DELHI PUBLIC SCHOOL GHAZIABAD MEERUT ROAD

Computer Science Practical File

Submitted By:
Nandita Krishnan
(XII- A5)

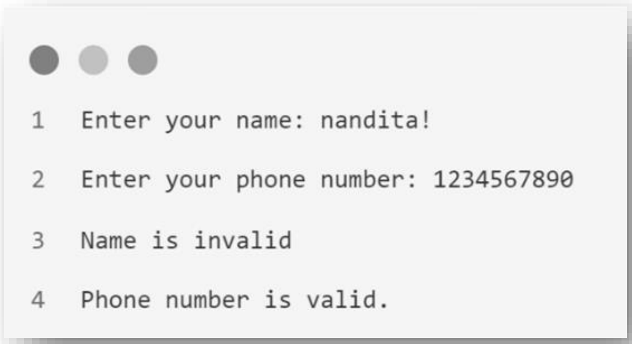
Submitted To:
Ruchi Sharma

PYTHON

1) Write a Python function to validate (nm, ph) return true if the name has all alphabets or spaces and the phone number has 10 digits only and returns false if any of the value is invalid.
   Use this function to check validity of any name and phone number entered by user.

```
global alpha
alpha = 'abcdefghijklmnopqrstuvwxyz '
def identifier(nm, ph):
      try:
        out = ''
        for i in str(nm):
            if i.isalpha() == True:
                out = 'Name is valid'
            elif i.isspace() == True:
                out = 'Name is valid'
            else:
                out = 'Name is invalid'
        print(out)
      except TypeError:
        print (“Data type not string.”)
        pass
      try:
        if len(str(ph)) == 10:
            print("Phone number is valid.")
        else:
            print("Phone number is invalid.")
      except TypeError:
        print(“Data type not numeral.”)
        pass
'''identifier('nandita@', 1234567)
identifier('nandita', 123456790)
identifier(' ', 67577256523753)'''
x = input("Enter your name: ")
y = int(input("Enter your phone number: "))
identifier(x,y)
```

OUTPUT



```
Enter your name: nandita!
Enter your phone number: 1234567890
Name is invalid
Phone number is valid.
```

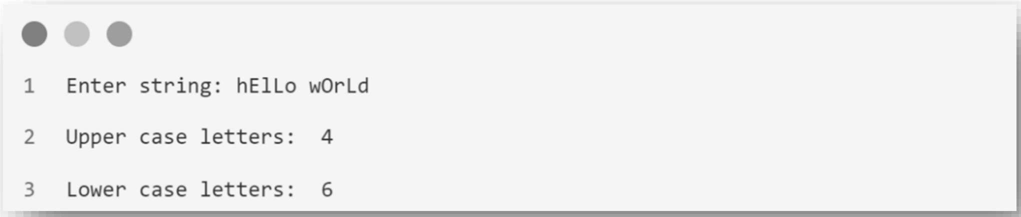
2) Write a Python function that accepts a string and calculates the number of upper-case letters and lower-case letters.
   Sample String: 'The quick Brow Fox'
   Expected Output:
   No. of Upper-case characters: 3
   No. of Lower case Characters: 12

Use this function in the same program and print number of upper- and lower-case letters in the program.

```
def case_checker(x):
    up = 0
    low = 0
    try:
        for i in x:
            if i.isupper() == True:
                up += 1
            if i.islower() == True:
                low += 1
    except:
        print("Error.")
        pass
    print("Upper case letters: ", up, "\nLower case letters: ", low)
```

```
'''case_checker(1)
case_checker('nAnDitA')'''
y = input("Enter string: ")
case_checker(y)
```

OUTPUT

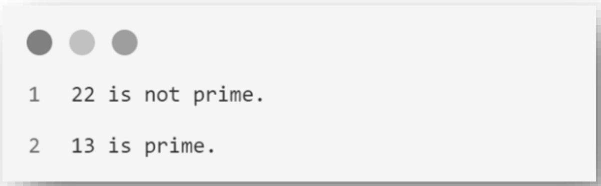


```
Enter string: hElLo wOrLd
Upper case letters:  4
Lower case letters:  6
```

3) Write a Python function that takes a number as a parameter and check the number is prime or not.

```
def isprime(x):
    for i in range(2, x):
        if x%i == 0:
            print(x, 'is not prime.')
            break
        else:
            print(x, 'is prime.')
            break
isprime(22)
isprime(13)
```

OUTPUT

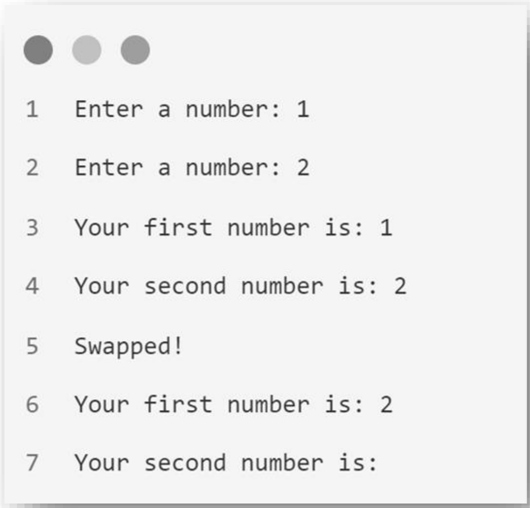


```
22 is not prime.
13 is prime.
```

4) Write a program that has a user-defined function to accept 2 numbers as parameters, if number 1 is less than number 2 then numbers are swapped and returned, i.e., number 2 is returned in place of number1 and number 1 is reformed in place of number 2, otherwise the same order is returned. Show the use of this function in the same program.

```
def swapper(x,y):
    if type(x) != int or type(y) != int:
        print("Invalid input")
    print("Your first number is: " +  str(x) + "\nYour second number
is: " + str(y))
    if x < y:
        print("Swapped!")
        x,y = y, x
        print("Your first number is: " +  str(x) + "\nYour second
number is: " + str(y))
    else:
        print("No change")
        print("Your first number is: " +  str(x) + "\nYour second
number is: " + str(y))

'''swapper(1,2)
swapper(5,2)'''
n = int(input("Enter a number: "))
m = int(input("Enter a number: "))
swapper(n,m)
```

OUTPUT

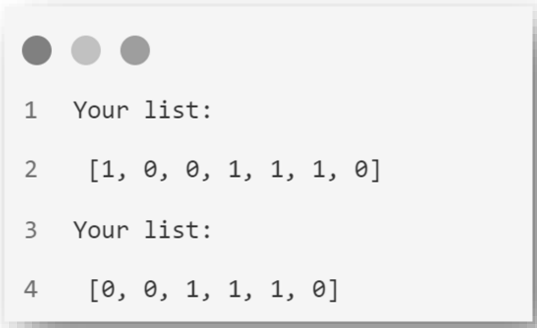


```
Enter a number: 1
Enter a number: 2
Your first number is: 1
Your second number is: 2
Swapped!
Your first number is: 2
Your second number is:
```

5) Write a Python function modify(lst) that takes a list of numbers as a parameter and changes all the multiples of 5 to 0 and other numbers to 1.

```
def modify(*lst):
    num = list(lst)
    for i in range(0, len(num)):
        try:
            if num[i]%5 == 0:
                num[i] = 0
            else:
                num[i] = 1
        except TypeError:
            print("Invalid Input. Check Data type.")
            pass
    print("Your list:\n",num)

modify(1,5,10,2,4,6,20)
modify(50, 435, 43, 756, 12, 95)
```

OUTPUT



```
Your list:
 [1, 0, 0, 1, 1, 1, 0]
Your list:
 [0, 0, 1, 1, 1, 0]
```

6) Write a program to create a list of lists for storing employee details (name, age, department and salary), for example emp=[[‘Rajnish’,36,’Testing’,45000], [‘Amrita’,45,’HR’,50000]] emp will be declared globally.
In this program create following functions:
 a. To add a new employee detail in list, function name will be add_emp(empd

ata), where parameter empdata is list of employee details (name, age, department and salary)

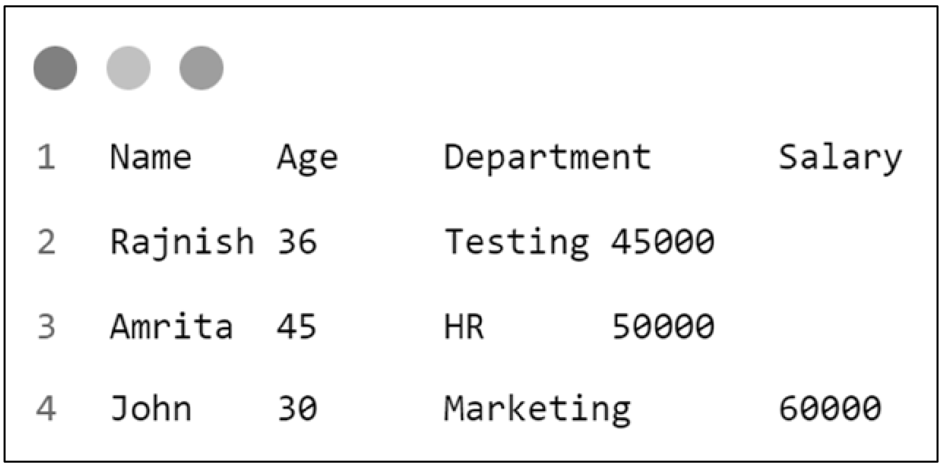
b. To display student details in tabular for as given below, show_employee(emp) where emp is list of employees:

Name Age dep Salary

Rajnish 16 testing 45000

Amrita 15 HR 50000

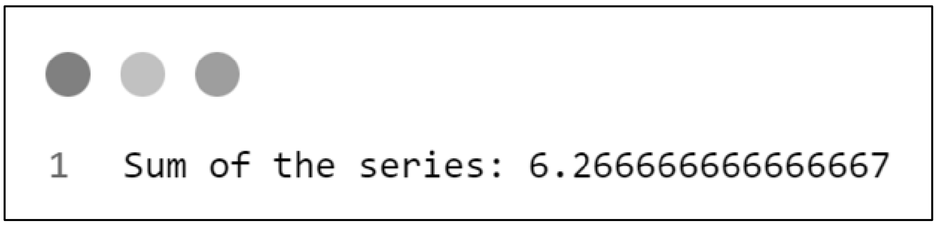
```
emp = [['Rajnish', 36, 'Testing', 45000], ['Amrita', 45, 'HR', 50000]]

def add_emp(empdata):
    emp.append(empdata)

def show_employee(emp):
    print("Name\tAge\tDepartment\tSalary")
    for e in emp:
        print("\t".join(map(str, e)))

new_emp_data = ['John', 30, 'Marketing', 60000]
add_emp(new_emp_data)
show_employee(emp)
```



```
Name    Age     Department      Salary
Rajnish 36      Testing 45000
Amrita  45      HR      50000
John    30      Marketing       60000
```

7) Define a function that will take two numbers as parameter Sum_of_series(x,n) and will evaluate and print sum of following series:
x+x2/2! + x3/3! +x4/4! ....xn/n!

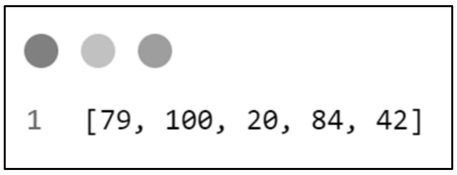
```
def factorial(num):
    result = 1
    for i in range(1, num + 1):
        result *= i
    return result

def sum_of_series(x, n):
    result = sum(x**i / factorial(i) for i in range(1, n + 1))
    print(f"Sum of the series: {result}")

sum_of_series(2, 5)
```



```
Sum of the series: 6.266666666666667
```

8) Write a function to create a list of numbers, where numbers will be generated randomly. Number should be in between an upper limit and lower limit where upper limit and lower limit will be taken as parameter of function.

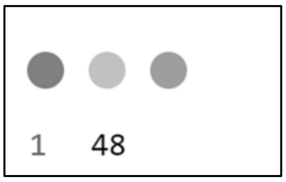
```
import random

def generate_random_numbers(lower_limit, upper_limit, count):
    return [random.randint(lower_limit, upper_limit) for _ in
range(count)]


random_numbers = generate_random_numbers(1, 100, 5)
print(random_numbers)
```



```
[79, 100, 20, 84, 42]
```

9) Write a program to take two number as input and display any random number between those two numbers.

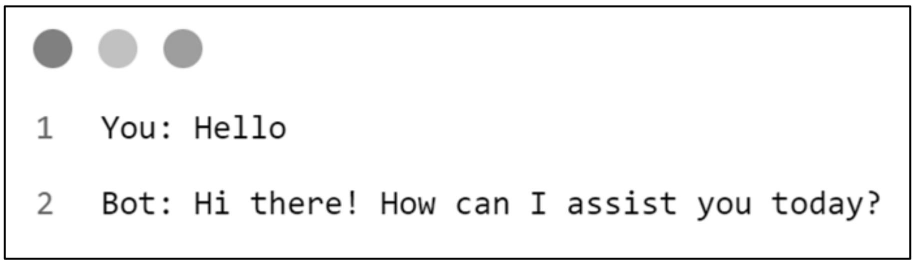
```
import random


def random_number_between(lower_limit, upper_limit):
    return random.randint(lower_limit, upper_limit)

result = random_number_between(10, 50)
print(result)
```



```
1   48
```

10) Write a program to create a simple chatbot using dictionary to store data and random number to generate response.

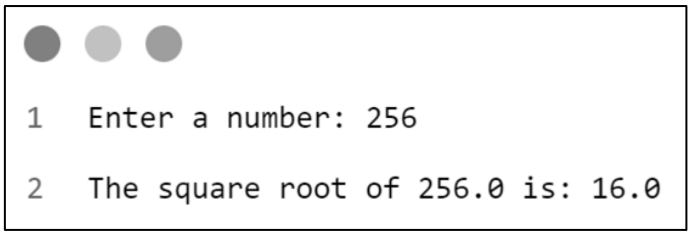
```
import random


responses = {
    "hello": "Hi there! How can I assist you today?",
    "how are you": "I'm just a computer program, but I'm doing well.
Thanks for asking!",
    "what's your name": "You can call me ChatBot.",
    "bye": "Goodbye! If you have more questions, feel free to ask.",
    "tell me a joke": "Why did the Python programmer not respond to
the email? It was in the spam folder!",
    "who created you": "I was created by a team of developers at
OpenAI.",
    "thanks": "You're welcome! If you need anything else, feel free
to ask.",
    "help": "Sure! You can ask me about jokes, information, or just
chat. Try 'who created you' or 'tell me a joke'!",
    "age": "I don't have an age. I'm always up to date!",
    "where are you from": "I exist in the digital realm, so you can
say I'm from the internet!",
    "favorite color": "I don't have a favorite color, but I like all
colors equally!",
    "who are you": "I'm ChatBot, here to assist and chat with you.",
    "weather": "I'm sorry, I don't have real-time information. You
can check a weather website for the latest updates!",
    "music": "I don't have personal preferences, but I can recommend
some popular genres. What kind of music do you like?",
    "movie recommendation": "Sure, I recommend watching 'The
Matrix'. It's a classic!",
    "default": "I don't understand that. Can you ask something
else?",
}

def simple_chatbot():
    user_input = input("You: ").lower()

    if any(greeting in user_input for greeting in ["hello", "hi",
"hey"]):
        response = responses["hello"]
    elif "how are you" in user_input:
        response = responses["how are you"]
    elif "who created you" in user_input:
        response = responses["who created you"]
    elif "tell me a joke" in user_input:
        response = responses["tell me a joke"]
    elif "thanks" in user_input or "thank you" in user_input:
        response = responses["thanks"]
    elif "help" in user_input:
        response = responses["help"]
    elif "bye" in user_input or "goodbye" in user_input:
        response = responses["bye"]
    elif "age" in user_input:
        response = responses["age"]
    elif "where are you from" in user_input:
        response = responses["where are you from"]
    elif "favorite color" in user_input:
        response = responses["favorite color"]
    elif "who are you" in user_input:
        response = responses["who are you"]
    elif "weather" in user_input:
        response = responses["weather"]
    elif "music" in user_input:
        response = responses["music"]
    elif "movie recommendation" in user_input:
        response = responses["movie recommendation"]
    else:
        response = responses["default"]

    print("Bot:", response)

simple_chatbot()
```



```
You: Hello
Bot: Hi there! How can I assist you today?
```

11) Write a program to take a number as input and display its square root using function of math module.

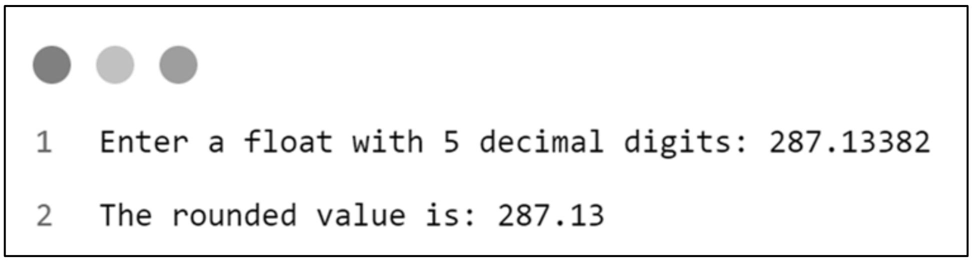
```
import math

def calculate_square_root():
    number = float(input("Enter a number: "))
    result = math.sqrt(number)
    print(f"The square root of {number} is: {result}")

calculate_square_root()
```



```
Enter a number: 256
The square root of 256.0 is: 16.0
```

12) Write a program to take a float with 5 decimal digits as input and display after rounding off that value to 2 decimal places using function of math module.

```python
import math

def round_float():
    float_number = float(input("Enter a float with 5 decimal digits: "))
    rounded_number = round(float_number, 2)
    print(f"The rounded value is: {rounded_number}")


round_float()
```



```
Enter a float with 5 decimal digits: 287.13382
The rounded value is: 287.13
```

MYSQL

A. Write the SQL commands to create the table MOVIES with the following constraints

MovieNo Primary Key

Title UNIQUE

TYPE NOT NULL

RELATION:MOVIES

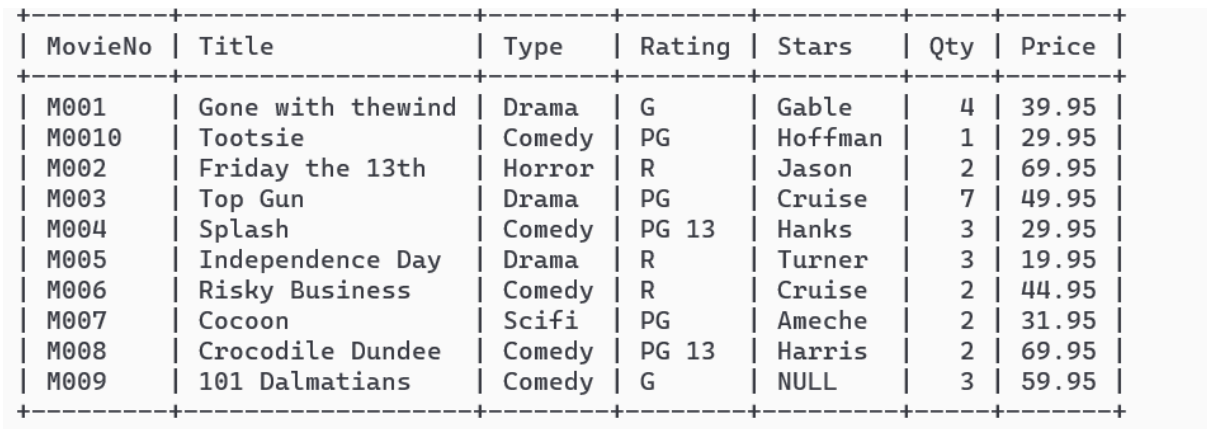
MovieNo.	Title	Type	Rating	Stars	Qty.	Price
M001	Gone with thewind	Drama	G	Gable	4	39.95
M002	Friday the 13th	Horror	R	jason	2	69.95
M003	Top Gun	Drama	PG	Cruise	7	49.95
M004	Splash	Comedy	PG 13	Hanks	3	29.95
M005	IndependenceDay	Drama	R	Turner	3	19.95
M006	Risky Business	Comedy	R	Cruise	2	44.95
M007	Cocoon	Scifi	PG	Ameche	2	31.95
M008	Crocodile Dundee	Comedy	PG13	Harris	2	69.95
M009	101 Dalmatians	Comedy	G	NULL	3	59.95
M0010	Tootsie	Comedy	PG	Hoffman	1	29.95

1. **Dislay the details of the movies having stars as NULL value**
2. **Display the Title and Rating of the movie type COMEDY**
3. **Display the details of movies having a quantity more than 3**
4. **Display the details of Drama movies having price in the range of 50 to 100**
5. **Display the details of movies with stars as "Cruise", "Harris", "Hoffman".**
6. **Display a list of all movies with Price over 20 and sorted by price.**
7. **Display all the movies sorted by QTY in decreasing order.**
8. **Display a report listing books id, Title and current value(Qty* Price).**
9. **Display the names of movies with title starting with the letter "C"**
10. **Display the names of movies with title with "o" as second character.**
11. **Display the names of movies with title with the length of the title name**

```
CREATE DATABASE assignments;
```

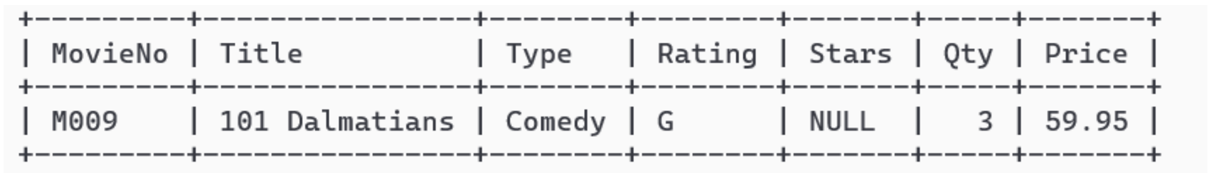
```
USE assigments;
CREATE TABLE Movies(
    MovieNo VARCHAR(5) PRIMARY KEY,
    Title VARCHAR(30) UNIQUE,
    Type VARCHAR(20) NOT NULL,
    Rating VARCHAR(5),
    Stars VARCHAR(10),
    Qty INTEGER,
    Price FLOAT);
INSERT INTO Movies
VALUES
    ('M001', 'Gone With The Wind', 'Drama', 'G', 'Gable', 4, 39.95),
    ('M002', 'Friday The 13th', 'Horror', 'R', 'Jason', 2, 69.95),
    ('M003', 'Top Gun', 'Drama', 'PG', 'Cruise', 7, 49.95),
    ('M004', 'Splash', 'Comedy', 'PG 13', 'Hanks', 3, 29.95),
    ('M005', 'Independence Day', 'Drama', 'R', 'Turner', 3, 19.95),
    ('M006', 'Risky Buisness', 'Comedy', 'R', 'Cruise', 2, 44.95),
    ('M007', 'Cocoon', 'Scifi', 'PG', 'Ameche', 2, 31.95),
    ('M008', 'Crocodile Dundee', 'Comedy', 'PG 13', 'Harris', 2, 69.95),
    ('M009', '101 Dalmatians', 'Comedy', 'G', NULL, 3, 59.95),
    ('M0010', 'Tootsie', 'Comedy', 'PG', 'Hoffman', 1, 29.95);

SELECT * FROM Movies;
```

<u>OUTPUT</u>



```
+---------+-------------------+--------+--------+---------+-----+-------+
| MovieNo | Title             | Type   | Rating | Stars   | Qty | Price |
+---------+-------------------+--------+--------+---------+-----+-------+
| M001    | Gone with thewind | Drama  | G      | Gable   |   4 | 39.95 |
| M0010   | Tootsie           | Comedy | PG     | Hoffman |   1 | 29.95 |
| M002    | Friday the 13th   | Horror | R      | Jason   |   2 | 69.95 |
| M003    | Top Gun           | Drama  | PG     | Cruise  |   7 | 49.95 |
| M004    | Splash            | Comedy | PG 13  | Hanks   |   3 | 29.95 |
| M005    | Independence Day  | Drama  | R      | Turner  |   3 | 19.95 |
| M006    | Risky Business    | Comedy | R      | Cruise  |   2 | 44.95 |
| M007    | Cocoon            | Scifi  | PG     | Ameche  |   2 | 31.95 |
| M008    | Crocodile Dundee  | Comedy | PG 13  | Harris  |   2 | 69.95 |
| M009    | 101 Dalmatians    | Comedy | G      | NULL    |   3 | 59.95 |
+---------+-------------------+--------+--------+---------+-----+-------+
SELECT * FROM Movies WHERE Stars IS NULL;
```

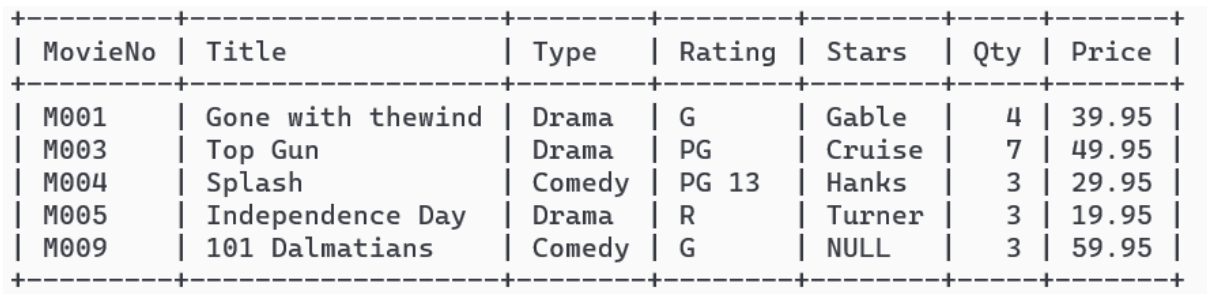
OUTPUT



```
+---------+----------------+--------+--------+-------+-----+-------+
| MovieNo | Title          | Type   | Rating | Stars | Qty | Price |
+---------+----------------+--------+--------+-------+-----+-------+
| M009    | 101 Dalmatians | Comedy | G      | NULL  |   3 | 59.95 |
+---------+----------------+--------+--------+-------+-----+-------+
```

```
SELECT * FROM Movies WHERE Qty >=3;
```

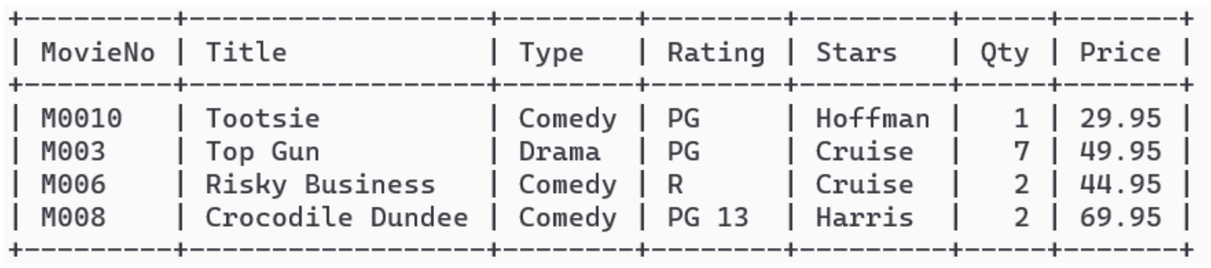
OUTPUT



```
+---------+--------------------+--------+--------+--------+-----+-------+
| MovieNo | Title              | Type   | Rating | Stars  | Qty | Price |
+---------+--------------------+--------+--------+--------+-----+-------+
| M001    | Gone with thewind  | Drama  | G      | Gable  |   4 | 39.95 |
| M003    | Top Gun            | Drama  | PG     | Cruise |   7 | 49.95 |
| M004    | Splash             | Comedy | PG 13  | Hanks  |   3 | 29.95 |
| M005    | Independence Day   | Drama  | R      | Turner |   3 | 19.95 |
| M009    | 101 Dalmatians     | Comedy | G      | NULL   |   3 | 59.95 |
+---------+--------------------+--------+--------+--------+-----+-------+
```

```
SELECT * FROM Movies WHERE Type = 'DRAMA' AND (PRICE BETWEEN 50 and 100);
```

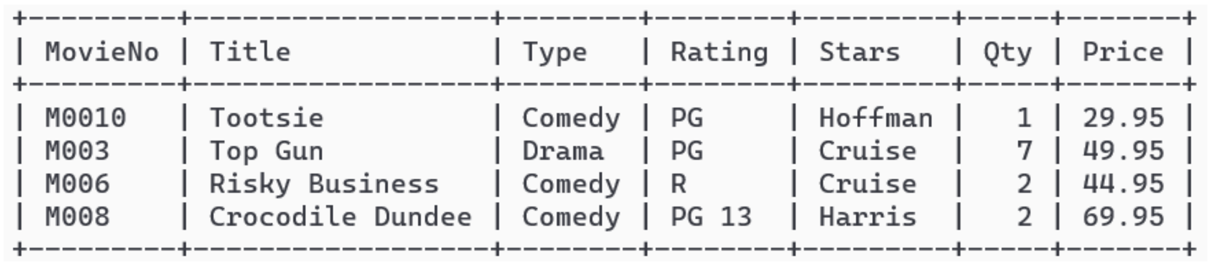
OUTPUT



```
+---------+------------------+--------+--------+---------+-----+-------+
| MovieNo | Title            | Type   | Rating | Stars   | Qty | Price |
+---------+------------------+--------+--------+---------+-----+-------+
| M0010   | Tootsie          | Comedy | PG     | Hoffman |   1 | 29.95 |
| M003    | Top Gun          | Drama  | PG     | Cruise  |   7 | 49.95 |
| M006    | Risky Business   | Comedy | R      | Cruise  |   2 | 44.95 |
| M008    | Crocodile Dundee | Comedy | PG 13  | Harris  |   2 | 69.95 |
+---------+------------------+--------+--------+---------+-----+-------+
```

```
SELECT * FROM MOVIES WHERE(
    Stars = 'Cruise' OR
    Stars = 'Harris' OR
    Stars = 'Hoffman');
```

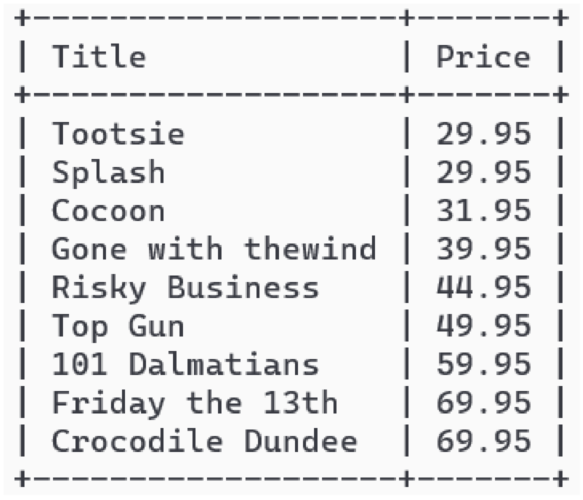
OUTPUT



```
+---------+------------------+--------+--------+---------+-----+-------+
| MovieNo | Title            | Type   | Rating | Stars   | Qty | Price |
+---------+------------------+--------+--------+---------+-----+-------+
| M0010   | Tootsie          | Comedy | PG     | Hoffman |   1 | 29.95 |
| M003    | Top Gun          | Drama  | PG     | Cruise  |   7 | 49.95 |
| M006    | Risky Business   | Comedy | R      | Cruise  |   2 | 44.95 |
| M008    | Crocodile Dundee | Comedy | PG 13  | Harris  |   2 | 69.95 |
+---------+------------------+--------+--------+---------+-----+-------+
```

```
SELECT Title, Price FROM Movies WHERE Price > 20 ORDER BY Price;
```

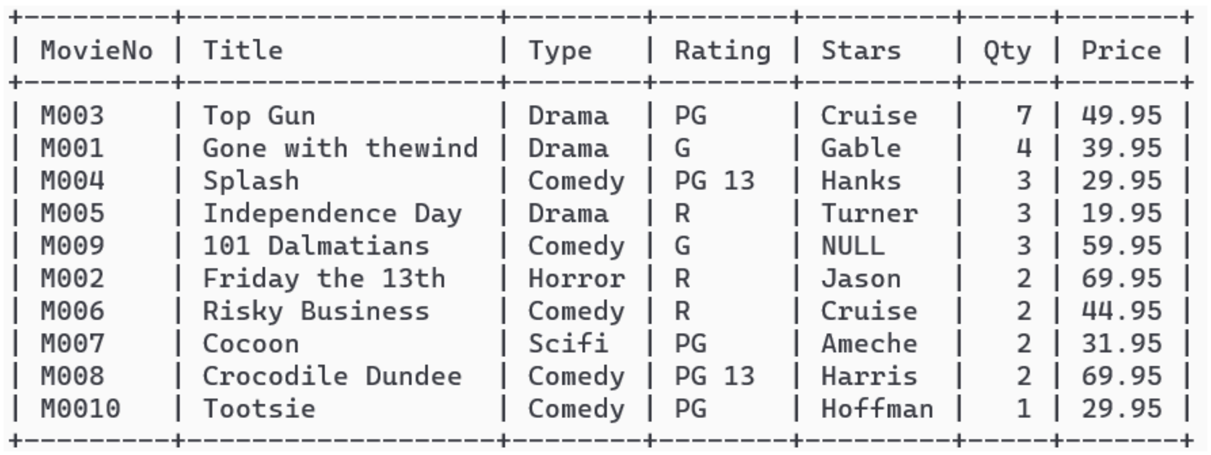
OUTPUT



```
+-------------------+-------+
| Title             | Price |
+-------------------+-------+
| Tootsie           | 29.95 |
| Splash            | 29.95 |
| Cocoon            | 31.95 |
| Gone with thewind | 39.95 |
| Risky Business    | 44.95 |
| Top Gun           | 49.95 |
| 101 Dalmatians    | 59.95 |
| Friday the 13th   | 69.95 |
| Crocodile Dundee  | 69.95 |
+-------------------+-------+
```

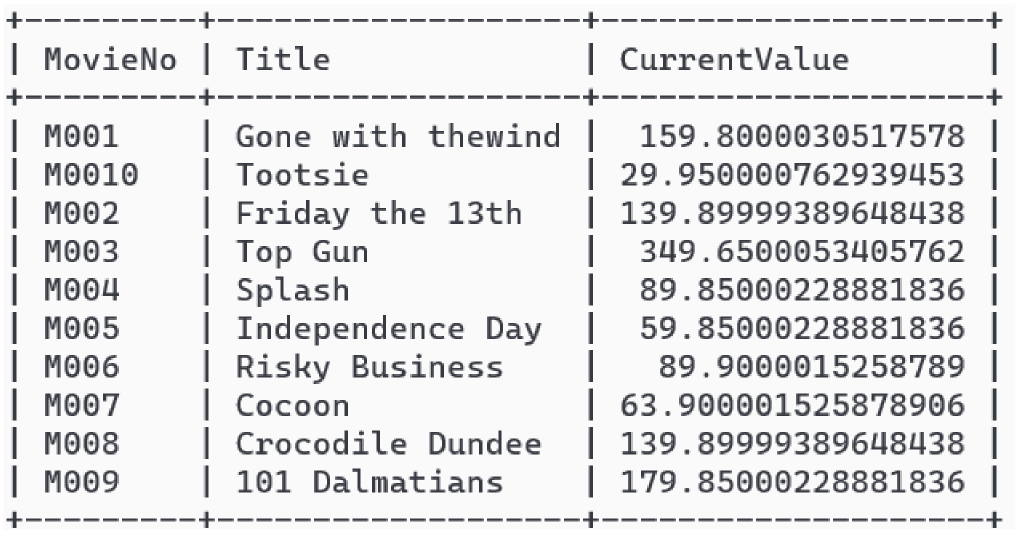
```
SELECT * FROM Movies ORDER BY Qty DESC;
```

OUTPUT



```
+---------+-------------------+--------+--------+---------+-----+-------+
| MovieNo | Title             | Type   | Rating | Stars   | Qty | Price |
+---------+-------------------+--------+--------+---------+-----+-------+
| M003    | Top Gun           | Drama  | PG     | Cruise  |   7 | 49.95 |
| M001    | Gone with thewind | Drama  | G      | Gable   |   4 | 39.95 |
| M004    | Splash            | Comedy | PG 13  | Hanks   |   3 | 29.95 |
| M005    | Independence Day  | Drama  | R      | Turner  |   3 | 19.95 |
| M009    | 101 Dalmatians    | Comedy | G      | NULL    |   3 | 59.95 |
| M002    | Friday the 13th   | Horror | R      | Jason   |   2 | 69.95 |
| M006    | Risky Business    | Comedy | R      | Cruise  |   2 | 44.95 |
| M007    | Cocoon            | Scifi  | PG     | Ameche  |   2 | 31.95 |
| M008    | Crocodile Dundee  | Comedy | PG 13  | Harris  |   2 | 69.95 |
| M0010   | Tootsie           | Comedy | PG     | Hoffman |   1 | 29.95 |
+---------+-------------------+--------+--------+---------+-----+-------+
SELECT MovieNo, Title, Qty*Price as CurrentValue FROM Movies;
```

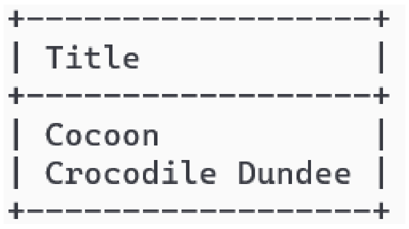
OUTPUT



```
+---------+-------------------+--------------------+
| MovieNo | Title             | CurrentValue       |
+---------+-------------------+--------------------+
| M001    | Gone with thewind |  159.8000030517578 |
| M0010   | Tootsie           | 29.950000762939453 |
| M002    | Friday the 13th   | 139.89999389648438 |
| M003    | Top Gun           |  349.6500053405762 |
| M004    | Splash            |  89.85000228881836 |
| M005    | Independence Day  |  59.85000228881836 |
| M006    | Risky Business    |   89.9000015258789 |
| M007    | Cocoon            | 63.900001525878906 |
| M008    | Crocodile Dundee  | 139.89999389648438 |
| M009    | 101 Dalmatians    | 179.85000228881836 |
+---------+-------------------+--------------------+
```

```
SELECT Title FROM Movies WHERE LEFT(Title,1) = 'C';
```

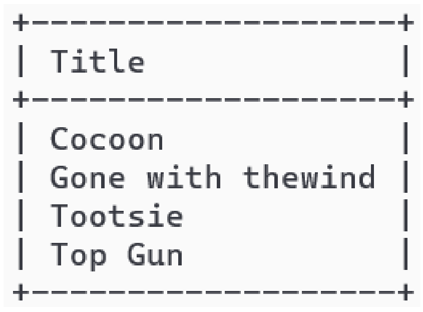
OUTPUT



```
+------------------+
| Title            |
+------------------+
| Cocoon           |
| Crocodile Dundee |
+------------------+
```

```
SELECT Title FROM Movies WHERE Title LIKE '_o%';
```

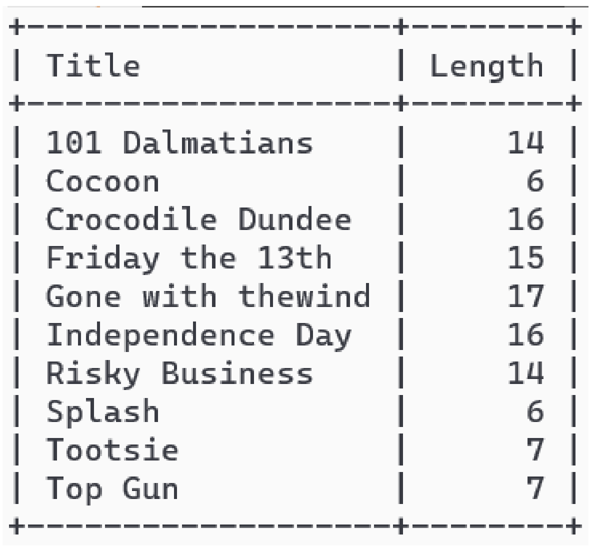
OUTPUT



```
+-------------------+
| Title             |
+-------------------+
| Cocoon            |
| Gone with thewind |
| Tootsie           |
| Top Gun           |
+-------------------+
```

```
SELECT Title, LENGTH(Title) as Length FROM Movies;
```

OUTPUT



```
+-------------------+--------+
| Title             | Length |
+-------------------+--------+
| 101 Dalmatians    |     14 |
| Cocoon            |      6 |
| Crocodile Dundee  |     16 |
| Friday the 13th   |     15 |
| Gone with thewind |     17 |
| Independence Day  |     16 |
| Risky Business    |     14 |
| Splash            |      6 |
| Tootsie           |      7 |
| Top Gun           |      7 |
+-------------------+--------+
```

B. Write SQL Queries for the following two tables.

RELATION: FACULTY

F_ID	Fname	Lname	Hire_date	Salary
102	Amit	Mishra	12-10-1998	12000
103	Nitin	Vyas	24-12-1994	8000
104	Rakshit	Soni	18-5-2001	14000
105	Rashmi	Malhotra	11-9-2004	11000
106	Sulekha	Srivastava	5-6-2006	10000

RELATION: Course

C_ID	F_ID	Cname	Fees
C21	102	Grid Computing	40000
C22	106	System Design	16000
C23	104	Computer Security	8000
C24	106	Human Biology	15000
C25	102	Computer Network	20000
C26	105	Visual Basic	6000

Create the above two tables with the following constraints.

RELATION: FACULTY

F_ID: Primary key

Fname and Lname: NOT NULL

RELATION: Course

C_ID – Primary key

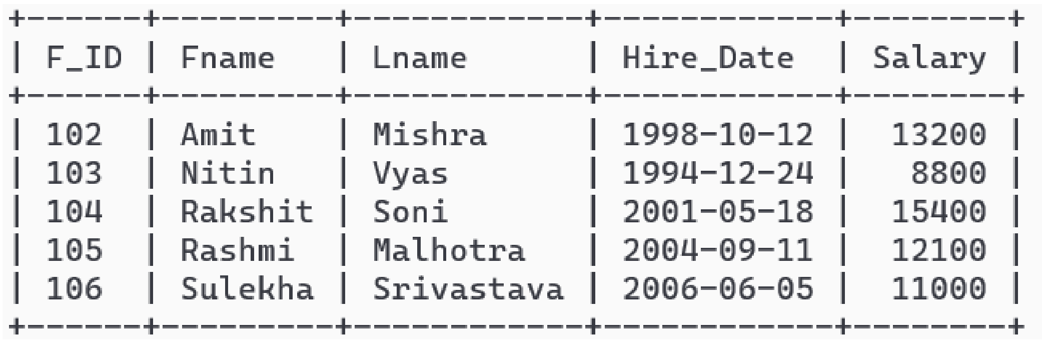
Cname – NOT NULL

F_ID: Foreign key of FACULTY (F_ID)

1. To display details of those Faculties whose salary is greater than 12000.
2. To display the details of courses whose fees are in the range of 15000 to 50000 (both values included).
3. To display details of those courses which are taught by 'Sulekha' in descending order of courses.
4. Increase the salary of all the faculty by 10 %
5. Delete the records of faculty 105.
6. Display the number of courses taught by each faculty

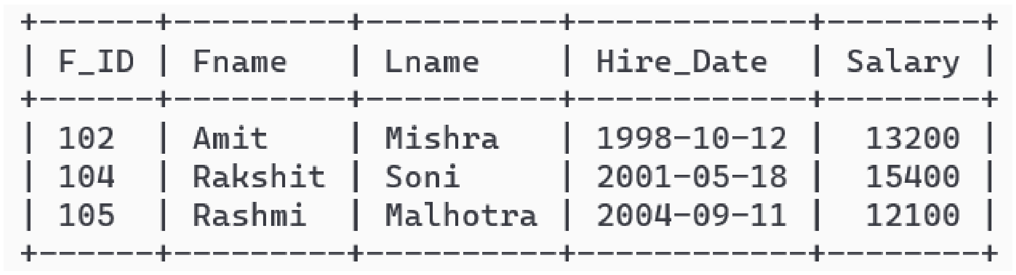
```
USE assignments;
CREATE TABLE Faculty(
    F_ID VARCHAR(3) PRIMARY KEY,
    Fname VARCHAR(10) NOT NULL,
    Lname VARCHAR(10) NOT NULL,
    Hire_Date DATE,
    Salary FLOAT);

INSERT INTO Faculty
VALUES
    ('102', 'Amit', 'Mishra', '1998-10-12', 12000),
    ('103', 'Nitin', 'Vyas', '1994-12-24', 8000),
    ('104', 'Rakshit', 'Soni', '2001-5-18', 14000),
    ('105', 'Rashmi', 'Malhotra', '2004-9-11', 11000),
    ('106', 'Sulekha', 'Srivastava', '2006-6-5', 10000);

CREATE TABLE Course(
    C_ID VARCHAR(3) PRIMARY KEY,
    F_ID VARCHAR(3),
    Cname VARCHAR(30) NOT NULL,
    Fees FLOAT,
    FOREIGN KEY (F_ID) REFERENCES Faculty(F_ID)
    ON DELETE CASCADE);


INSERT INTO Course
VALUES
    ('C21', '102', 'Grid Computing', 40000),
    ('C22', '106', 'System Design', 16000),
    ('C23', '104', 'Computer Security', 8000),
    ('C24', '106', 'Human Biology', 15000),
    ('C25', '102', 'Computer Network', 20000),
    ('C26', '105', 'Visual Basic', 6000);


USE assignments;

SELECT * FROM faculty;
```

<u>OUTPUT</u>



```
+------+---------+------------+------------+--------+
| F_ID | Fname   | Lname      | Hire_Date  | Salary |
+------+---------+------------+------------+--------+
| 102  | Amit    | Mishra     | 1998-10-12 |  13200 |
| 103  | Nitin   | Vyas       | 1994-12-24 |   8800 |
| 104  | Rakshit | Soni       | 2001-05-18 |  15400 |
| 105  | Rashmi  | Malhotra   | 2004-09-11 |  12100 |
| 106  | Sulekha | Srivastava | 2006-06-05 |  11000 |
+------+---------+------------+------------+--------+
SELECT * FROM Faculty WHERE Salary > 12000
```

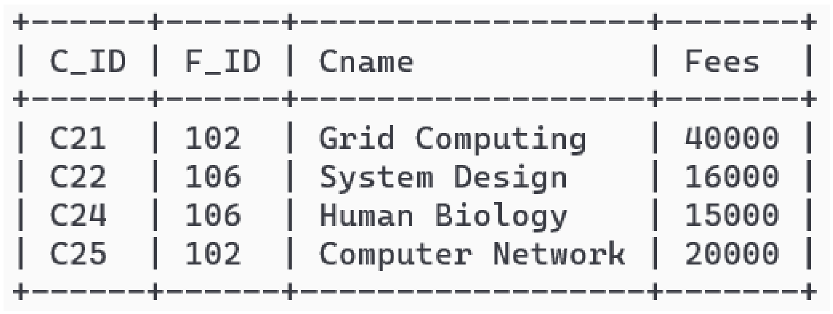
OUTPUT



```
+------+---------+----------+------------+--------+
| F_ID | Fname   | Lname    | Hire_Date  | Salary |
+------+---------+----------+------------+--------+
| 102  | Amit    | Mishra   | 1998-10-12 |  13200 |
| 104  | Rakshit | Soni     | 2001-05-18 |  15400 |
| 105  | Rashmi  | Malhotra | 2004-09-11 |  12100 |
+------+---------+----------+------------+--------+
```

```
SELECT * FROM Course WHERE (Fees BETWEEN 15000 AND 50000);
```

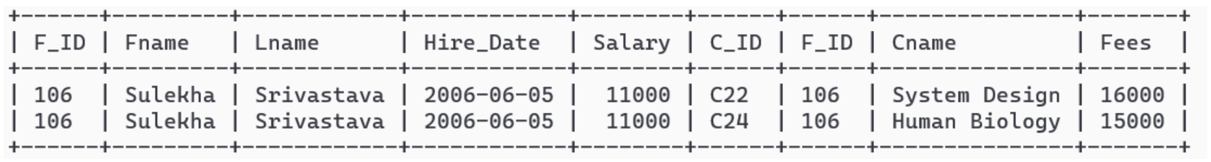
OUTPUT



```
+------+------+------------------+-------+
| C_ID | F_ID | Cname            | Fees  |
+------+------+------------------+-------+
| C21  | 102  | Grid Computing   | 40000 |
| C22  | 106  | System Design    | 16000 |
| C24  | 106  | Human Biology    | 15000 |
| C25  | 102  | Computer Network | 20000 |
+------+------+------------------+-------+
```

```
SELECT * FROM Faculty F, Course C WHERE F.F_ID = C.F_ID AND F.Fname =
'Sulekha' ORDER BY Cname DESC;
```

OUTPUT



```
+------+---------+------------+------------+--------+------+------+---------------+-------+
| F_ID | Fname   | Lname      | Hire_Date  | Salary | C_ID | F_ID | Cname         | Fees  |
+------+---------+------------+------------+--------+------+------+---------------+-------+
| 106  | Sulekha | Srivastava | 2006-06-05 |  11000 | C22  | 106  | System Design | 16000 |
| 106  | Sulekha | Srivastava | 2006-06-05 |  11000 | C24  | 106  | Human Biology | 15000 |
+------+---------+------------+------------+--------+------+------+---------------+-------+
```

```
UPDATE Faculty SET Salary = Salary + (0.1*Salary);
DELETE FROM Faculty WHERE F_ID = '105';
```

C. Write SQL query to create a table for following details:

TABLE: SALESPERSON

Attribute	Datatype	Size	Constraint
SID	Varchar	4	Primary Key
Name	Varchar	60	Not null
Phone	INT	10	Not null
DOB	Date		
Salary	Decimal	8,2	Not null

```
CREATE TABLE SALESPERSON (
    SID VARCHAR(4) PRIMARY KEY,
    Name VARCHAR(60) NOT NULL,
    Phone INT(10) NOT NULL,
    DOB DATE NOT NULL,
    Salary DECIMAL(8,2) NOT NULL);
```

TABLE: SPORTS

Attribute	Datatype	Size	Constraint
Sportsname	Varchar	20	Primary Key
No_of_teams	INT		Not null
No_of_games	INT		Not null
Prize_money	INT		Not null

```
CREATE TABLE SPORTS (
    Sportsname VARCHAR(20) PRIMARY KEY,
    No_of_teams INT NOT NULL,
    No_of_games INT NOT NULL,
    Prize_money INT NOT NULL);
```

D. For the following tables write SQL query for given questions:

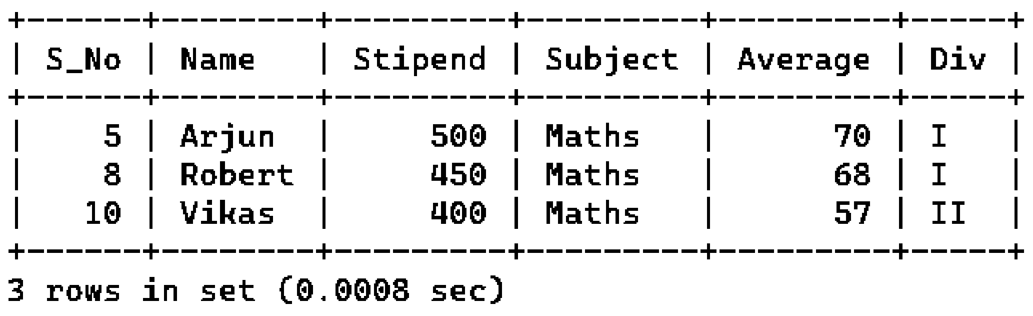
S.No	Name	Stipend	Subject	Average	Div
1	Karan	400	Physics	68	I
2	Diwakar	450	Comp. Sc.	68	I
3	Divya	300	Chemistry	62	I
4	Rekha	350	Physics	63	I

5	**Arjun**	**500**	**Maths**	**70**	**I**
6	**Sabina**	**400**	**Chemistry**	**55**	**II**
7	**John**	**250**	**Physics**	**64**	**I**
8	**Robert**	**450**	**Maths**	**68**	**I**
9	**Rubina**	**500**	**Comp. Sc.**	**62**	**I**
10	**Vikas**	**400**	**Maths**	**57**	**II**

```
CREATE TABLE Graduate (
    SNo INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Stipend INT NOT NULL,
    Subject VARCHAR(50) NOT NULL,
    Average INT NOT NULL,
    Div VARCHAR(2) NOT NULL);

INSERT INTO Graduate (SNo, Name, Stipend, Subject, Average, Div) VALUES
    (1, 'Karan', 400, 'Physics', 68, 'I'),
    (2, 'Diwakar', 450, 'Comp. Sc.', 68, 'I'),
    (3, 'Divya', 300, 'Chemistry', 62, 'I'),
    (4, 'Rekha', 350, 'Physics', 63, 'I'),
    (5, 'Arjun', 500, 'Maths', 70, 'I'),
    (6, 'Sabina', 400, 'Chemistry', 55, 'II'),
    (7, 'John', 250, 'Physics', 64, 'I'),
    (8, 'Robert', 450, 'Maths', 68, 'I'),
    (9, 'Rubina', 500, 'Comp. Sc.', 62, 'I'),
    (10, 'Vikas', 400, 'Maths', 57, 'II');
```

i) Write query to display all data where subject is maths

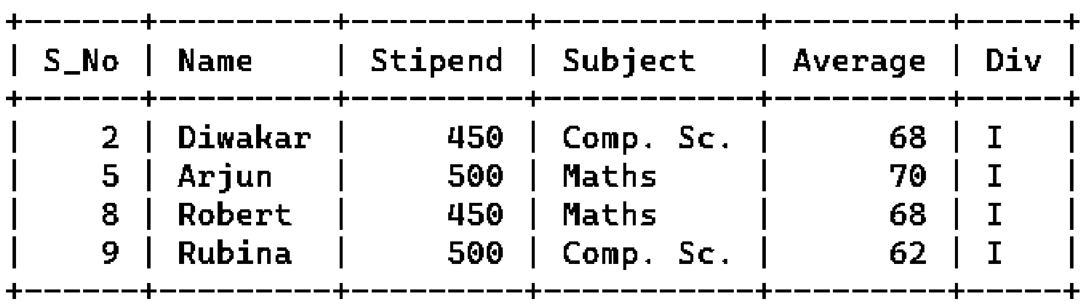
```
SELECT * FROM Graduate WHERE Subject = 'Maths';
```



```
+------+--------+---------+---------+---------+-----+
| S_No | Name   | Stipend | Subject | Average | Div |
+------+--------+---------+---------+---------+-----+
|    5 | Arjun  |     500 | Maths   |      70 | I   |
|    8 | Robert |     450 | Maths   |      68 | I   |
|   10 | Vikas  |     400 | Maths   |      57 | II  |
+------+--------+---------+---------+---------+-----+
3 rows in set (0.0008 sec)
```

ii) Write query to display all data from graduate table where stipend is greater than 400

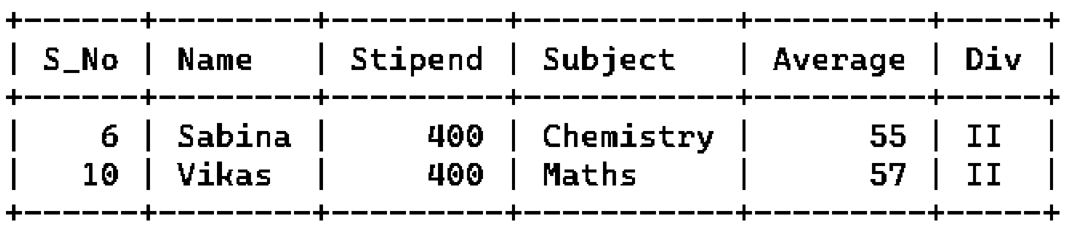
```
SELECT * FROM Graduate WHERE Stipend > 400;
```



```
+------+---------+---------+-----------+---------+-----+
| S_No | Name    | Stipend | Subject   | Average | Div |
+------+---------+---------+-----------+---------+-----+
|    2 | Diwakar |     450 | Comp. Sc. |      68 | I   |
|    5 | Arjun   |     500 | Maths     |      70 | I   |
|    8 | Robert  |     450 | Maths     |      68 | I   |
|    9 | Rubina  |     500 | Comp. Sc. |      62 | I   |
+------+---------+---------+-----------+---------+-----+
```

iii) Write query to display all data from graduate table where div is second

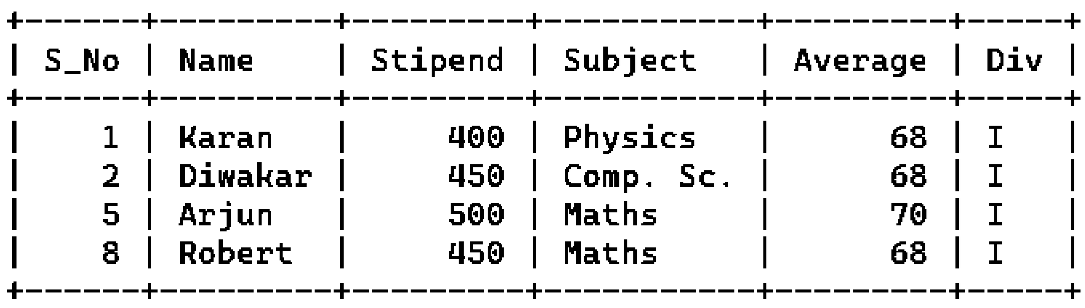
```
SELECT * FROM Graduate WHERE `Div` = 'II';
```



```
+------+--------+---------+-----------+---------+-----+
| S_No | Name   | Stipend | Subject   | Average | Div |
+------+--------+---------+-----------+---------+-----+
|    6 | Sabina |     400 | Chemistry |      55 | II  |
|   10 | Vikas  |     400 | Maths     |      57 | II  |
+------+--------+---------+-----------+---------+-----+
```

iv) Write query to display all data from graduate where average is greater than 65

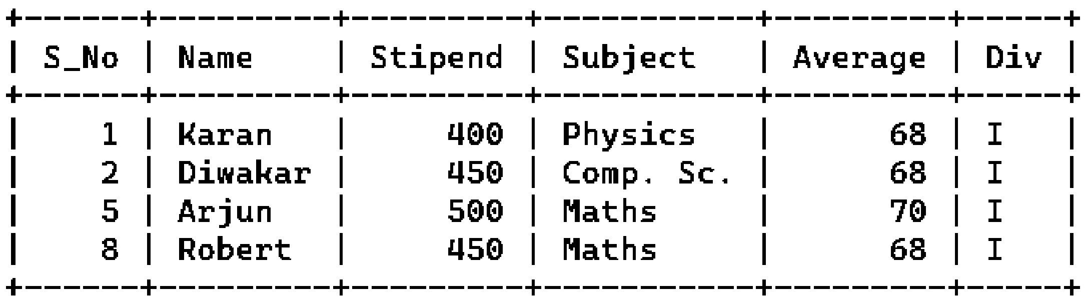
```
SELECT * FROM Graduate WHERE Average > 65;
```



```
+------+---------+---------+-----------+---------+-----+
| S_No | Name    | Stipend | Subject   | Average | Div |
+------+---------+---------+-----------+---------+-----+
|    1 | Karan   |     400 | Physics   |      68 | I   |
|    2 | Diwakar |     450 | Comp. Sc. |      68 | I   |
|    5 | Arjun   |     500 | Maths     |      70 | I   |
|    8 | Robert  |     450 | Maths     |      68 | I   |
+------+---------+---------+-----------+---------+-----+
```

v) Write query to display all data from graduate where average is greater than equal to 65

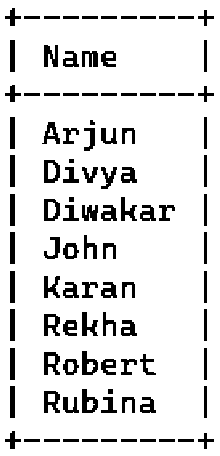
```
SELECT * FROM Graduate WHERE Average >= 65;
```



```
+------+---------+---------+-----------+---------+-----+
| S_No | Name    | Stipend | Subject   | Average | Div |
+------+---------+---------+-----------+---------+-----+
|    1 | Karan   |     400 | Physics   |      68 | I   |
|    2 | Diwakar |     450 | Comp. Sc. |      68 | I   |
|    5 | Arjun   |     500 | Maths     |      70 | I   |
|    8 | Robert  |     450 | Maths     |      68 | I   |
+------+---------+---------+-----------+---------+-----+
```

vi) List the names of those students who have obtained DIV 1 sorted by NAME.

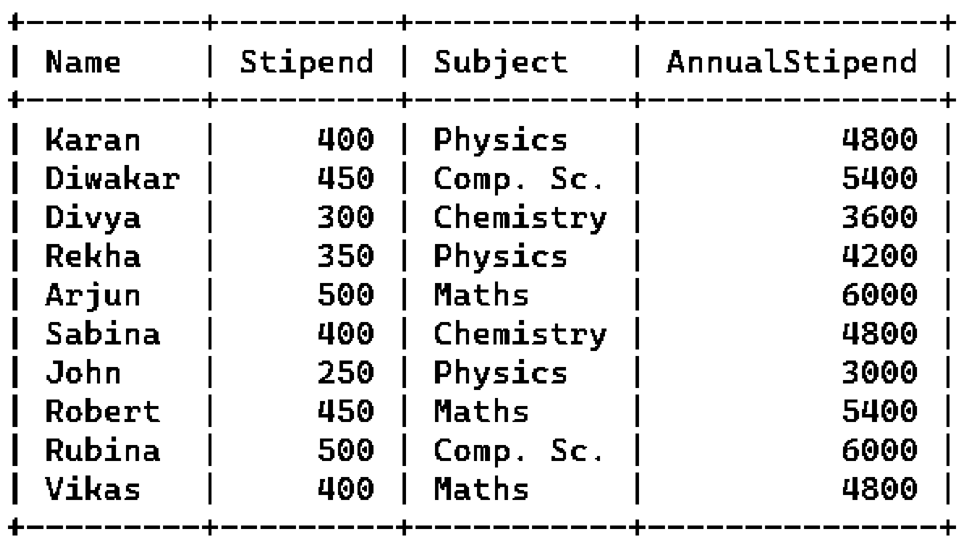
```
SELECT Name FROM Graduate WHERE `Div` = 'I' ORDER BY Name;
```



```
+---------+
| Name    |
+---------+
| Arjun   |
| Divya   |
| Diwakar |
| John    |
| Karan   |
| Rekha   |
| Robert  |
| Rubina  |
+---------+
```

vii) Display a report, listing NAME, STIPEND, SUBJECT and amount of stipend received in a year assuming that the STIPEND is paid every month.

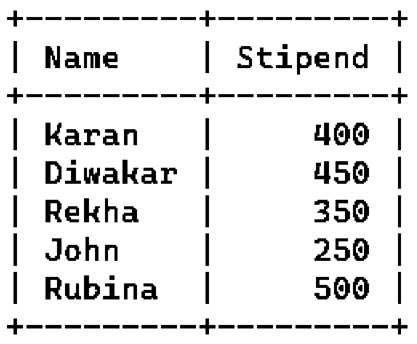
```
SELECT Name, Stipend, Subject, Stipend * 12 AS AnnualStipend FROM
Graduate;
```



```
+---------+---------+-----------+---------------+
| Name    | Stipend | Subject   | AnnualStipend |
+---------+---------+-----------+---------------+
| Karan   |     400 | Physics   |          4800 |
| Diwakar |     450 | Comp. Sc. |          5400 |
| Divya   |     300 | Chemistry |          3600 |
| Rekha   |     350 | Physics   |          4200 |
| Arjun   |     500 | Maths     |          6000 |
| Sabina  |     400 | Chemistry |          4800 |
| John    |     250 | Physics   |          3000 |
| Robert  |     450 | Maths     |          5400 |
| Rubina  |     500 | Comp. Sc. |          6000 |
| Vikas   |     400 | Maths     |          4800 |
+---------+---------+-----------+---------------+
```

viii) To display name of students, stipend who are either PHYSICS or COMPUTER SC graduates.

```
SELECT Name, Stipend FROM Graduate WHERE Subject IN ('Physics', 'Comp. Sc.');
```



```
+---------+---------+
| Name    | Stipend |
+---------+---------+
| Karan   |     400 |
| Diwakar |     450 |
| Rekha   |     350 |
| John    |     250 |
| Rubina  |     500 |
+---------+---------+
```

ix) To insert a new row in the GRADUATE table: 11, "KAJOL", 300, "Comp. Sc.", 75, 1

```
INSERT INTO Graduate (S_No, Name, Stipend, Subject, Average, `Div`) VALUES (11, 'KAJOL', 300, 'Computer Sc.', 75, 'I');
```

x) To display name and subject of graduates whose average is greater than 65.

```
SELECT Name, Subject FROM Graduate WHERE Average > 65;
```

```
+---------+--------------+
| Name    | Subject      |
+---------+--------------+
| Karan   | Physics      |
| Diwakar | Comp. Sc.    |
| Arjun   | Maths        |
| Robert  | Maths        |
| KAJOL   | Computer Sc. |
+---------+--------------+
```

xi) Display Name, stipend, subject where name has b as third character and a as last character.

```
SELECT Name, Stipend, Subject FROM Graduate WHERE SUBSTRING(Name, 3, 1) =
'b' AND RIGHT(Name, 1) = 'a';
```

```
+--------+---------+-----------+
| Name   | Stipend | Subject   |
+--------+---------+-----------+
| Sabina |     400 | Chemistry |
| Rubina |     500 | Comp. Sc. |
+--------+---------+-----------+
```

xii) Display all data from graduate in order or their names.

```
SELECT * FROM Graduate ORDER BY Name;
```

```
+------+---------+---------+--------------+---------+-----+
| S_No | Name    | Stipend | Subject      | Average | Div |
+------+---------+---------+--------------+---------+-----+
|    5 | Arjun   |     500 | Maths        |      70 | I   |
|    3 | Divya   |     300 | Chemistry    |      62 | I   |
|    2 | Diwakar |     450 | Comp. Sc.    |      68 | I   |
|    7 | John    |     250 | Physics      |      64 | I   |
|   11 | KAJOL   |     300 | Computer Sc. |      75 | I   |
|    1 | Karan   |     400 | Physics      |      68 | I   |
|    4 | Rekha   |     350 | Physics      |      63 | I   |
|    8 | Robert  |     450 | Maths        |      68 | I   |
|    9 | Rubina  |     500 | Comp. Sc.    |      62 | I   |
|    6 | Sabina  |     400 | Chemistry    |      55 | II  |
|   10 | Vikas   |     400 | Maths        |      57 | II  |
+------+---------+---------+--------------+---------+-----+
```

E. Consider the following Table:

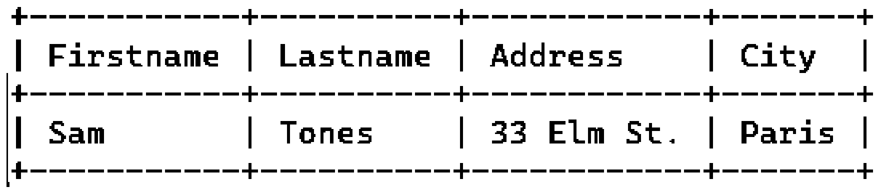
TABLE: EMPLOYEES

Empid	**Firstname**	**Lastname**	**Address**	**City**	**Designation**	**Salary**
010	Ravi	Kumar	Rajnagar	GZB	Manager	75000
105	Harry	Waltor	Gandhinagar	GZB	Manager	65000
152	Sam	Tones	33 Elm St.	Paris	Director	80000
215	Sarah	Ackerman	440 U.S. 110	Upton	Manager	75000
244	Manila	Sengupta	24 Friends Street	New Delhi	Clerk	50000
300	Robert	Samuel	9 Fifth Cross	Washington	Clerk	45000
335	Ritu	Tondon	Shastri Nagar	GZB	Clerk	40000
400	Rachel	Lee	121 Harrison St.	New York	Salesman	32000
441	Peter	Thompson	11 Red Road	Paris	Salesman	28000

```
CREATE TABLE EMPLOYEES (
    Empid INT PRIMARY KEY,
    Firstname VARCHAR(50) NOT NULL,
    Lastname VARCHAR(50) NOT NULL,
    Address VARCHAR(100),
    City VARCHAR(50),
    Designation VARCHAR(50),
    Salary INT);

INSERT INTO EMPLOYEES (Empid, Firstname, Lastname, Address, City,
Designation, Salary) VALUES
    (10, 'Ravi', 'Kumar', 'Rajnagar', 'GZB', 'Manager', 75000),
    (105, 'Harry', 'Waltor', 'Gandhinagar', 'GZB', 'Manager', 65000),
    (152, 'Sam', 'Tones', '33 Elm St.', 'Paris', 'Director', 80000),
    (215, 'Sarah', 'Ackerman', '440 U.S. 110', 'Upton', 'Manager', 75000),
    (244, 'Manila', 'Sengupta', '24 Friends Street', 'New Delhi', 'Clerk',
50000),
    (300, 'Robert', 'Samuel', '9 Fifth Cross', 'Washington', 'Clerk',
45000),
    (335, 'Ritu', 'Tondon', 'Shastri Nagar', 'GZB', 'Clerk', 40000),
    (400, 'Rachel', 'Lee', '121 Harrison St.', 'New York', 'Salesman',
32000),
    (441, 'Peter', 'Thompson', '11 Red Road', 'Paris', 'Salesman', 28000);
```

i) To show firstname, lastname, address and city of all employees living in paris and whose name has a as second character

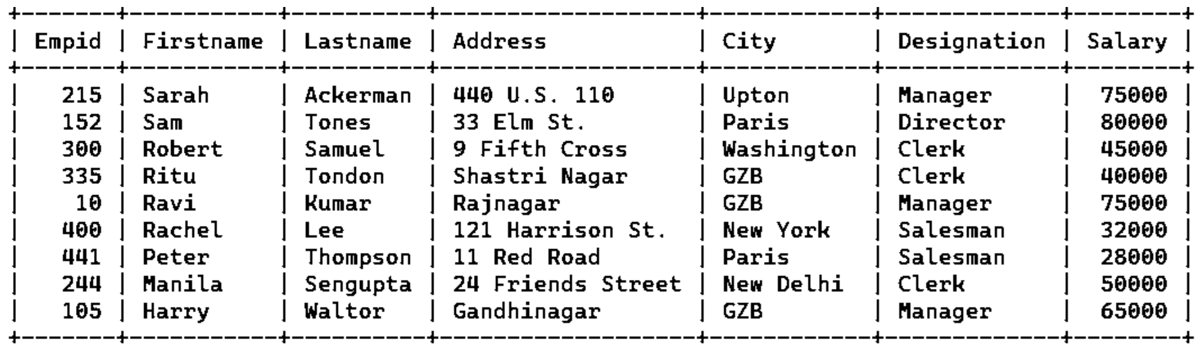
```
SELECT Firstname, Lastname, Address, City FROM EMPLOYEES
WHERE City = 'Paris' AND SUBSTRING(Firstname, 2, 1) = 'a';
```



```
+-----------+----------+------------+-------+
| Firstname | Lastname | Address    | City  |
+-----------+----------+------------+-------+
| Sam       | Tones    | 33 Elm St. | Paris |
+-----------+----------+------------+-------+
```

ii) To display the content of Employees table in descending order of Firstname.

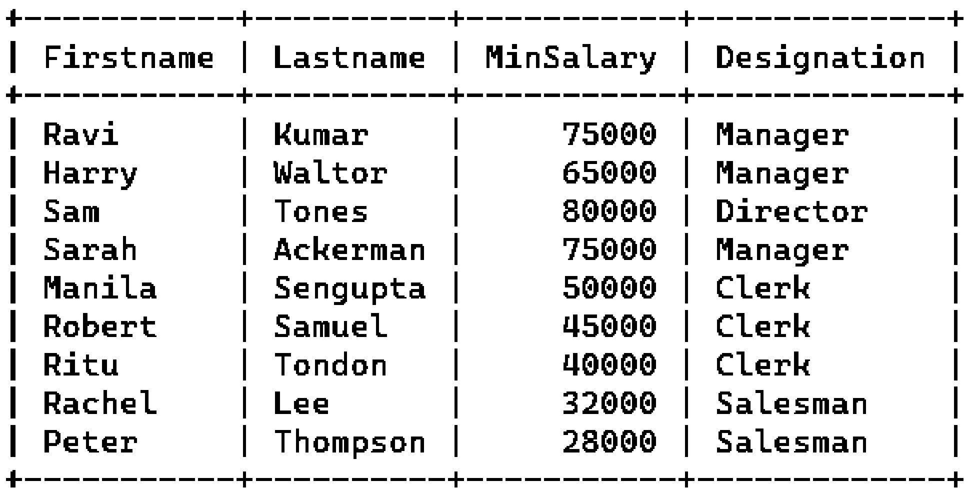
```
SELECT * FROM EMPLOYEES
ORDER BY Firstname DESC;
```



```
+-------+-----------+----------+-------------------+------------+-------------+--------+
| Empid | Firstname | Lastname | Address           | City       | Designation | Salary |
+-------+-----------+----------+-------------------+------------+-------------+--------+
|   215 | Sarah     | Ackerman | 440 U.S. 110      | Upton      | Manager     |  75000 |
|   152 | Sam       | Tones    | 33 Elm St.        | Paris      | Director    |  80000 |
|   300 | Robert    | Samuel   | 9 Fifth Cross     | Washington | Clerk       |  45000 |
|   335 | Ritu      | Tondon   | Shastri Nagar     | GZB        | Clerk       |  40000 |
|    10 | Ravi      | Kumar    | Rajnagar          | GZB        | Manager     |  75000 |
|   400 | Rachel    | Lee      | 121 Harrison St.  | New York   | Salesman    |  32000 |
|   441 | Peter     | Thompson | 11 Red Road       | Paris      | Salesman    |  28000 |
|   244 | Manila    | Sengupta | 24 Friends Street | New Delhi  | Clerk       |  50000 |
|   105 | Harry     | Waltor   | Gandhinagar       | GZB        | Manager     |  65000 |
+-------+-----------+----------+-------------------+------------+-------------+--------+
```

iii) To display the firstname,lastname,min salary of employee from the tables Employee of each designation

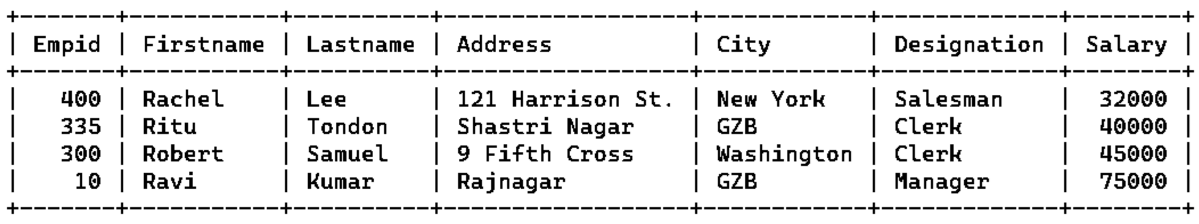
```
SELECT Firstname, Lastname, MIN(Salary) AS MinSalary, Designation
FROM EMPLOYEES
GROUP BY Firstname, Lastname, Designation;
```



```
+-----------+----------+-----------+-------------+
| Firstname | Lastname | MinSalary | Designation |
+-----------+----------+-----------+-------------+
| Ravi      | Kumar    |     75000 | Manager     |
| Harry     | Waltor   |     65000 | Manager     |
| Sam       | Tones    |     80000 | Director    |
| Sarah     | Ackerman |     75000 | Manager     |
| Manila    | Sengupta |     50000 | Clerk       |
| Robert    | Samuel   |     45000 | Clerk       |
| Ritu      | Tondon   |     40000 | Clerk       |
| Rachel    | Lee      |     32000 | Salesman    |
| Peter     | Thompson |     28000 | Salesman    |
+-----------+----------+-----------+-------------+
```

iv) To display all the data of employee table where first name starts with R in order of their salary.

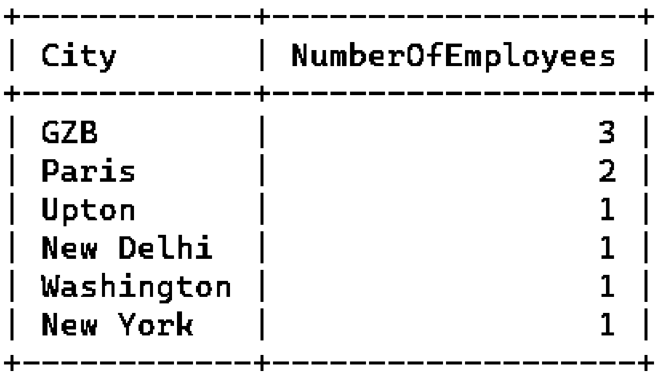
```
SELECT * FROM EMPLOYEES
WHERE Firstname LIKE 'R%' ORDER BY Salary;
```



```
+-------+-----------+----------+------------------+------------+-------------+--------+
| Empid | Firstname | Lastname | Address          | City       | Designation | Salary |
+-------+-----------+----------+------------------+------------+-------------+--------+
|   400 | Rachel    | Lee      | 121 Harrison St. | New York   | Salesman    |  32000 |
|   335 | Ritu      | Tondon   | Shastri Nagar    | GZB        | Clerk       |  40000 |
|   300 | Robert    | Samuel   | 9 Fifth Cross    | Washington | Clerk       |  45000 |
|    10 | Ravi      | Kumar    | Rajnagar         | GZB        | Manager     |  75000 |
+-------+-----------+----------+------------------+------------+-------------+--------+
```

v) To display number of employee from each city

```
SELECT City, COUNT(*) AS NumberOfEmployees
FROM EMPLOYEES
GROUP BY City;
```

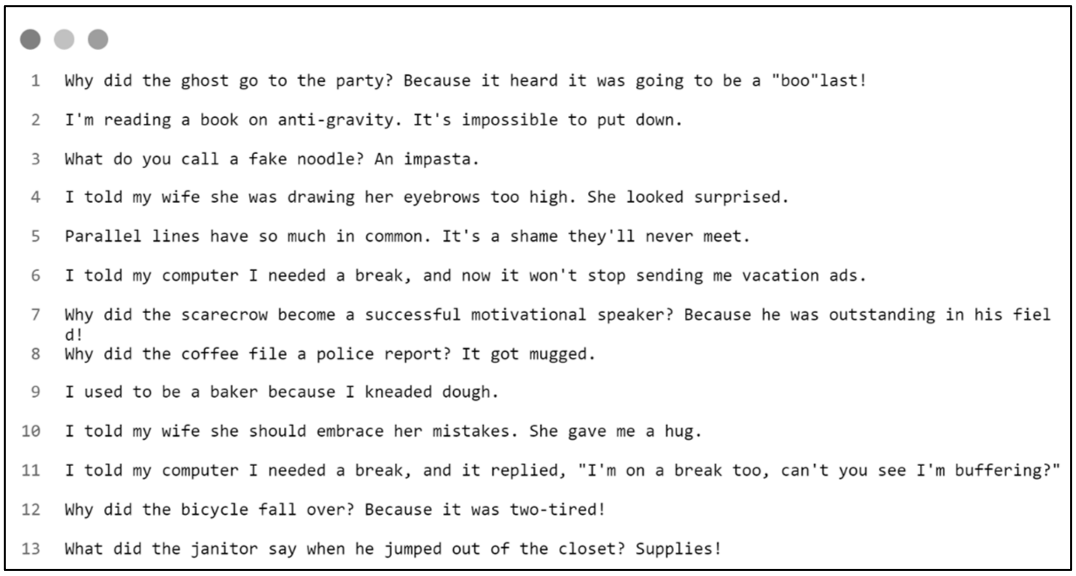


```
+------------+-------------------+
| City       | NumberOfEmployees |
+------------+-------------------+
| GZB        |                 3 |
| Paris      |                 2 |
| Upton      |                 1 |
| New Delhi  |                 1 |
| Washington |                 1 |
| New York   |                 1 |
+------------+-------------------+
```

FILE HANDLING

1. Write program to read a text file in python and print its content.

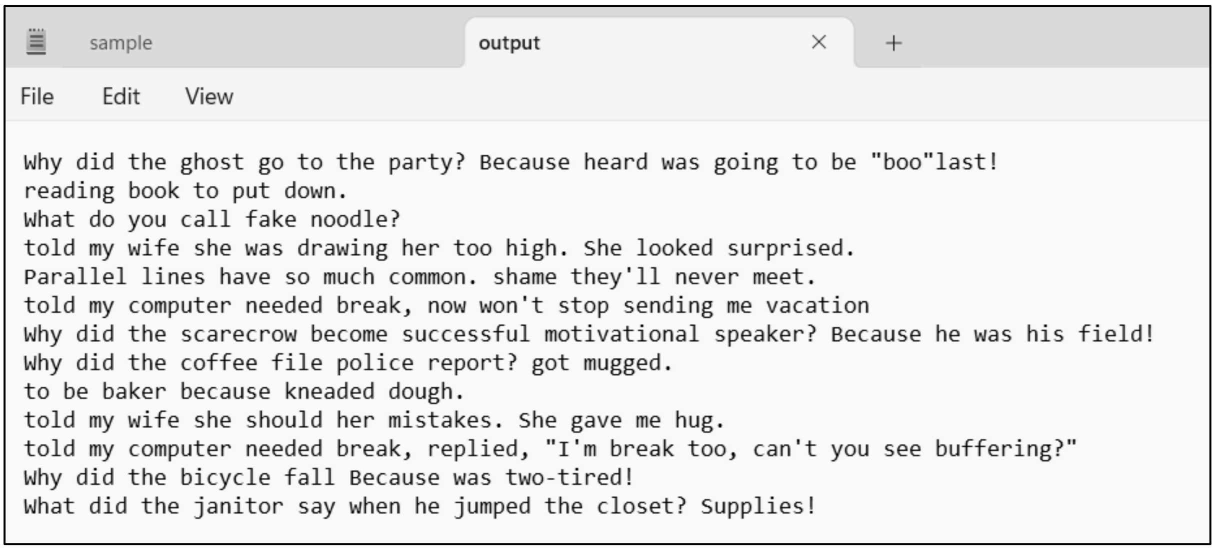
```
def read_file(file_path):

    try:
        with open(file_path, 'r') as file:
            content = file.read()
            print(content)
    except FileNotFoundError:
        print(f"File '{file_path}' not found.")
path = r'C:\Users\hp\Desktop\repera\coding\python\school xii\file
handling\sample.txt'
read_file(path)
```



```
Why did the ghost go to the party? Because it heard it was going to be a "boo"last!
I'm reading a book on anti-gravity. It's impossible to put down.
What do you call a fake noodle? An impasta.
I told my wife she was drawing her eyebrows too high. She looked surprised.
Parallel lines have so much in common. It's a shame they'll never meet.
I told my computer I needed a break, and now it won't stop sending me vacation ads.
Why did the scarecrow become a successful motivational speaker? Because he was outstanding in his fiel
d!
Why did the coffee file a police report? It got mugged.
I used to be a baker because I kneaded dough.
I told my wife she should embrace her mistakes. She gave me a hug.
I told my computer I needed a break, and it replied, "I'm on a break too, can't you see I'm buffering?"
Why did the bicycle fall over? Because it was two-tired!
What did the janitor say when he jumped out of the closet? Supplies!
```

2. Assuming that a text file named sample.txt already contains some text written into it, write a function named vowelwords(), that reads the file sample.txt and creates a new file named sample.txt, which shall contain only those words from the file sample.txt which don't start with an uppercase vowel (i.e., with 'A', 'E', 'I', 'O', 'U'). For example, if the file TEXT1.TXT contains Carry Umbrella and Overcoat When It rains Then the text file TEXT2.TXT shall contain Carry and When rains

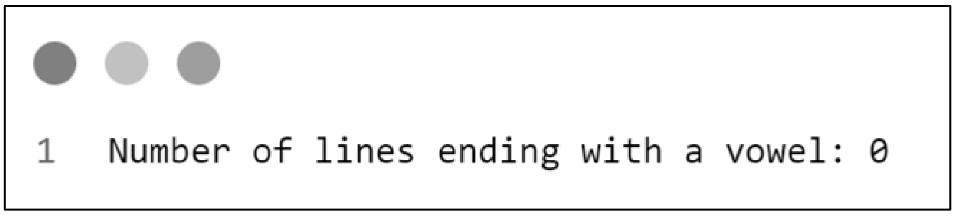
```
def vowelwords(input_file, output_file):
    vowels = {'A', 'E', 'I', 'O', 'U'}
    try:
        with open(input_file, 'r') as file1, open(output_file, 'w') as 
file2:
            for line in file1:
                words = line.split()
                non_vowel_words = [word for word in words if 
word[0].upper() not in vowels]
                file2.write(" ".join(non_vowel_words) + '\n')
    except FileNotFoundError:
        print(f"File '{input_file}' not found.")
samp_path = r"C:\Users\hp\Desktop\repera\coding\python\school xii\file 
handling\sample.txt"
vowelwords(samp_path, 'output.txt')
```



```
Why did the ghost go to the party? Because heard was going to be "boo"last!
reading book to put down.
What do you call fake noodle?
told my wife she was drawing her too high. She looked surprised.
Parallel lines have so much common. shame they'll never meet.
told my computer needed break, now won't stop sending me vacation
Why did the scarecrow become successful motivational speaker? Because he was his field!
Why did the coffee file police report? got mugged.
to be baker because kneaded dough.
told my wife she should her mistakes. She gave me hug.
told my computer needed break, replied, "I'm break too, can't you see buffering?"
Why did the bicycle fall Because was two-tired!
What did the janitor say when he jumped the closet? Supplies!
```

3. Write a function in PYTHON to count the number of lines ending with a vowel from a text file “sample.txt”.

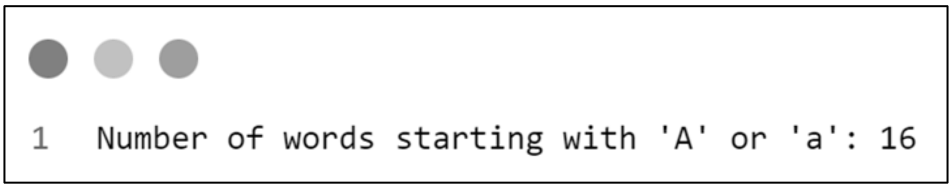
```
def count_lines_ending_with_vowel(file_path):
    vowels = {'a', 'e', 'i', 'o', 'u'}
    count = 0
    try:
        with open(file_path, 'r') as file:
            for line in file:
                line = line.strip()
                if line and line[-1].lower() in vowels:
                    count += 1
        print(f"Number of lines ending with a vowel: {count}")
    except FileNotFoundError:
        print(f"File '{file_path}' not found.")
path = r'C:\Users\hp\Desktop\repera\coding\python\school xii\file
handling\sample.txt'
count_lines_ending_with_vowel(path)
```



```
Number of lines ending with a vowel: 0
```

4. Write a function in PYTHON to count and display the number of words starting with alphabet ‘A’ or ‘a’ present in a text file “LINES.TXT”

```
def words_startwith(file_path):
    count = 0
    try:
        with open(file_path, 'r') as file:
            for line in file:
                words = line.split()
                for word in words:
                    if word and word[0].lower() == 'a':
                        count += 1
    except FileNotFoundError:
        print(f"File '{file_path}' not found.")
        return
    print(f"Number of words starting with 'A' or 'a': {count}")

path = r'C:\Users\hp\Desktop\repera\coding\python\school xii\file
handling\sample.txt'
words_startwith(path)
```



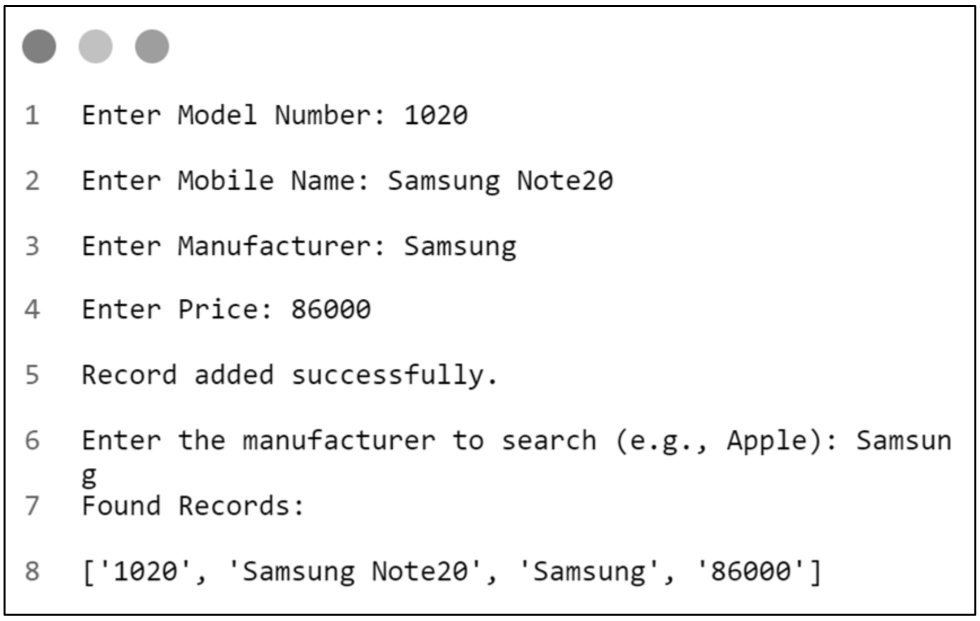
```
Number of words starting with 'A' or 'a': 16
```

5. Assuming the tuple Vehicle as follows: (vehicletype, no_of_wheels) Where vehicletype is a string and no_of_wheels is an integer. Write a function showfile() to read all the records present in an already existing binary file SPEED.DAT and display them on the screen, also count the number of records present in the file.

```
import pickle
def showfile(file_path):
    try:
        with open(file_path, 'rb+') as file:
            record_count = 0

            while True:
                try:
                    record = pickle.load(file)
                    vehicletype, no_of_wheels = record

                    print(f"Vehicle Type: {vehicletype}")
                    print(f"No. of Wheels: {no_of_wheels}")
                    print('-' * 30)

                    record_count += 1
                except EOFError:
                    break  # Reached the end of the file

            print(f"Number of Records: {record_count}")
    except FileNotFoundError:
        print(f"File '{file_path}' not found.")
        return
```

6. Write a program in Python that defines and calls the following user-defined functions:
   (i) AddRecord() – To accept and add data of Mobile phones to a CSV file 'Mobile_Phones.csv'. Each record consists of a list with field elements as ModelNo, MobileName, Manufacturer and Price to store model number, mobile name, manufacturer and price respectively.
   (ii) Find() – To search the records of mobiles manufactured by Samsung present in the CSV file named 'Mobile_Phones.csv'.

```
import csv

def AddRecord():
    model_no = input("Enter Model Number: ")
    mobile_name = input("Enter Mobile Name: ")
    manufacturer = input("Enter Manufacturer: ")
    price = input("Enter Price: ")

    record = [model_no, mobile_name, manufacturer, price]

    try:
        with open('Mobile_Phones.csv', 'a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(record)
        print("Record added successfully.")
    except Exception as e:
        print("Error:", e)

def Find():
    manufacturer_to_find = input("Enter the manufacturer to search (e.g.,
Apple): ")

    try:
        with open('Mobile_Phones.csv', 'r') as file:
            reader = csv.reader(file)
            found_records = [record for record in reader if
record[2].lower() == manufacturer_to_find.lower()]

        if found_records:
            print("Found Records:")
            for record in found_records:
                print(record)
        else:
            print("No records found for manufacturer:",
manufacturer_to_find)

    except Exception as e:
        print("Error:", e)

AddRecord()
Find()
```



```
Enter Model Number: 1020
Enter Mobile Name: Samsung Note20
Enter Manufacturer: Samsung
Enter Price: 86000
Record added successfully.
Enter the manufacturer to search (e.g., Apple): Samsung
Found Records:
['1020', 'Samsung Note20', 'Samsung', '86000']
```

CONNECTIVITY

1. Write the code to read the following record from the table named Employee and display only those records who have a Salary greater than 75000: EmpNo – integer EmpName – string Designation – string Salary – integer Note the following to establish connectivity between Python and MySQL:
   a. Username is root
   b. Password is sales_emp
   c. The table exists in a MySQLdatabase named "Office".

```
import mysql.connector

def display_records():
    username = 'root'
    password = 'sales_emp'
    database_name = 'Office'
    connection = None
    cursor = None
    try:
        connection = mysql.connector.connect(user=username,
password=password, database=database_name)
        cursor = connection.cursor()
        query = "SELECT * FROM Employee WHERE Salary > 75000"
        cursor.execute(query)
        records = cursor.fetchall()
        if records:
            print("Records with Salary greater than 75000:")
            for record in records:
                print(record)
        else:
            print("No records found.")

    except mysql.connector.Error as e:
        print("Error:", e)

    finally:
        if cursor:
            cursor.close()
        if connection:
            connection.close()

display_records()
```

2. Write the python code to establish connectivity between python and MYSQL to do following:
   a. Add a new column city in already created table salesman.
   b. Update salary of salesperson by 5% whose city is Mumbai.

```
import mysql.connector

def add_column_city():
    try:
        username = 'root'
        password = 'sales_emp'
        database_name = 'Office'
        connection = mysql.connector.connect(user=username,
password=password, database=database_name)
        cursor = connection.cursor()
        add_column_query = "ALTER TABLE salesman ADD COLUMN city
VARCHAR(255)"
        cursor.execute(add_column_query)
        connection.commit()
        print("New column 'city' added successfully.")

    except mysql.connector.Error as e:
        print("Error: ", e)

    finally:
        if cursor:
            cursor.close()
        if connection:
            connection.close()

def update_salary_mumbai():
    try:
        username = 'root'
        password = 'sales_emp'
        database_name = 'Office'
        connection = mysql.connector.connect(user=username,
password=password, database=database_name)
        cursor = connection.cursor()
        update_salary_query = "UPDATE salesman SET Salary = Salary * 1.05
WHERE city = 'Mumbai'"
        cursor.execute(update_salary_query)
        connection.commit()
        print("Salary updated successfully for salespersons in Mumbai.")
    except mysql.connector.Error as e:
        print("Error: ", e)

    finally:
        if cursor:
            cursor.close()
        if connection:
            connection.close()

add_column_city()
update_salary_mumbai()
```

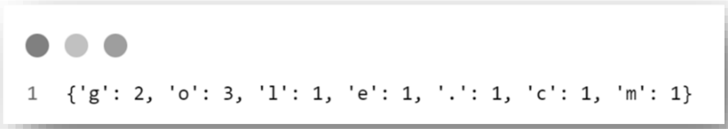
SUMMER VACATION HOLIDAY HOMEWORK 2023-24

CLASS-XII

SUBJECT- COMPUTER SCIENCE

1. Write a Python program to count the number of characters (character frequency) in a string.
   Sample String: ‘google.com'
   Expected Result: {'g': 2, 'o': 3, 'l': 1, 'e': 1, ‘.’: 1, 'c': 1, 'm': 1}

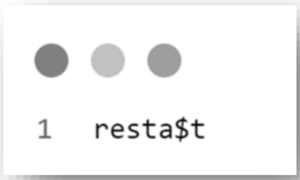
```
def frequency(s):
    d = {}
    for i in s:
        j = d.keys()
        if i in j:
            d[i]+=1
        else:
            d[i] = 1
    return d

print (frequency('google.com'))
```



```
{'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}
```

2. Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself.
   Sample String: 'restart'
   Expected Result: 'resta$t'

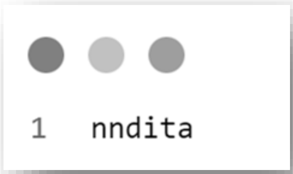
```
def change_dollar(s):
    x = s[0]
    a = s.replace(x, '$')
    s = x + a[1:]
    return s
print(change_dollar('restart'))
```



```
resta$t
```

3. Write a Python program to remove the nth index character from a nonempty string.

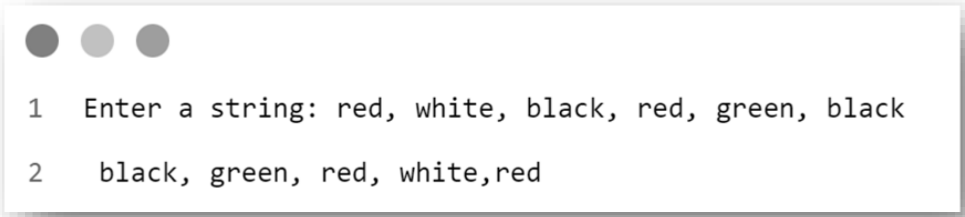
```
def removen(s, n):
    output = s[:n] + s[n+1:]
    return output

print(removen('nandita', 1))
```



```
nndita
```

4. Write a Python program that accepts a comma separated sequence of words as input and prints the unique words in sorted form (alphanumerically).
   Sample Words: red, white, black, red, green, black
   Expected Result: black, green, red, white, red

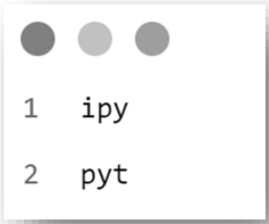
```
def sorter():
    s = input('Enter a string: ')
    l1 = s.split(',')
    x = sorted(set(l1))
    x = ','.join(x)
    print(x)

sorter()
```



```
Enter a string: red, white, black, red, green, black
 black, green, red, white,red
```

5. Write a Python function to get a string made of its first three characters of a specified string. If the length of the string is less than 3, then return the original string. Sample function and result: first_three('ipy') -> ipy first_three('python') -> pyt

```
def first_three(s):
    if len(s) < 3:
        print (s)
    else:
        print(s[:3])

first_three('ipy')
first_three('python')
```



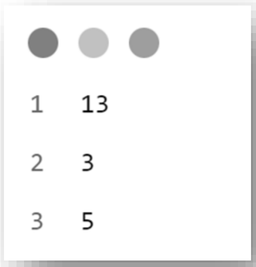
```
ipy
pyt
```

6. Write a Python program to check whether a string starts with specified characters. ***Note:*** In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code, or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

```
def spec_char(s,x):
    if s.lower().startswith(x.lower()) == True:
        return True
    else:
        return False

print(spec_char('Delhi Public School Ghaziabad Meerut Road', 'del'))
print(spec_char('Delhi Public School Ghaziabad Meerut Road', 'Road'))
```

7. Write a Python program to print the following floating numbers with no decimal places.

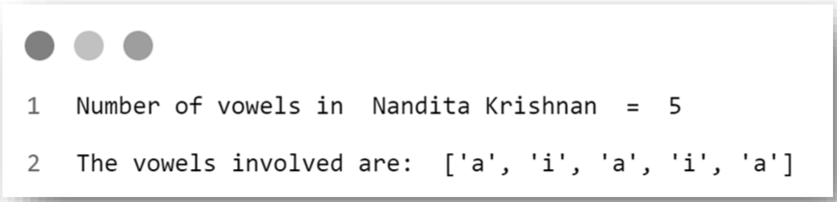
```
def integer(n):
    out = '{:.0f}'.format(n)
    print(out)

integer(12.99)
integer(3.1498361276)
integer(5.0)
```



```
13
3
5
```

8. Write a Python program to print the index of the character in a string. Sample string: Python Program
   Expected output:
   Current character P position at 0
   Current character r position at 1
   Current character o position at 2
9. Write a Python program to count and display the vowels of a given text.

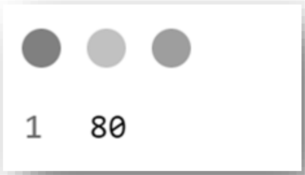
```
def vowel_check(s):
    vowels = 'aeiou'
    count = 0
    v = []
    for i in s:
        if i.lower() in vowels:
            count += 1
            v.append(i)

    print('Number of vowels in ', s, ' = ', count, '\nThe vowels involved
are: ', v)

vowel_check('Nandita Krishnan')
```



```
Number of vowels in  Nandita Krishnan  =  5
The vowels involved are:  ['a', 'i', 'a', 'i', 'a']
```

10. Write a Python program to sum all the items in a list.

```
def sum_list(*n):
    l = []
    for i in n:
        l.append(i)
    out = sum(l)
    print(out)

sum_list(10,20,30,12,8,0)
```

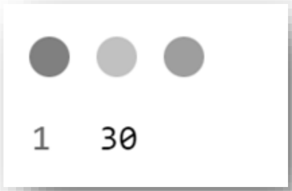


```
80
```

11. Write a Python program to get the largest number from a list.

```
def max_list(*n):
    l = []
    for i in n:
        l.append(i)
    out = max(l)
    print(out)
```

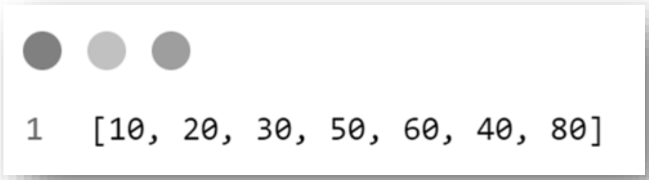
```
max_list(10,20,30,12,8,0)
```



```
30
```

12. Write a Python program to remove duplicates from a list.
a = [10,20,30,20,10,50,60,40,80,50,40]

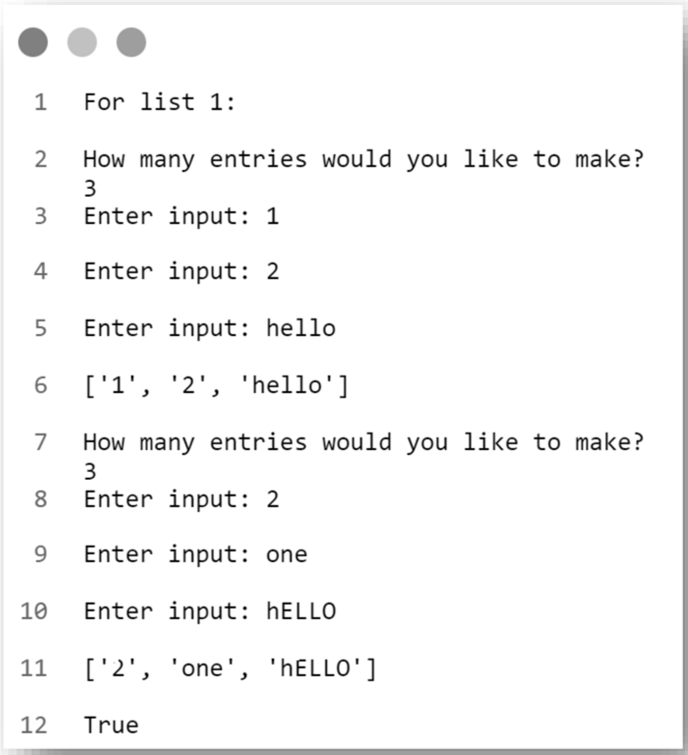
```
def remove_dupli(*n):
    l = []
    for i in n:
        if i not in l:
            l.append(i)
    print (l)

remove_dupli(10,20,30,20,10,50,60,40,80,50,40)
```



```
[10, 20, 30, 50, 60, 40, 80]
```

13. Write a Python function that takes two lists and returns True if they have at least one common member.

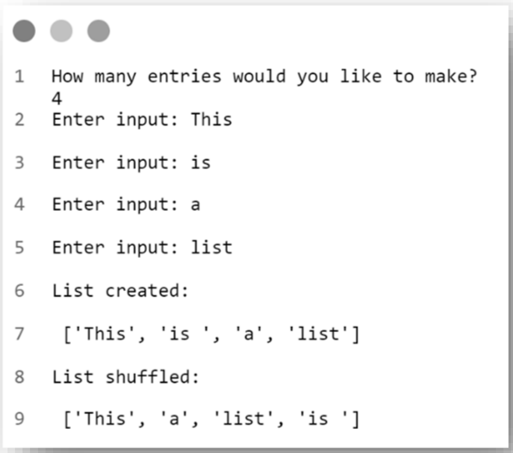
```
def create_list():
    n = int(input("How many entries would you like to make? "))
    l = []
    for i in range(n):
        j = input("Enter input: ")
        l.append(j)
    return l
def common_member():
    print("For list 1: ")
    l1 = create_list()
    print(l1)
    l2 = create_list()
    print(l2)
    for i in l1:
        for j in l2:
            if i == j:
                return True

print(common_member())
```



```
For list 1:
How many entries would you like to make?
3
Enter input: 1
Enter input: 2
Enter input: hello
['1', '2', 'hello']
How many entries would you like to make?
3
Enter input: 2
Enter input: one
Enter input: hELLO
['2', 'one', 'hELLO']
True
```

14. Write a Python program to shuffle and print a specified list.

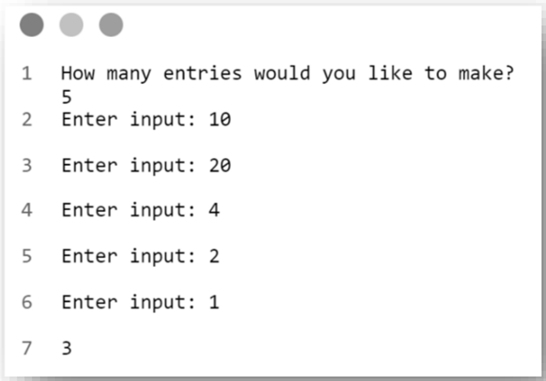
```
from random import shuffle
def shuffle_list():
    n = int(input("How many entries would you like to make? "))
    l = []
    for i in range(n):
        j = input("Enter input: ")
        l.append(j)
    print("List created: \n", l)
    shuffle(l)
    print("List shuffled: \n", l)
shuffle_list()
```



```
How many entries would you like to make?
4
Enter input: This

Enter input: is

Enter input: a

Enter input: list

List created:

 ['This', 'is ', 'a', 'list']

List shuffled:

 ['This', 'a', 'list', 'is ']
```

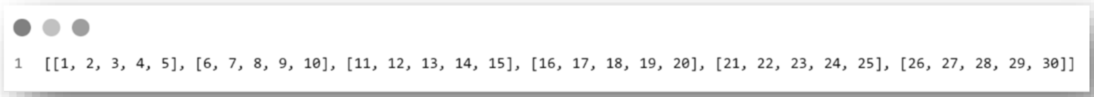
15. Write a Python program to count the number of elements in a list within a specified range.

```
def create_list():
    n = int(input("How many entries would you like to make? "))
    l = []
    for i in range(n):
        j = input("Enter input: ")
        l.append(j)
    return l
l1 = create_list()
def count_range(l, min, max):
    count = 0
    for x in l1:
        if min <= int(x) <= max:
            count += 1
    return count

print(count_range(l1,0,5))
```


```
How many entries would you like to make?
5
Enter input: 10

Enter input: 20

Enter input: 4

Enter input: 2

Enter input: 1

3
```

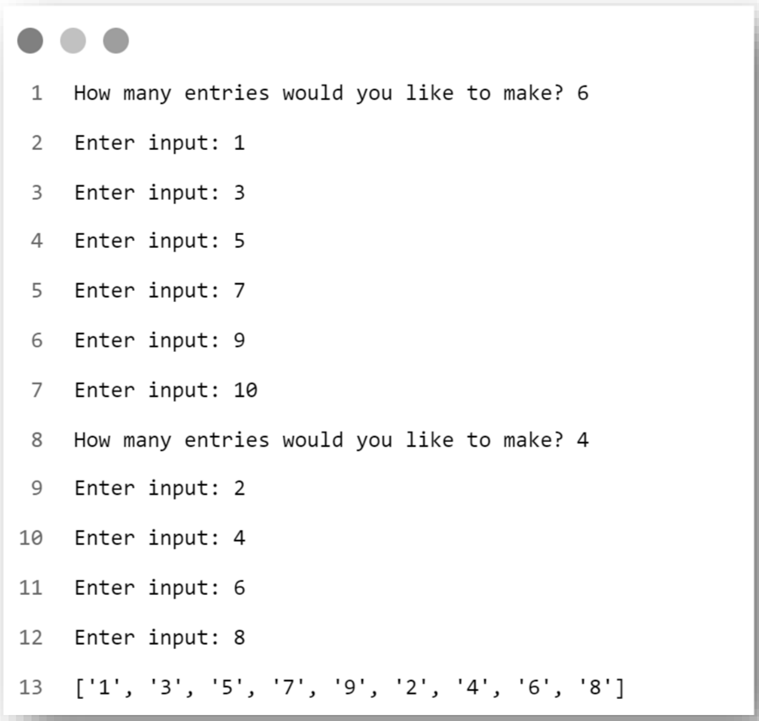
16. Write a Python program to generate groups of five consecutive numbers in a list.

```
def consec_num(n):
    l = [[5*i + j for j in range(1,6)] for i in range(n)]
    print(l)

consec_num(6)
```


```
[[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15], [16, 17, 18, 19, 20], [21, 22, 23, 24, 25], [26, 27, 28, 29, 30]]
```

17. Write a Python program to replace the last element in a list with another list.
    Sample data: [1, 3, 5, 7, 9, 10], [2, 4, 6, 8]
    Expected Output: [1, 3, 5, 7, 9, 2, 4, 6, 8]

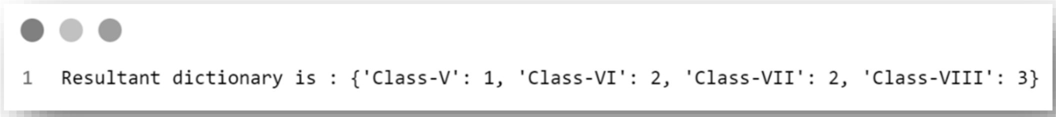
```
def create_list():
    n = int(input("How many entries would you like to make? "))
    l = []
    for i in range(n):
        j = input("Enter input: ")
        l.append(j)
    return l
l1 = create_list()
l2 = create_list()
def element_replace(l1, l2):
    '''l1 = [1, 3, 5, 7, 9, 10]
    l2 = [2, 4, 6, 8]'''
    l1[-1:] = l2
    print(l1)

element_replace(l1, l2)
```



```
How many entries would you like to make? 6
Enter input: 1
Enter input: 3
Enter input: 5
Enter input: 7
Enter input: 9
Enter input: 10
How many entries would you like to make? 4
Enter input: 2
Enter input: 4
Enter input: 6
Enter input: 8
['1', '3', '5', '7', '9', '2', '4', '6', '8']
```

18. Write a Python program to create a dictionary from two lists without losing duplicate values. Sample data: ['Class-V', 'Class-VI', 'Class-VII', 'Class-VIII'], [1, 2, 2, 3] Expected Output: defaultdict(<class 'set'>, {'Class-V':{1}, 'Class-VI':{2}, 'Class-VII':{2}, 'Class- VIII':{3}})

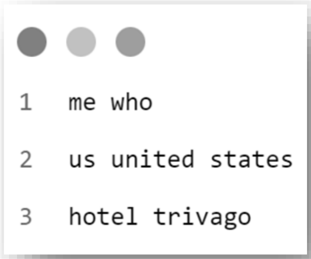
```
l1 =  ['Class-V', 'Class-VI', 'Class-VII', 'Class-VIII']
l2 = [1, 2, 2, 3]
d = {}
for key in l1:
    for value in l2:
        d[key] = value
        l2.remove(value)
        break

print("Resultant dictionary is : " + str(d))
```



```
Resultant dictionary is : {'Class-V': 1, 'Class-VI': 2, 'Class-VII': 2, 'Class-VIII': 3}
```

19. Write a Python program to iterate over dictionaries using for loops.

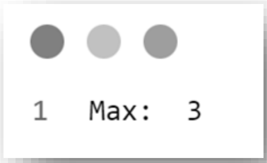
```
d = {'me': 'who', 'us': 'united states', 'hotel': 'trivago'}

for key, value in d.items():
    print(key, value)
```



```
me who
us united states
hotel trivago
```

20. Write a Python function to find the maximum of three numbers.

```
def max_three(n1,n2,n3):
    l = []
    l.append(n1)
    l.append(n2)
    l.append(n3)
    print('Max: ', max(l))


max_three(1, 2, 3)
```



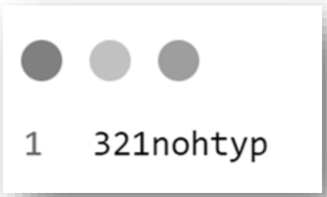
```
Max:  3
```

21. Write a Python function to sum all the numbers in a list. Sample List: (4, 6, 3, 5, 6)
    Expected Output: 24

Same as question 10

22. Write a Python function to reverse a string. Sample String: "python123" Expected Output: "321nohtyp"

```
def reverse_s(s):
    out = s[::-1]
    print(out)

reverse_s('python123')
```



```
321nohtyp
```

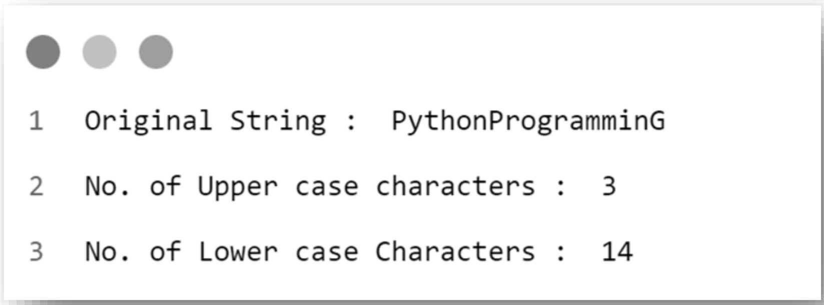
23. Write a Python function that accepts a string and calculates the number of uppercase letters and lowercase letters.

    Sample String: PythonProgramminG

    Expected Output: Original String: Python Programming No. of Uppercase characters: 3

    No. of Lowercase characters: 14

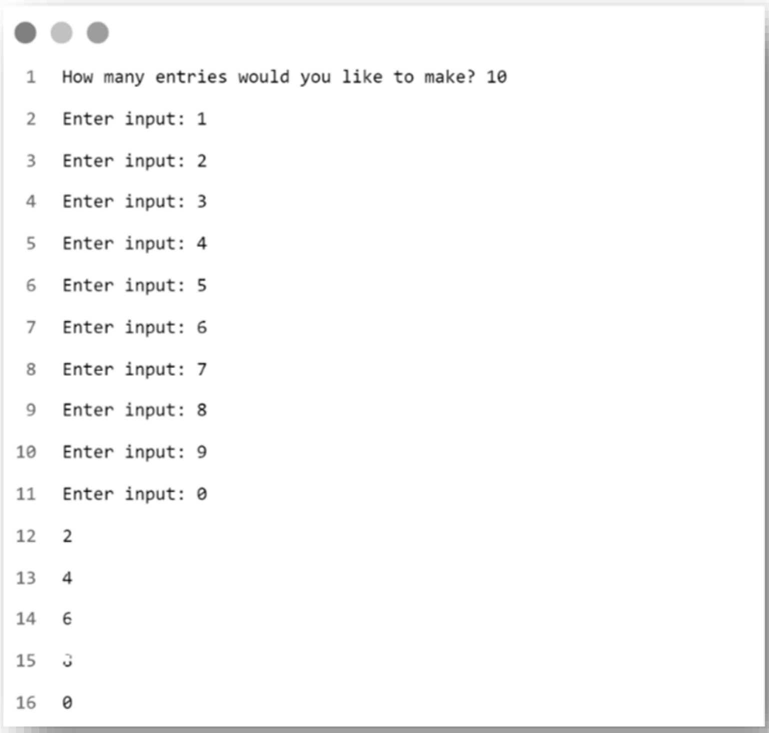
```
def counter(s):
    d={"upper":0, "lower":0}
    for c in s:
        if c.isupper():
            d["upper"]+=1
        elif c.islower():
            d["lower"]+=1
        else:
            pass
    print ("Original String : ", s)
    print ("No. of Upper case characters : ", d["upper"])
    print ("No. of Lower case Characters : ", d["lower"])

counter('PythonProgramminG')
```



```
Original String :  PythonProgramminG
No. of Upper case characters :  3
No. of Lower case Characters :  14
```

24. Write a Python program to print the even numbers from a given list.
    Sample List: [1, 2, 3, 4, 5, 6, 7,8, 9]
    Expected Result: [2, 4, 6, 8]

```
def create_list():
    n = int(input("How many entries would you like to make? "))
    l = []
    for i in range(n):
        j = input("Enter input: ")
        l.append(j)
    return l

l = create_list()
def even_numbers(l):
    for i in l:
        if int(i)%2 == 0:
            print (i)
```



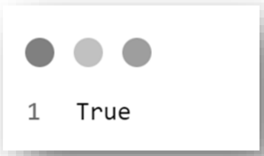
```
How many entries would you like to make? 10
Enter input: 1
Enter input: 2
Enter input: 3
Enter input: 4
Enter input: 5
Enter input: 6
Enter input: 7
Enter input: 8
Enter input: 9
Enter input: 0
2
4
6
8
0
```

```
even_numbers(l)
```

25. Write a Python function to check whether a number is perfect or not.

    According to Wikipedia, in number theory, a perfect number is a positive integer that is equal to the sum of its proper positive divisors, that is, the sum of its positive divisors excluding the number itself (also known as its aliquot sum). Equivalently, a perfect number is a number that is half the sum of all of its positive divisors (including itself). Example: The first perfect number is 6, because 1, 2, and 3 are its proper positive divisors, and 1 + 2 + 3 = 6. Equivalently, the number 6 is equal to half the sum of all its positive divisors: (1 + 2 + 3 + 6)/ 2 = 6. The next perfect number is 28
    = 1 + 2 + 4 + 7 + 14. This is followed by the perfect numbers 496 and 8128.

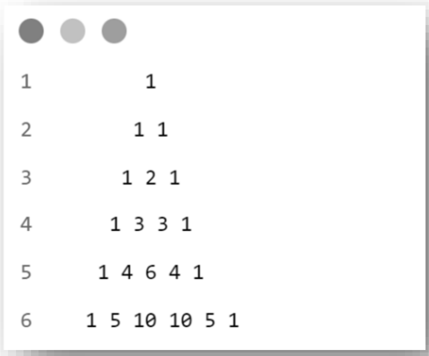
```
def perfect_number(n):
    sum = 0
    for x in range(1, n):
        if n % x == 0:
            sum += x
    return sum == n
print(perfect_number(6))
```



```
True
```

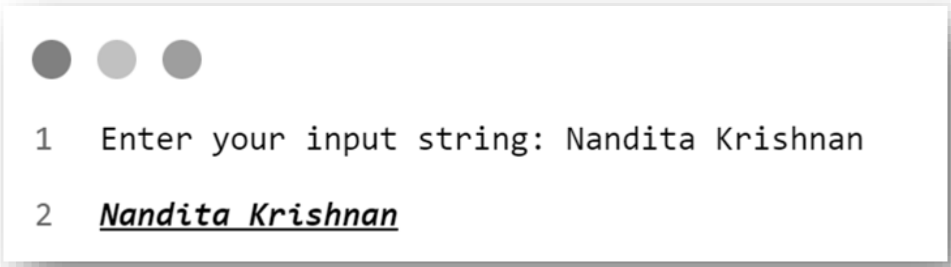
22. Write a Python function that prints the first n rows of Pascal's triangle. Note: Pascal's triangle is an arithmetic and geometric figure first imagined by Blaise Pascal.

```
def pascal(n):
    for i in range(1, n+1):
        for j in range(0, n-i+1):
            print(' ', end='')

        # first element is always 1
        C = 1
        for j in range(1, i+1):

            # first value in a line is always 1
            print(' ', C, sep='', end='')

            # using Binomial Coefficient
            C = C * (i - j) // j
        print()

pascal(6)
```



```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
```

```python
def bold(fn):
    def wrapper(text):
        return "<b>" + fn(text) + "</b>"
    return wrapper

def italic(fn):
    def wrapper(text):
        return "<i>" + fn(text) + "</i>"
    return wrapper

def underline(fn):
    def wrapper(text):
        return "<u>" + fn(text) + "</u>"
    return wrapper
@bold
@italic
@underline
def get_str(text):
    return text


i = input("Enter your input string: ")
print(get_str(i))
```



```
Enter your input string: Nandita Krishnan
```
<u>Nandita Krishnan</u>