

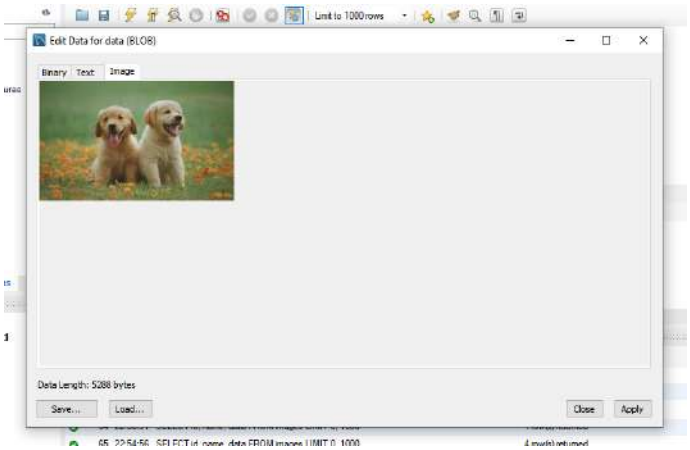
Task: Develop a Python program to store and retrieve images in a MySQL database. The program should convert an image file to binary data and insert it into a BLOB column in a MySQL table. After inserting the image, it should also include functionality to fetch the image from the database, save it to a file, and open the image for viewing.

Program code:

```python
import mysql.connector

def convert_to_binary_data(filename):
    # Convert digital data to binary format
    with open(filename, 'rb') as file:
        binary_data = file.read()
    return binary_data

def insert_image(name, photo):
    try:
        connection = mysql.connector.connect(
            host="localhost",
            username="root",
            password="Sairam2003!123",
            database="saveimage"
        )

        cursor = connection.cursor()
        sql_insert_blob_query = """ INSERT INTO images (name, data) VALUES (%s,%s)"""

        binary_data = convert_to_binary_data(photo)

        # Convert data into tuple format
        insert_blob_tuple = (name, binary_data)
        result = cursor.execute(sql_insert_blob_query, insert_blob_tuple)
        connection.commit()
        print("Image inserted successfully as a BLOB into images table")

    except mysql.connector.Error as error:
        print("Failed inserting BLOB data into MySQL table {}".format(error))

    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("MySQL connection is closed")

insert_image("dogs.jpeg", "Dogs.jpeg")
```

Input and output:

Query to display the image from MYSQL :


Limit to 1000 rows
Edit Data for data (BLOB)
Binary
Text
Image
Data Length: 5288 bytes
Save...
Load...
Close
Apply