Experiment No. 1

Aim: Calculate factorial of a number inputted by user.

Code

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

n = int(input("Enter a number: "))
print(factorial(n))
```

Output

```
#Run 1

Enter a number: 0

1


#Run 2

Enter a number: 69

171122452428141311372468338881272839092270544893520369393648040923257279754
140647424000000000000000


#Run 3

Enter a number: 420

1179832395293178259148587778443982767423908163629667689799210969550884231351169347
8047667995005102940503883496965320847293740875333842040193228929611788194646981212
6353301268533527300429478938265247732446542700170132623014591146631602964471437174
8823861128004214806081770714277374544632880180009063325310867611466814559562175609
4143401774174785802909812926615867007680755447883602420534368994391860098591471476
5387864406466779970942769373120803592028405220313102208368842580526563153497848176
1954009800546844281261649619610291306374918025956972209823833523561696079181976208
7836628182356136151492963439310892952344021300432534898269280971992110743409299161
6162585470522759556509074096211379330874264959860396374796094106383547466430697189
2700806057422478626083960243385932102946293048920279760860198799159782580284293120
0000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000
```

Experiment No. 2

Aim: Check whether the number inputted by user is a prime number or not.

Code

```
def check_prime():
    num = int(input("Enter a number: "))
    if num > 1:
        for i in range(2, num):
            if (num % i) == 0:
                print(num, "is not a prime number")
                break
        else:
            print(num, "is a prime number")
    else:
        print(num, "is not a prime number")


check_prime()
```

Output

```
#Run 1

Enter a number: 0

0 is not a prime number


#Run 2

Enter a number: 1

1 is not a prime number


#Run 3

Enter a number: 5

5 is a prime number


#Run 4

Enter a number: 15

15 is not a prime number
```

Experiment No. 3

Aim: Search for the existence of a substring within an inputted string.

Code

```
def search_in_string():
    s = input("Enter a string: ")
    t = input("Enter a substring to search in string:
")
    if t in s:
        print("Substring found")
        print("The number of times the substring is
present in the string is", s.count(t))
    else:
        print("Substring not found")


search_in_string()
```

Output

```
#Run 1
Enter a string: Hello World
Enter a substring to search in string: hello
Substring not found

#Run 2
Enter a string: Hello World
Enter a substring to search in string: Hello
Substring found
The number of times the substring is present in the string is 1
```

Experiment No. 4

Aim: Perform Bubble Sort using a user-defined function.

Code

```
def BubbleSort():
    L = [eval(i) for i in input("Enter the list items :
").split()]
    for i in range(len(L)):
        for j in range(len(L)-1):
            if L[j] > L[j+1]:
                L[j], L[j+1] = L[j+1], L[j]
    return L


print(BubbleSort())
```

Output

```
#Run 1
Enter the list items : 10 5 7 8 2 4 6 1 3 9
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]


#Run 2
Enter the list items : 'Shehzad' 'Abdullahir' 'Taha' 'Farhan' 'Abhinav'
['Abdullahir', 'Abhinav', 'Farhan', 'Shehzad', 'Taha']
```

Experiment No. 5

Aim: Display file content line by line, with each word separated by '#".

Code

```
def split():
    f = open("data.txt", "r")
    for line in f.readlines():
        print(line.replace(' ', '#'), end='')
    f.close()


split()
```

File Content

```
Hello World

This is a new sentence
```

Output

```
Hello#World

This#is#a#new#sentence
```

Experiment No. 6

Aim: Count and display the number of vowels, consonants, lowercase and uppercase characters in a file.

Code

```
def count_vowels():
    vowels = 0
    for line in L:
        for char in line:
            if char.lower() in 'aeiou':
                vowels += 1
    print("The number of vowels in the file is",
vowels)


def count_consonants():
    consonants = 0
    for line in L:
        for char in line:
            if char.isalpha() and char.lower() not in
'aeiou':
                consonants += 1
    print("The number of consonants in the file is",
consonants)


def count_lowercase():
    lowercase = 0
    for line in L:
        for char in line:
            if char.islower():
                lowercase += 1
    print("The number of lowercase letters in the file
is", lowercase)
```

```
def count_uppercase():
    uppercase = 0
    for line in L:
        for char in line:
            if char.isupper():
                uppercase += 1
    print("The number of uppercase letters in the file
is", uppercase)


f = open("data.txt", "r")
L = f.readlines()
f.close()

count_vowels()
count_consonants()
count_lowercase()
count_uppercase()
```

File Content

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent in placerat tortor, vitae hendrerit mauris. Vestibulum tempor non enim vitae tempor. Quisque nec varius dui, ut dapibus elit. Sed nec gravida libero. Integer euismod purus quis odio sagittis elementum. Mauris id placerat odio, sed mattis nulla. Duis eleifend eget ligula eu rhoncus. Mauris facilisis euismod quam vel rhoncus.

Output

```
The number of vowels in the file is 142

The number of consonants in the file is 179

The number of lowercase letters in the file is 312

The number of uppercase letters in the file is 9
```

Experiment No. 7

Aim: Create a binary file to store Roll Number and Name and perform search operations on it.

Code

```
import pickle
L = []


def add_student():
    f = open("student.dat", "wb+")
    while True:
        roll = int(input("Enter roll number: "))
        name = input("Enter name: ")
        L.append([roll, name])
        if input("Would you like to add another
student? (y/n): ") != "y":
            break
        else:
            continue
    print()
    pickle.dump(L, f)
    f.close()


def search_student():
    roll = int(input("Enter roll number to search: "))
    f = open("student.dat", "rb")
    c = pickle.load(f)
    for i in c:
        if roll == i[0]:
            print("Roll number:", i[0], "Name:", i[1],
'\n')
            break
    else:
        print("Roll number not found")
    f.close()
```

```
while True:
    print("Menu")
    print("1. Add student\n2. Search student\n3. Exit")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        add_student()
    elif choice == 2:
        search_student()
    elif choice == 3:
        exit()
```

Output

```
Menu
1. Add student
2. Search student
3. Exit
Enter your choice: 1
Enter roll number: 001
Enter name: Taha
Would you like to add another student? (y/n): y
Enter roll number: 102
Enter name: Shehzad
Would you like to add another student? (y/n): n

Menu
1. Add student
2. Search student
3. Exit
Enter your choice: 2
Enter roll number to search: 102
Roll number: 102 Name: Shehzad
```

```
Menu
1. Add student
2. Search student
3. Exit
Enter your choice: 2
Enter roll number to search: 103
Roll number not found
Menu
1. Add student
2. Search student
3. Exit
Enter your choice: 3
```

Experiment No. 8

Aim: Create a binary file to store Roll Number and Name of student, and to perform search operations on it.

Code

```
import pickle
L = []

def add_student():
    f = open("student.dat", "wb+")
    while True:
        roll = int(input("Enter roll number: "))
        name = input("Enter name: ")
        marks = int(input("Enter marks: "))
        L.append([roll, name, marks])
        if input("Would you like to add another
student? (y/n): ") != "y":
            break
        else:
            continue
    print()
    pickle.dump(L, f)
    f.close()


def search_student():
    roll = int(input("Enter roll number to search: "))
    f = open("student.dat", "rb")
    c = pickle.load(f)
    for i in c:
        if roll == i[0]:
            print("Roll number:", i[0], "Name:", i[1],
"Marks: ", i[2], '\n')
            break
    else:
        print("Roll number not found")
    f.close()
```

```
def edit_student():
    roll = int(input("Enter roll number to edit: "))
    mark = int(input("Enter new marks: "))
    f = open("student.dat", "rb")
    c = pickle.load(f)
    f.close()
    for i in c:
        if roll == i[0]:
            i[2] = mark
            f = open("student.dat", "wb")
            pickle.dump(c, f)
            f.close()
            break
    else:
        print("Roll number not found")
    print("Marks updated")


while True:
    print("Menu")
    print("1. Add student\n2. Search student\n3. Edit
student\n4. Exit")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        add_student()
    elif choice == 2:
        search_student()
    elif choice == 3:
        edit_student()
    elif choice == 4:
        exit()
```

Output

```
Menu
1. Add student
2. Search student
3. Edit student
4. Exit
Enter your choice: 1
Enter roll number: 001
Enter name: Taha
Enter marks: 99
Would you like to add another student? (y/n): y
Enter roll number: 102
Enter name: Shehzad
Enter marks: 95
Would you like to add another student? (y/n): n

Menu
1. Add student
2. Search student
3. Edit student
4. Exit
Enter your choice: 2
Enter roll number to search: 001
Roll number: 1 Name: Taha Marks:  99

Menu
1. Add student
2. Search student
3. Edit student
4. Exit
Enter your choice: 3
Enter roll number to edit: 102
Enter new marks: 99
```

```
Marks updated
Menu
1. Add student
2. Search student
3. Edit student
4. Exit
Enter your choice: 2
Enter roll number to search: 102
Roll number: 102 Name: Shehzad Marks:  99

Menu
1. Add student
2. Search student
3. Edit student
4. Exit
Enter your choice: 4
```

Experiment No. 9

Aim: Copy a file into a new file except for lines containing the letter 'a'.

Code

```
f1 = open("file1.txt", "r")
f2 = open("file2.txt", "w")
c = f1.readlines()
L = []
for i in c:
    if 'a' not in i.lower():
        L.append(i)
f2.writelines(L)
f1.close()
f2.close()
```

File Content

File1.txt

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Maecenas sed diam efficitur, pretium nulla eu, fermentum purus.

Nullam lobortis est a fermentum hendrerit.

Quisque efficitur, purus at consequat iaculis, metus purus euismod quam, at imperdiet felis metus at dolor.

Nulla porta tellus urna, non tincidunt diam bibendum at.

Quisque ac risus condimentum, suscipit leo aliquet, efficitur sapien.

Sed eleifend orci ac est placerat tempus.

Phasellus accumsan ullamcorper elementum.

Quisque sed ante mattis lacus porttitor interdum.

Nunc id magna nec dolor condimentum euismod.

Hello World

File2.txt

Hello World

Experiment No. 10

Aim: Create a CSV file and store employee ID, name and salary. Perform search operations.

Code

```
import csv
L = []
f = open('data.csv', 'w', newline='')
writer = csv.writer(f, delimiter=',')
writer.writerow(['Employee ID', 'Employee Name', 'Employee
Salary'])
f.close()

def create():
    f = open('data.csv', 'a+', newline='')
    writer = csv.writer(f, delimiter=',')
    while True:
        idno = input("Enter Employee ID: ")
        name = input("Enter Employee Name: ")
        salr = input("Enter Employee Salary: ")
        L.append([idno, name, salr])
        if input("Do you want to continue? (y/n):
").lower() != 'y':
            break
    writer.writerows(L)
    f.close()


def search():
    f = open('data.csv', 'r')
    reader = csv.reader(f, delimiter=',')
    idno = input("Enter Employee ID: ")
    for row in reader:
        if idno == row[0]:
            print("Employee ID: ", row[0], "\nEmployee Name: ",
row[1], "\nEmployee Salary: ", row[2])
            break
    f.close()
```

```
while True:
    print("Menu")
    print("1. Create\n2. Search\n3. Exit")

    ch = int(input("Enter your choice: "))
    if ch == 1:
        create()
    elif ch == 2:
        search()
    elif ch == 3:
        exit()
    else:
        print("Invalid choice")
```

Output

```
Menu
1. Create
2. Search
3. Exit
Enter your choice: 1
Enter Employee ID: 101
Enter Employee Name: Bob
Enter Employee Salary: 1000
Do you want to continue? (y/n): y
Enter Employee ID: 102
Enter Employee Name: Thomas
Enter Employee Salary: 1100
Do you want to continue? (y/n): n
Menu
1. Create
2. Search
3. Exit
Enter your choice: 2
```

```
Enter Employee ID: 101
Employee ID:  101
Employee Name:  Bob
Employee Salary:  1000
Menu
1. Create
2. Search
3. Exit
Enter your choice: 2
Enter Employee ID: 103
Menu
1. Create
2. Search
3. Exit
Enter your choice: 3
```

Experiment No. 11

Aim: Simulate a dice.

Code

```
import random
import time


def simulate_dice():
    try:
        print("Press Ctrl+C to stop the dice")
        while True:
            n = random.randint(1, 6)
            print(n, end=' ')  # This statement can be
commented out to keep the terminal clean
            time.sleep(0.001)
    except KeyboardInterrupt:
        print("\nDice stopped")
        print("Your number is", n)


simulate_dice()
```

Output

```
Press Ctrl+C to stop the dice

2 5 3 1 1 6 4 5 5 5 3 1 3 1 6 4 3 5 4 2 2 3 1 4 4 3 5 1 2 4 4 2 6 1 6 1 6 3
3 5 3 4 2 3 2 3 4 6 1 2 2 6 1 1 3 5 5 2 1 2 6 5 2 3 5 5 5 2 4 2 3 3 6 2 2 2
4 3 1 4 3 2 2 1 4 3 3 2 2 5 6 4 6 4 4 6 4 1 6 5 2 4 6 1 1 5 5 6 1 3 6 5 1 5
2 1 4 5 6 5 3 6 2 3 1 5 6 1 2 5 2 1 2 3 3

Dice stopped

Your number is 3
```

Experiment No. 12

Aim: Implement a stack in Python using lists.

Code

```
S = []


def isEmpty(S):
    return True if len(S) == 0 else False


def push(S, x):
    S.append(x)
    top = len(S) - 1


def pop(S):
    return "Underflow" if isEmpty(S) else S.pop()


def peek(S):
    return "Underflow" if isEmpty(S) else S[-1]


def show(S):
    print("No elements in the stack") if isEmpty(S)
else None
    t = len(S) - 1
    while t >= 0:
        print(S[t], end=' ')
        t -= 1
    print()
```

```
while True:
    print("Menu")
    print("1. Push\n2. Pop\n3. Peek\n4. Show\n5. Exit")
    c = int(input("Enter your choice: "))
    if c == 1:
        x = input("Enter the element to be pushed: ")
        push(S, x)
    elif c == 2:
        print(pop(S))
    elif c == 3:
        print(peek(S))
    elif c == 4:
        show(S)
    elif c == 5:
        exit()
    else:
        print("Invalid choice")
```

Output

```
Menu
1. Push
2. Pop
3. Peek
4. Show
5. Exit
Enter your choice: 1
Enter the element to be pushed: 1
Menu
1. Push
2. Pop
3. Peek
4. Show
5. Exit
Enter your choice: 1
```

```
Enter the element to be pushed: 3
Menu
1. Push
2. Pop
3. Peek
4. Show
5. Exit
Enter your choice: 1
Enter the element to be pushed: 5
Menu
1. Push
2. Pop
3. Peek
4. Show
5. Exit
Enter your choice: 2
5
Menu
1. Push
2. Pop
3. Peek
4. Show
5. Exit
Enter your choice: 3
3
Menu
1. Push
2. Pop
3. Peek
4. Show
5. Exit
Enter your choice: 4
3 1
```

```
Menu
1. Push
2. Pop
3. Peek
4. Show
5. Exit
Enter your choice: 5
```

Experiment No. 13

Aim: Linear and Binary Search using lists.

Code

```
def LinearSearch():
    L = [eval(i) for i in input("Enter the list items :
").split()]
    c = eval(input("Enter element to search : "))
    for i in range(len(L)):
        if L[i] == c:
            print("Element found at index", i)
            break
    else:
        print("Element not found")


def BinarySearch():
    L = [eval(i) for i in input("Enter the list items :
").split()]
    c = eval(input("Enter element to search : "))
    L.sort()
    low = found = 0
    high = len(L) - 1
    while low <= high:
        mid = (low + high) // 2
        if L[mid] == c:
            found = 1
            break
        elif L[mid] > c:
            high = mid - 1
        else:
            low = mid + 1
    if found == 1:
        print("Element found")
    else:
        print("Element not found")
```

```
LinearSearch()
BinarySearch()
```

Output

```
Enter the list items : 1 2 4 5 7 8 9 10 3 5 6

Enter element to search : 11

Element not found

Enter the list items : 1 2 4 5 7 8 9 10 3 5 6

Enter element to search : 1

Element found
```

Experiment No. 14

Aim: Find the most common domain in phishing emails.

Code

```
L = []
d = {}


def findmostoccuringdomain():
    while True:
        c = input("Enter phishing emails (enter q to
exit loop): ")
        if c == "q":
            break
        else:
            L.append(c)
    for i in range(len(L)):
        L[i] = L[i].split("@")[1]

    for i in L:
        d[i] = d.get(i, 0) + 1

    max = 0
    for i in d:
        if d[i] > max:
            max = d[i]
            domain = i
    print("\nMost occuring domain is", domain, "with",
max, "occurences")


findmostoccuringdomain()
```

Output

```
Enter phishing emails (enter q to exit loop): jackpotwin@lottery.com
Enter phishing emails (enter q to exit loop): claimtheprize@mylife.com
Enter phishing emails (enter q to exit loop): youarethewinner@lottery.com
Enter phishing emails (enter q to exit loop): luckywinner@mylife.com
Enter phishing emails (enter q to exit loop): spinthewheel@flipkart.com
Enter phishing emails (enter q to exit loop): dealwinner@snapdeal.com
Enter phishing emails (enter q to exit loop): luckywinner@snapdeal.com
Enter phishing emails (enter q to exit loop):
luckyjackpot@americanlottery.com
Enter phishing emails (enter q to exit loop):
claimtheprize@lootolottery.com
Enter phishing emails (enter q to exit loop): youarelucky@mylife.com
Enter phishing emails (enter q to exit loop): q


Most occuring domain is mylife.com with 3 occurences
```

Experiment No. 15

Aim: Connect with a database and store employee details and display them.

Code

```
from mysql.connector import connect as c
cdb = c(user='root', password='root', host='localhost')
db = cdb.cursor()

db.execute("CREATE DATABASE IF NOT EXISTS company")
db.execute("USE company")
db.execute("CREATE TABLE IF NOT EXISTS employee (id INT
PRIMARY KEY, name VARCHAR(255), salary INT, department
VARCHAR(255))")
cdb.commit()


def add_records():
    while True:
        id = int(input("Enter ID: "))
        name = input("Enter Name: ")
        salary = int(input("Enter Salary: "))
        department = input("Enter Department: ")
        db.execute("INSERT INTO employee VALUES (%s,
%s, %s, %s)", (id, name, salary, department))
        cdb.commit()
        print("Record Added Successfully")
        ch = input("Do you want to add more records?
(y/n): ")
        if ch != 'y':
            break
```

```
def show_records():
    db.execute("SELECT * FROM employee")
    rs = db.fetchall()
    print("%10s" % "Employee ID", "%20s" % "Employee
Name", "%10s" % "Salary", "%20s" % "Department")
    for i in rs:
        print("%7s" % i[0], "%20s" % i[1], "%12s" %
i[2], "%17s" % i[3])


while True:
    print("Menu")
    print("1. Add Records\n2. Show Records\n3. Exit")
    ch = int(input("Enter your choice: "))
    if ch == 1:
        add_records()
    elif ch == 2:
        show_records()
    elif ch == 3:
        exit()
```

Output

```
Menu
1. Add Records
2. Show Records
3. Exit
Enter your choice: 1
Enter ID: 101
Enter Name: Taha
Enter Salary: 10000
Enter Department: HR
Record Added Successfully
Do you want to add more records? (y/n): y
Enter ID: 102
Enter Name: Shehzad
Enter Salary: 10000
Enter Department: HR
Record Added Successfully
Do you want to add more records? (y/n): n
Menu
1. Add Records
2. Show Records
3. Exit
Enter your choice: 2
Employee ID        Employee Name     Salary           Department
    101                 Taha        10000               HR
    102               Shehzad       10000               HR
Menu
1. Add Records
2. Show Records
3. Exit
Enter your choice: 3
```

Experiment No. 16

Aim: Connect with a database and search employee details and display them.

Code

```
from mysql.connector import connect as c
cdb = c(user='root', password='root', host='localhost')
db = cdb.cursor()

db.execute("CREATE DATABASE IF NOT EXISTS company")
db.execute("USE company")
db.execute("CREATE TABLE IF NOT EXISTS employee (id INT
PRIMARY KEY, name VARCHAR(255), salary INT, department
VARCHAR(255))")
cdb.commit()


def search_records():
    id = int(input("Enter ID: "))
    db.execute("SELECT * FROM employee WHERE id = %s",
(id,))
    rs = db.fetchall()
    if len(rs) == 0:
        print("Employee not found")
    else:
        print("Employee ID: ", rs[0][0], "\nEmployee
Name: ", rs[0][1], "\nSalary: ", rs[0][2],
"\nDepartment: ", rs[0][3], "\n")


print("Employee Search")
search_records()
```

Output

```
#Run 1
Employee Search
Enter ID: 101
Employee ID:  101
Employee Name:  Taha
Salary:  10000
Department:  HR

#Run 2
Employee Search
Enter ID: 102
Employee ID:  102
Employee Name:  Shehzad
Salary:  10000
Department:  HR
```

Experiment No. 17

Aim: Connect with a database and update employee details.

Code

```
from mysql.connector import connect as c
cdb = c(user='root', password='root', host='localhost')
db = cdb.cursor()

db.execute("CREATE DATABASE IF NOT EXISTS company")
db.execute("USE company")
db.execute("CREATE TABLE IF NOT EXISTS employee (id INT
PRIMARY KEY, name VARCHAR(255), salary INT, department
VARCHAR(255))")
cdb.commit()


def edit_records():
    id = int(input("Enter ID: "))
    db.execute("SELECT * FROM employee WHERE id = %s",
(id,))
    rs = db.fetchall()
    if len(rs) == 0:
        print("Employee not found")
        exit()
    else:
        print("Employee ID: ", rs[0][0], "\nEmployee
Name: ", rs[0][1], "\nSalary: ", rs[0][2],
"\nDepartment: ", rs[0][3], "\n")
    name = input("Enter new name: ")
    salary = int(input("Enter new salary: "))
    department = input("Enter new department: ")
    db.execute("UPDATE employee SET name = %s, salary =
%s, department = %s WHERE id = %s", (name, salary,
department, id))
    cdb.commit()
    print("Record updated successfully")
print("Employee Updation")
```

```
edit_records()
```

Output

```
Employee Updation
Enter ID: 101
Employee ID:  101
Employee Name:  Taha
Salary:  10000
Department:  HR

Enter new name: Taha Parker
Enter new salary: 20000
Enter new department: IT
Record updated successfully
```

Experiment No. 18

Aim: Connect with a database and delete employee details.

Code

```
from mysql.connector import connect as c
cdb = c(user='root', password='root', host='localhost')
db = cdb.cursor()

db.execute("CREATE DATABASE IF NOT EXISTS company")
db.execute("USE company")
db.execute("CREATE TABLE IF NOT EXISTS employee (id INT
PRIMARY KEY, name VARCHAR(255), salary INT, department
VARCHAR(255))")
cdb.commit()

def delete_records():
    id = int(input("Enter ID: "))
    db.execute("SELECT * FROM employee WHERE id = %s",
(id,))
    rs = db.fetchall()
    if len(rs) == 0:
        print("Employee not found")
        exit()
    else:
        print("Employee ID: ", rs[0][0], "\nEmployee
Name: ", rs[0][1], "\nSalary: ", rs[0][2],
"\nDepartment: ", rs[0][3], "\n")
    ch = input("Do you want to delete this record?
(y/n): ")
    if ch == "y":
        db.execute("DELETE FROM employee WHERE id =
%s", (id,))
        cdb.commit()
        print("Employee deleted successfully")
    else:
        print("Employee not deleted")
        exit()
```

```
print("Employee Deletion")
delete_records()
```

Output

```
Employee Deletion
Enter ID: 102
Employee ID:  102
Employee Name:  Shehzad
Salary:  10000
Department:  HR

Do you want to delete this record? (y/n): y
Record deleted successfully
```

Experiment No. 19

Aim: Write a method CREATE() to create an EMP.csv file with the following details:

- id to store employee number of integer type
- name to store employee name of string type
- dept to store their respective department of string type
- basic to store basic salary of respective employee
- hra to be calculated from his/her basic salary which is 10% of basic
- sal to be calculated as salary = basic + hra

Code

```
import csv


def create():
    f = open('employee.csv', 'w', newline='')
    w = csv.writer(f)
    w.writerow(['ID', 'Name', 'Salary', 'Department'])
    f.flush()
    while True:
        id = int(input("Enter ID: "))
        name = input("Enter Name: ")
        dept = input("Enter Department: ")
        basic = int(input("Enter Basic Salary: "))
        hra = 0.1 * basic  # House Rent Allowance
        sal = basic + hra
        w.writerow([id, name, dept, str(basic),
str(hra), str(sal)])
        f.flush()
        ch = input("Do you want to add more records?
(y/n): ")
        if ch != 'y':
            break
    f.close()


create()
```

Output

```
Enter ID: 101
Enter Name: Taha
Enter Department: IT
Enter Basic Salary: 1000
Do you want to add more records? (y/n): y
Enter ID: 102
Enter Name: Shehzad
Enter Department: HR
Enter Basic Salary: 1000
Do you want to add more records? (y/n): n
```

File Content

```
ID,Name,Salary,Department
101,Taha,IT,1000,100.0,1100.0
102,Shehzad,HR,1000,100.0,1100.0
```

Experiment No. 20

Aim: Copy file1.csv into file2.csv.

Code

```
import csv


def copy():
    f = open('csv1.csv', 'r')
    if not f:
        print("File not found!")
        exit()
    r = csv.reader(f)
    f1 = open('csv2.csv', 'w', newline='')
    w = csv.writer(f1)
    for row in r:
        w.writerow(row)
    f.close()
    f1.close()


copy()
```

File Content

csv1.csv

```
Name,Abbreviation,Numeric
January,Jan,1
Feburary,Feb,2
March,Mar,3
April,Apr,4
May,May,5
June,June,6
July,July,7
August,Aug,8
September,Sept,9
October,Oct,10
November,Nov,11
December,Dec,12
```

csv2.csv

```
Name,Abbreviation,Numeric
January,Jan,1
Feburary,Feb,2
March,Mar,3
April,Apr,4
May,May,5
June,June,6
July,July,7
August,Aug,8
September,Sept,9
October,Oct,10
November,Nov,11
December,Dec,12
```

1. Display all the existing databases in your system

```
mysql> SHOW databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

2. Create a database 2022

```
mysql> CREATE DATABASE AY2022;
Query OK, 1 row affected (0.00 sec)
```

3. Check whether 2022 database created or not

```
mysql> SHOW databases;
+--------------------+
| Database           |
+--------------------+
| ay2022             |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```

4. Open database 2022

```
mysql> USE AY2022;
Database changed
```

5. Create a table STUDENT with the following specifications:

FIELD NAME	DATA TYPE AND SIZE	CONSTRAINTS
ROLL_NO	INT(5)	
STUD_NAME	VARCHAR(35)	
STREAM	CHAR(20)	
MARK1	DECIMAL(6,2)	
MARK2	DECIMAL(6,2)	
DOB	DATE	

ROLL_NO	STUD_NAME	STREAM	MARK1	MARK2	DOB
101	Surya Takur	Science	90	87	10/10/1990
102	Chris Tom	Humanities	88	91	5/1/1994
103	Abel George	Commerce	93	95	7/10/1993
104	Nathel Pillai	Science	56	58	8/8/1994

```
mysql> CREATE TABLE student (ROLL_NO INT(5), STUD_NAME VARCHAR(35), STREAM
CHAR(20), MARK1 DECIMAL(6,2), MARK2 DECIMAL(6,2), DOB DATE);
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

6. Display all the existing table in the database 2022

```
mysql> SHOW tables;
+------------------+
| Tables_in_ay2022 |
+------------------+
| student          |
+------------------+
1 row in set (0.00 sec)
```

7. Delete the table STUDENT and check.

```
mysql> DROP table student;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SHOW tables;
Empty set (0.00 sec)
```

8. Create a table TEACHER with the following specifications in the database AY2022:

FIELD NAME	DATA TYPE AND SIZE	CONSTRAINTS
TR_ID	INT(10)	
TR_NAME	VARCHAR(35)	
TR_SAL	DECIMAL(15,2)	
ROLL_NO	INT(5)	

```
mysql> CREATE TABLE teacher (TR_ID INT(10), TR_NAME VARCHAR(35), TR_SAL
DECIMAL(15,2), ROLL_NO INT(5));
Query OK, 0 rows affected, 2 warnings (0.00 sec)
```

9. Insert values given to STUDENT table.

```
mysql> INSERT INTO student VALUES(101, 'Surya Takur', 'Science', 90, 87, '1990/10/10');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student VALUES(102, 'Chris Tom', 'Humanities', 88, 91, '1994/1/5');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student VALUES(103, 'Abel George', 'Commerce', 93, 95, '1993/10/7');

Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO student VALUES(104, 'Nathel Pillai', 'Science', 56, 58, '1994/8/8');

Query OK, 1 row affected (0.00 sec)
```

10. Display the structure of the STUDENT table.

```
mysql> desc student;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| ROLL_NO   | int          | YES  |     | NULL    |       |
| STUD_NAME | varchar(35)  | YES  |     | NULL    |       |
| STREAM    | char(20)     | YES  |     | NULL    |       |
| MARK1     | decimal(6,2) | YES  |     | NULL    |       |
| MARK2     | decimal(6,2) | YES  |     | NULL    |       |
| DOB       | date         | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

11. Delete the table TEACHER and check

```
mysql> DROP TABLE teacher;
Query OK, 0 rows affected (0.00 sec)
```

12. Create a database SCHOOL, display all databases, delete it and check.

```
mysql> CREATE DATABASE school;
Query OK, 1 row affected (0.00 sec)
mysql> DROP DATABASE school;
Query OK, 0 rows affected (0.00 sec)
```

13. Display all the contents from STUDENT table.

```
mysql> SELECT * FROM student;
+---------+---------------+------------+-------+-------+------------+
| ROLL_NO | STUD_NAME     | STREAM     | MARK1 | MARK2 | DOB        |
+---------+---------------+------------+-------+-------+------------+
|     101 | Surya Takur   | Science    | 90.00 | 87.00 | 1990-10-10 |
|     102 | Chris Tom     | Humanities | 88.00 | 91.00 | 1994-01-05 |
|     103 | Abel George   | Commerce   | 93.00 | 95.00 | 1993-10-07 |
|     104 | Nathel Pillai | Science    | 56.00 | 58.00 | 1994-08-08 |
+---------+---------------+------------+-------+-------+------------+
4 rows in set (0.00 sec)
```

14. Display all the contents from STUDENT table.

```
mysql> SELECT stud_name, stream FROM STUDENT;
+---------------+------------+
| stud_name     | stream     |
+---------------+------------+
| Surya Takur   | Science    |
| Chris Tom     | Humanities |
| Abel George   | Commerce   |
| Nathel Pillai | Science    |
+---------------+------------+
4 rows in set (0.00 sec)
```

15. Display all the students in science stream.

```
mysql> SELECT stud_name FROM student WHERE stream='Science';
+---------------+
| stud_name     |
+---------------+
| Surya Takur   |
| Nathel Pillai |
+---------------+
2 rows in set (0.00 sec)
```

16. Display all student name and both marks whose mark1 is greater than 89.

```
mysql> SELECT stud_name, mark1, mark2 FROM student WHERE mark1 > 89;
+-------------+-------+-------+
| stud_name   | mark1 | mark2 |
+-------------+-------+-------+
| Surya Takur | 90.00 | 87.00 |
| Abel George | 93.00 | 95.00 |
+-------------+-------+-------+
2 rows in set (0.00 sec)
```

17. Display all student details whose mark2 is between 91 and 95.

```
mysql> SELECT * FROM student WHERE mark2 BETWEEN 91 AND 95;
+---------+-------------+------------+-------+-------+------------+
| ROLL_NO | STUD_NAME   | STREAM     | MARK1 | MARK2 | DOB        |
+---------+-------------+------------+-------+-------+------------+
|     102 | Chris Tom   | Humanities | 88.00 | 91.00 | 1994-01-05 |
|     103 | Abel George | Commerce   | 93.00 | 95.00 | 1993-10-07 |
+---------+-------------+------------+-------+-------+------------+
2 rows in set (0.00 sec)
```

18. Display all student details whose Date of Birth is after January 1, 1994.

```
mysql> SELECT * FROM student WHERE DOB >= '1994-01-01';
+---------+---------------+------------+-------+-------+------------+
| ROLL_NO | STUD_NAME     | STREAM     | MARK1 | MARK2 | DOB        |
+---------+---------------+------------+-------+-------+------------+
|     102 | Chris Tom     | Humanities | 88.00 | 91.00 | 1994-01-05 |
|     104 | Nathel Pillai | Science    | 56.00 | 58.00 | 1994-08-08 |
+---------+---------------+------------+-------+-------+------------+
2 rows in set (0.00 sec)
```

19. Display all student details whose Date of Birth is January 1, 1994.

```
mysql> SELECT * FROM student WHERE DOB = '1994-01-01';
Empty set (0.00 sec)
```

20. Insert using field names and using null

```
mysql> INSERT INTO student VALUES(110, 'Karthik Kiran', null, 100, null,
'1993/10/10');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO student(roll_no, stud_name, dob) VALUES(111, 'Jerry John',
'1994/11/19');
Query OK, 1 row affected (0.00 sec)
```

21. Display all the records with either mark1 or mark2 is null

```
mysql> SELECT * FROM student WHERE mark1 IS NULL OR mark2 IS NULL;
+---------+---------------+--------+--------+-------+------------+
| ROLL_NO | STUD_NAME     | STREAM | MARK1  | MARK2 | DOB        |
+---------+---------------+--------+--------+-------+------------+
|     110 | Karthik Kiran | NULL   | 100.00 |  NULL | 1993-10-10 |
|     111 | Jerry John    | NULL   |   NULL |  NULL | 1994-11-19 |
+---------+---------------+--------+--------+-------+------------+
2 rows in set (0.00 sec)
```

22. Display all the records with either mark1 or mark2 is not null

```
mysql> SELECT * FROM student WHERE mark1 IS NOT NULL OR MARK2 IS NOT NULL;
+---------+---------------+------------+--------+-------+------------+
| ROLL_NO | STUD_NAME     | STREAM     | MARK1  | MARK2 | DOB        |
+---------+---------------+------------+--------+-------+------------+
|     101 | Surya Takur   | Science    |  90.00 | 87.00 | 1990-10-10 |
|     102 | Chris Tom     | Humanities |  88.00 | 91.00 | 1994-01-05 |
|     103 | Abel George   | Commerce   |  93.00 | 95.00 | 1993-10-07 |
|     104 | Nathel Pillai | Science    |  56.00 | 58.00 | 1994-08-08 |
|     110 | Karthik Kiran | NULL       | 100.00 |  NULL | 1993-10-10 |
+---------+---------------+------------+--------+-------+------------+
5 rows in set (0.00 sec)
```

23. Display all the streams available in STUDENT table without duplicating.

```
mysql> SELECT DISTINCT stream FROM student;
+------------+
| stream     |
+------------+
| Science    |
| Humanities |
| Commerce   |
| NULL       |
+------------+
4 rows in set (0.00 sec)
```

24. Display the student names along with roll number with appropriate field names as student name and roll number (alias names / another name)

```
mysql> SELECT roll_no ROLL_NUMBER, stud_name STUDENT_NAME FROM student;
+-------------+--------------+
| ROLL_NUMBER | STUDENT_NAME  |
+-------------+--------------+
|         101 | Surya Takur   |
|         102 | Chris Tom     |
|         103 | Abel George   |
|         104 | Nathel Pillai |
|         110 | Karthik Kiran |
|         111 | Jerry John    |
+-------------+--------------+
6 rows in set (0.00 sec)
```

25. Display student names along with date of birth whose mark2 is 91 and 95 (using IN operator)

```
mysql> SELECT stud_name, dob FROM student WHERE mark2 IN (91,95);
+-------------+------------+
| stud_name   | dob        |
+-------------+------------+
| Chris Tom   | 1994-01-05 |
| Abel George | 1993-10-07 |
+-------------+------------+
2 rows in set (0.00 sec)
```

26. Display all the student details whose date of birth is in the year 1994 (using year() function)

```
mysql> SELECT * FROM student WHERE year(dob)='1994';
+---------+---------------+------------+-------+-------+------------+
| ROLL_NO | STUD_NAME     | STREAM     | MARK1 | MARK2 | DOB        |
+---------+---------------+------------+-------+-------+------------+
|     102 | Chris Tom     | Humanities | 88.00 | 91.00 | 1994-01-05 |
|     104 | Nathel Pillai | Science    | 56.00 | 58.00 | 1994-08-08 |
|     111 | Jerry John    | NULL       |  NULL |  NULL | 1994-11-19 |
+---------+---------------+------------+-------+-------+------------+
3 rows in set (0.00 sec)
```

27. Display the student roll number along with student name whose date of birth is in October month (using month() function)

```
mysql> SELECT roll_no, stud_name FROM student WHERE month(dob)='10';
+---------+---------------+
| roll_no | stud_name     |
+---------+---------------+
|     101 | Surya Takur   |
|     103 | Abel George   |
|     110 | Karthik Kiran |
+---------+---------------+
3 rows in set (0.00 sec)
```

28. Display the student roll number along with student name whose date of birth is in date after 9th of the month (using day() function)

```
mysql> SELECT roll_no, stud_name FROM student where day(dob)>9;
+---------+---------------+
| roll_no | stud_name     |
+---------+---------------+
|     101 | Surya Takur   |
|     110 | Karthik Kiran |
|     111 | Jerry John    |
+---------+---------------+
3 rows in set (0.00 sec)
```

29. Display all the stream names ending with the letter 'e' from the table STUDENT

```
mysql> SELECT stream FROM student WHERE stream LIKE "%e%";
+------------+
| stream     |
+------------+
| Science    |
| Humanities |
| Commerce   |
| Science    |
+------------+
4 rows in set (0.00 sec)
```

30. Display student names that contain the letter 'i'

```
mysql> SELECT stud_name FROM student WHERE stud_name LIKE "%i%";
+---------------+
| stud_name     |
+---------------+
| Chris Tom     |
| Nathel Pillai |
| Karthik Kiran |
+---------------+
3 rows in set (0.00 sec)
```

31. Display the student names whose first names have EXACTLY 4 characters

```
mysql> SELECT stud_name FROM student WHERE stud_name LIKE "____ %";
+-------------+
| stud_name   |
+-------------+
| Abel George |
+-------------+
1 row in set (0.00 sec)
```

32. Display stream names whose third character is the letter 'm'

```
mysql> SELECT stud_name FROM student WHERE stream LIKE "__m%";
+-------------+
| stud_name   |
+-------------+
| Chris Tom   |
| Abel George |
+-------------+
2 rows in set (0.00 sec)
```

33. Display student name along with the date of birth whose date of birth is in October

```
mysql> SELECT stud_name, dob FROM student WHERE dob LIKE "%-10-%";
+---------------+------------+
| stud_name     | dob        |
+---------------+------------+
| Surya Takur   | 1990-10-10 |
| Abel George   | 1993-10-07 |
| Karthik Kiran | 1993-10-10 |
+---------------+------------+
3 rows in set (0.00 sec)
```

34. Display student names in descending order

```
mysql> SELECT stud_name FROM student ORDER BY stud_name DESC;
+---------------+
| stud_name     |
+---------------+
| Surya Takur   |
| Nathel Pillai |
| Karthik Kiran |
| Jerry John    |
| Chris Tom     |
| Abel George   |
+---------------+
6 rows in set (0.00 sec)
```

35. Display student names and date of birth from science stream in the ascending order of birth

```
mysql> SELECT stud_name, dob FROM student WHERE stream="Science" ORDER BY dob;
+---------------+------------+
| stud_name     | dob        |
+---------------+------------+
| Surya Takur   | 1990-10-10 |
| Nathel Pillai | 1994-08-08 |
+---------------+------------+
2 rows in set (0.00 sec)
```

36. Display stream names whose length is more than 7 characters

```
mysql> SELECT stream FROM student WHERE length(stream)>7;
+------------+
| stream     |
+------------+
| Humanities |
| Commerce   |
+------------+
2 rows in set (0.00 sec)
```

37. Change the date of birth of roll no 102 as 1/5/1994

```
mysql> UPDATE student SET dob="1994-5-1" WHERE roll_no=102;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

38. Increase the mark1 for all students by 2.

```
mysql> UPDATE student SET mark1=mark1+2;
Query OK, 5 rows affected (0.00 sec)
Rows matched: 6  Changed: 5  Warnings: 0
```

39. Change the name of student Nathel Pillai to Nithin Pillai whose roll number is 104.

```
mysql> UPDATE student SET stud_name="Nithin Pillai" WHERE roll_no=104;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

40. Delete student details who score less than 88 in both marks

```
mysql> DELETE FROM student WHERE mark1<88 AND mark2<88;
Query OK, 1 row affected (0.00 sec)
```

41. Delete all the contents from student table

```
mysql> DELETE FROM student;
Query OK, 5 rows affected (0.00 sec)
```

42. Create a view ‘name’ from student table with roll number and student name whose both marks are above 90

```
mysql> CREATE VIEW name AS SELECT roll_no, stud_name FROM student WHERE mark1>90
AND mark2>90;
Query OK, 0 rows affected (0.00 sec)
```

43. Delete the view ‘name’ from the AY2022 database

```
mysql> DROP VIEW name;
Query OK, 0 rows affected (0.00 sec)
```

44. Add a column LOCATION to student table with data type char of size 50

```
mysql> ALTER TABLE student ADD COLUMN location char(50);
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC student;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| ROLL_NO   | int          | YES  |     | NULL    |       |
| STUD_NAME | varchar(35)  | YES  |     | NULL    |       |
| STREAM    | char(20)     | YES  |     | NULL    |       |
| MARK1     | decimal(6,2) | YES  |     | NULL    |       |
| MARK2     | decimal(6,2) | YES  |     | NULL    |       |
| DOB       | date         | YES  |     | NULL    |       |
| location  | varchar(50)  | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

45. Delete the column LOCATION from the student table

```
mysql> ALTER TABLE student DROP COLUMN location;
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC student;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| ROLL_NO   | int          | YES  |     | NULL    |       |
| STUD_NAME | varchar(35)  | YES  |     | NULL    |       |
| STREAM    | char(20)     | YES  |     | NULL    |       |
| MARK1     | decimal(6,2) | YES  |     | NULL    |       |
| MARK2     | decimal(6,2) | YES  |     | NULL    |       |
| DOB       | date         | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

46. Modify column LOCATION details as data type varchar of size 40.

```
mysql> ALTER TABLE student MODIFY location varchar(40);
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc STUDENT;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| roll_no   | int(5)       | NO   | PRI | NULL    |       |
| stud_name | varchar(35)  | YES  |     | NULL    |       |
| stream    | char(20)     | YES  |     | NULL    |       |
| mark1     | decimal(6,2) | YES  |     | NULL    |       |
| mark2     | decimal(6,2) | YES  |     | NULL    |       |
| dob       | date         | YES  |     | NULL    |       |
| location  | varchar(40)  | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

47. Rename the column LOCATION from the student table as STREET with varchar data type and size 90.

```
mysql> ALTER TABLE student CHANGE location street varchar(90);
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> DESC student;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| roll_no   | int(5)       | NO   | PRI | NULL    |       |
| stud_name | varchar(35)  | YES  |     | NULL    |       |
| stream    | char(20)     | YES  |     | NULL    |       |
| mark1     | decimal(6,2) | YES  |     | NULL    |       |
| mark2     | decimal(6,2) | YES  |     | NULL    |       |
| dob       | date         | YES  |     | NULL    |       |
| street    | varchar(90)  | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

48. Display the total mark1 marks from student table.

```
mysql> SELECT SUM(mark1) FROM student;
+------------+
| sum(mark1) |
+------------+
|     327.00 |
+------------+
1 row in set (0.00 sec)
```

49. Display senior-most and junior-most student from the student table.

```
mysql> SELECT MIN(dob), MAX(dob) FROM student;
+------------+------------+
| min(dob)   | max(dob)   |
+------------+------------+
| 1990-10-10 | 1994-08-08 |
+------------+------------+
1 row in set (0.00 sec)
```

50. Display the number of students whose mark1 is more than 90 marks.

```
mysql> SELECT COUNT(*) FROM student WHERE mark1>90;
+----------+
| COUNT(*) |
+----------+
|        1 |
+----------+
1 row in set (0.00 sec)
```

51. Display the average marks from mark2 whose second name starts with the letter ‘t’, round off to two dp.

```
mysql> SELECT round(AVG(mark2),2) FROM student WHERE stud_name LIKE "% T%";
+---------------------+
| round(AVG(mark2),2) |
+---------------------+
|               89.00 |
+---------------------+
1 row in set (0.00 sec)
```

52. Display all student details date of birth wise whose names contains letter ‘a’.

```
mysql> SELECT * FROM student WHERE stud_name LIKE "%a%" GROUP BY dob;
+---------+---------------+----------+-------+-------+------------+--------+
| roll_no | stud_name     | stream   | mark1 | mark2 | dob        | street |
+---------+---------------+----------+-------+-------+------------+--------+
|     101 | Surya Takur   | Science  | 90.00 | 87.00 | 1990-10-10 | Dubai  |
|     103 | Abel George   | Commerce | 93.00 | 95.00 | 1993-10-07 | Dubai  |
|     104 | Nathel Pillai | Science  | 56.00 | 58.00 | 1994-08-08 | Dubai  |
+---------+---------------+----------+-------+-------+------------+--------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM student GROUP BY dob HAVING stud_name LIKE "%a%";
+---------+---------------+----------+-------+-------+------------+--------+
| roll_no | stud_name     | stream   | mark1 | mark2 | dob        | street |
+---------+---------------+----------+-------+-------+------------+--------+
|     101 | Surya Takur   | Science  | 90.00 | 87.00 | 1990-10-10 | Dubai  |
|     103 | Abel George   | Commerce | 93.00 | 95.00 | 1993-10-07 | Dubai  |
|     104 | Nathel Pillai | Science  | 56.00 | 58.00 | 1994-08-08 | Dubai  |
+---------+---------------+----------+-------+-------+------------+--------+
3 rows in set (0.00 sec)
```

53. Display number of students in each stream

```
mysql> SELECT stream, COUNT(stud_name) FROM student GROUP BY stream;
+------------+------------------+
| stream     | COUNT(stud_name) |
+------------+------------------+
| Commerce   |                1 |
| Humanities |                1 |
| Science    |                2 |
+------------+------------------+
3 rows in set (0.00 sec)
```

54. Display number of students in each stream whose date of birth is after 1990

```
mysql> SELECT stream, COUNT(stud_name) FROM student WHERE YEAR(dob)>=1990 GROUP BY
stream;
+------------+------------------+
| stream     | COUNT(stud_name) |
+------------+------------------+
| Science    |                2 |
| Humanities |                1 |
| Commerce   |                1 |
+------------+------------------+
3 rows in set (0.00 sec)
```

55. Display all details from teacher and student tables who mark1 is greater than 90

```
mysql> SELECT * FROM student, teacher WHERE student.roll_no=teacher.roll_no AND
mark1>90;
+---------+-------------+----------+-------+-------+------------+--------+-------+------------+---------+-------
--+
| roll_no | stud_name   | stream   | mark1 | mark2 | dob        | street | tr_id | tr_name    | tr_sal  |
roll_no |
+---------+-------------+----------+-------+-------+------------+--------+-------+------------+---------+-------
--+
|     103 | Abel George | Commerce | 93.00 | 95.00 | 1993-10-07 | Dubai  |   999 | Elias John | 9900.00 |
103 |
+---------+-------------+----------+-------+-------+------------+--------+-------+------------+---------+-------
--+
1 row in set (0.00 sec)
```

Write the output for the following MySQL Commands:

56. SELECT POW(5,2), POW(2,5);

```
mysql> SELECT POW(5,2), POW(2,5);
+----------+----------+
| POW(5,2) | POW(2,5) |
+----------+----------+
|       25 |       32 |
+----------+----------+
1 row in set (0.00 sec)
```

57. SELECT ROUND(576.9875,2), ROUND(576.9875), ROUND(576.9875,-1);

```
mysql> SELECT ROUND(576.9875,2), ROUND(576.9875), ROUND(576.9875,-1);
+-------------------+-----------------+--------------------+
| ROUND(576.9875,2) | ROUND(576.9875) | ROUND(576.9875,-1) |
+-------------------+-----------------+--------------------+
|            576.99 |             577 |                580 |
+-------------------+-----------------+--------------------+
1 row in set (0.01 sec)
```

58. SELECT TRUNCATE(576.9875,2),TRUNCATE(576.9875,-1);

```
mysql> SELECT TRUNCATE(576.9875,2),TRUNCATE(576.9875,-1);
+----------------------+-----------------------+
| TRUNCATE(576.9875,2) | TRUNCATE(576.9875,-1) |
+----------------------+-----------------------+
|               576.98 |                   570 |
+----------------------+-----------------------+
1 row in set (0.00 sec)
```

59. SELECT LENGTH("Never Gonna Give You Up");

```
mysql> SELECT LENGTH("Never Gonna Give You Up");
+-----------------------------------+
| LENGTH("Never Gonna Give You Up") |
+-----------------------------------+
|                                23 |
+-----------------------------------+
1 row in set (0.00 sec)
```

60. SELECT CONCAT("Rick ","Astley")"Full Name";

```
mysql> SELECT CONCAT("Rick ","Astley")"Full Name";
+-------------+
| Full Name   |
+-------------+
| Rick Astley |
+-------------+
1 row in set (0.00 sec)
```

61. SELECT YEAR(CURDATE()), MONTH(CURDATE()), DAY(CURDATE());

```
mysql> SELECT YEAR(CURDATE()), MONTH(CURDATE()), DAY(CURDATE());
+-----------------+------------------+----------------+
| YEAR(CURDATE()) | month(CURDATE()) | day(CURDATE()) |
+-----------------+------------------+----------------+
|            2022 |               11 |             19 |
+-----------------+------------------+----------------+
1 row in set (0.01 sec)
```

62. SELECT DAYOFYEAR(CURDATE()), DAYOFMONTH(CURDATE()), DAYNAME(CURDATE());

```
mysql> SELECT DAYOFYEAR(CURDATE()), DAYOFMONTH(CURDATE()), DAYNAME(CURDATE());
+---------------------+----------------------+--------------------+
| DAYOFYEAR(CURDATE()) | DAYOFMONTH(CURDATE()) | DAYNAME(CURDATE()) |
+---------------------+----------------------+--------------------+
|                  323 |                   19 | Saturday           |
+---------------------+----------------------+--------------------+
1 row in set (0.00 sec)
```

63. SELECT LEFT("Unicode", 3),RIGHT("Unicode", 4);

```
mysql> SELECT LEFT("Unicode", 3),RIGHT("Unicode", 4);
+--------------------+---------------------+
| LEFT("Unicode", 3) | RIGHT("Unicode", 4) |
+--------------------+---------------------+
| Uni                | code                |
+--------------------+---------------------+
1 row in set (0.00 sec)
```

64. SELECT INSTR("UnitedWeStand", "it"), INSTR("UnitedWeStand", "ye");

```
mysql> SELECT INSTR("UnitedWeStand", "it"), INSTR("UnitedWeStand", "ye");
+-----------------------------+-----------------------------+
| INSTR("UnitedWeStand", "it") | INSTR("UnitedWeStand", "ye") |
+-----------------------------+-----------------------------+
|                           3 |                           0 |
+-----------------------------+-----------------------------+
1 row in set (0.00 sec)
```

65. SELECT MID("UnitedWeStand", 5, 6), SUBSTR("UnitedWeStand", 1);

```
mysql> SELECT MID("UnitedWeStand", 5, 6), SUBSTR("UnitedWeStand", 1);
+---------------------------+---------------------------+
| MID("UnitedWeStand", 5, 6) | SUBSTR("UnitedWeStand", 1) |
+---------------------------+---------------------------+
| edWeSt                    | UnitedWeStand             |
+---------------------------+---------------------------+
1 row in set (0.00 sec)
```