

## Exercise 1:

# CREATING A MENU DRIVEN PROGRAM TO PERFORM ARITHMETIC OPERATIONS

### Objective:

Develop a Python program with a menu-driven interface enabling users to execute arithmetic operations (+, -, \*, /) based on their preferences.

```
1 def arithmetic_operation(x, y, operation):
2     return (
3         x + y if operation == '1' else
4         x - y if operation == '2' else
5         x * y if operation == '3' else
6         x / y if operation == '4' and y != 0 else
7         "Error: Invalid choice"
8     )
9
10 while True:
11     print("\nMenu:\n1. Add\n2. Subtract\n3. Multiply\n4. Divide\n5. Exit")
12     choice = input("Enter choice (1/2/3/4/5): ")
13
14     if choice == '5':
15         print("Exiting the program.")
16         break
17
18     if choice in ('1', '2', '3', '4'):
19         num1, num2 = float(input("Enter first number: ")), float(input("Enter second number: "))
20         print("Result:", arithmetic_operation(num1, num2, choice))
21     else:
22         print("Invalid choice. Please enter a valid option.")
23
```

```
Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter choice (1/2/3/4/5): 1
Enter first number: 50
Enter second number: 19
Result: 69.0
```

```
Menu:
1. Add
2. Subtract
```

### Outcome:

Consequently, the Python script has been executed, and the output has been duly validated.

## Exercise 2:

# CREATING A PYTHON PROGRAM TO DISPLAY FIBONACCI SERIES

### Objective:

To write a Python Program to display Fibonacci Series up to 'n' numbers

```
1 def fibonacci(n):
2     a, b = 0, 1
3     fib_series = [a for _ in range(n)]
4
5     for i in range(1, n):
6         a, b = b, a + b
7         fib_series[i] = a
8
9     return fib_series
10
11 n = int(input("Enter the number of terms for Fibonacci series: "))
12 if n > 0:
13     print(f"Fibonacci Series up to {n} terms: {fibonacci(n)}")
14 else:
15     print("Please enter a positive integer.")
```

### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

```
Enter the number of terms for Fibonacci series: 7
Fibonacci Series up to 7 terms: [0, 1, 1, 2, 3, 5, 8]
PS C:\Users\levono\OneDrive\Documents\School Project (Real)> |
```

## Exercise 3:

# CREATING A MENU DRIVEN PROGRAM TO FIND FACTORIAL AND SUM OF LIST OF NUMBERS USING FUNCTION

## Objective:

To write a menu driven Python Program to find Factorial and sum of list of numbers using function

```
1
2 def factorial(num):
3     return 1 if num in (0, 1) else num * factorial(num - 1)
4
5 def sum_of_list(numbers):
6     total = 0
7     for num in numbers:
8         total += float(num)
9     return total
10
11 while True:
12     print("\nMenu:\n1. Find Factorial\n2. Find Sum of List\n3. Exit")
13     choice = input("Enter choice (1/2/3): ")
14
15     if choice == '3':
16         print("Exiting the program.")
17         break
18
19     if choice == '1':
20         num = int(input("Enter a number to find its factorial: "))
21         print(f"Factorial of {num}: {factorial(num)}")
22
23     elif choice == '2':
24         num_list = input("Enter a list of numbers separated by space: ").split()
25         print(f"Sum of the list: {sum_of_list(num_list)}")
26
27     else:
28         print("Invalid choice. Please enter a valid option.")
```

**Outcome:**

Thus, the above Python program has been executed and the output is verified successfully

```
Menu:
1. Find Factorial
2. Find Sum of List
3. Exit
Enter choice (1/2/3): 1
Enter a number to find its factorial: 3
Factorial of 3: 6
```

```
Menu:
1. Find Factorial
2. Find Sum of List
3. Exit
Enter choice (1/2/3): 2
Enter a list of numbers separated by space: 2 3 17 54 28 53 7 44
Sum of the list: 208.0
```

## Exercise 4:

# CREATING A PYTHON PROGRAM TO IMPLEMENT RETURNING VALUE(S) FROM FUNCTION

## Objective:

To Write a Python program to define the function check(no1,no2) that take two numbers and Returns the number that has minimum ones digit

```
1 def check(num1,num2):
2     return num1 if (num1%10) > (num2%10) else num2
3
4 numbers = input("Enter two numbers separated by spaces ").split()
5 print(check(int(numbers[0]),int(numbers[1])))
```

### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

```
Enter two numbers separated by spaces92 29
29
```

## Exercise 5:

# CREATING A PYTHON PROGRAM TO IMPLEMENT MATHEMATICAL FUNCTIONS

### Objective:

To write a Python program to implement python mathematical functions to find: (i) To find Square of a Number. (ii) To find Log of a Number(i.e. Log10) (iii) To find Quad of a Number

```
1  import math
2
3  def find_square(number):
4      return number ** 2
5
6  def find_log10(number):
7      return math.log10(number)
8
9  def find_quad(number):
10     return number ** 4
11
12  num = float(input("Enter a number: "))
13
14  print(f"Square of {num}: {find_square(num)}")
15  print(f"Log10 of {num}: {find_log10(num)}")
16  print(f"Quad of {num}: {find_quad(num)}")
```

### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

```
Enter a number: 60
Square of 60.0: 3600.0
Log10 of 60.0: 1.7781512503836436
Quad of 60.0: 12960000.0
```

## Exercise 6:

### CREATING A PYTHON PROGRAM TO GENERATE RANDOM NUMBER BETWEEN 1 TO 6

### Objective:

To write a Python program to generate random number between 1 to 6 to simulate the dice.

```
1 import random
2
3 def roll_dice():
4     return random.randint(1, 6)
5
6 # Example usage:
7 result = roll_dice()
8 print(f"The dice rolled: {result}")
```

### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

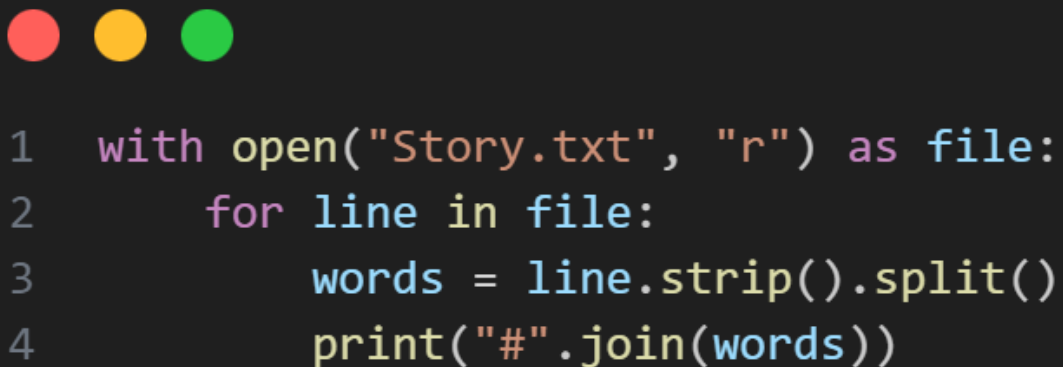
```
The dice rolled: 3
```

## Exercise 7:

### CREATING A PYTHON PROGRAM TO READ A TEXT FILE LINE BY LINE AND DISPLAY EACH WORD SEPARATED BY '#'

#### Objective:

To write a Python Program to Read a text file "Story.txt" line by line and display each word separated by '#'.



```
1  with open("Story.txt", "r") as file:
2      for line in file:
3          words = line.strip().split()
4          print("#".join(words))
```

#### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

## Exercise 8:

### CREATING A PYTHON PROGRAM TO READ A TEXT FILE AND DISPLAY THE NUMBER OF VOWELS/CONSONANTS/LOWER CASE/ UPPER CASE CHARACTERS.

#### Objective:

To write a Python Program to read a text file "Story.txt" and displays the number of Vowels/ Consonants/ Lowercase / Uppercase/characters in the file.

```
1 with open("chandransh.txt", "r") as file:
2     content = file.read()
3
4 vowels = sum(1 for char in content if char.lower() in "aeiou" if char.isalpha())
5 consonants = sum(1 for char in content if char.isalpha() and char.lower() not in "aeiou")
6 lowercase = sum(1 for char in content if char.islower())
7 uppercase = sum(1 for char in content if char.isupper())
8 total_chars = len(content)
9
10 print(f"Number of vowels: {vowels}")
11 print(f"Number of consonants: {consonants}")
12 print(f"Number of lowercase letters: {lowercase}")
13 print(f"Number of uppercase letters: {uppercase}")
14 print(f"Total number of characters: {total_chars}")
```

### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

## Exercise 9:

# CREATING PYTHON PROGRAM TO DISPLAY SHORT WORDS FROM A TEXT FILE

### Objective:

To Write a method Disp() in Python, to read the lines from poem.txt and display those words which are less than 5 characters.



```

1  def Disp(file_path):
2      try:
3          with open(file_path, 'r') as file:
4              for line in file:
5                  words = line.strip().split()
6                  short_words = [word for word in words if len(word) < 5]
7                  print("#".join(short_words))
8      except FileNotFoundError:
9          print(f"File '{file_path}' not found.")
10
11 # Example usage:
12 Disp("poem.txt")

```

### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

## Exercise 10:

### CREATING A PYTHON PROGRAM TO COPY PARTICULAR LINES OF A TEXT FILE INTO AN ANOTHER TEXT FILE

### Objective:

To write a python program to read lines from a text file "Sample.txt" and copy those lines into another file which are starting with an alphabet 'a' or 'A'.

```

1  input_file_path = "Sample.txt"
2  output_file_path = "Output.txt"
3
4  try:
5      with open(input_file_path, 'r') as input_file, open(output_file_path, 'w') as output_file:
6          for line in input_file:
7              if line.strip().lower().startswith('a'):
8                  output_file.write(line)
9              print(f"Lines starting with 'a' or 'A' copied to {output_file_path}")
10 except FileNotFoundError:
11     print(f"File '{input_file_path}' not found.")
12 except Exception as e:
13     print(f"An error occurred: {e}")

```

## Outcome:

Thus, the above Python program has been executed and the output is verified successfully

## Exercise 11:

# CREATING A PYTHON PROGRAM TO CREATE AND SEARCH RECORDS IN BINARY FILE

## Objective:

To write a Python Program to Create a binary file with roll number and name. Search for a given roll number and display the name, if not found display appropriate message.

```
1 import pickle
2
3 def write_binary_file(file_path, data):
4     with open(file_path, 'wb') as file:
5         pickle.dump(data, file)
6
7 def search_by_roll_number(file_path, search_roll_number):
8     return next((name for roll, name in pickle.load(open(file_path, 'rb')) if roll == search_roll_number), None)
9
10 # Example usage:
11 file_path, data = "student_data.pkl", [(101, 'Chandu'), (102, 'Vivek'), (103, 'Vedansh')]
12 write_binary_file(file_path, data)
13
14 search_roll_number = int(input("Enter the roll number to search: "))
15 result = search_by_roll_number(file_path, search_roll_number)
16
17 print(f"Name for Roll Number {search_roll_number}: {result}" if result else f"Roll Number {search_roll_number} not found.")
```

## Outcome:

Thus, the above Python program has been executed and the output is verified successfully

## Exercise 12:

### CREATING A PYTHON PROGRAM TO CREATE AND UPDATE/MODIFY RECORDS IN BINARY FILE

#### Objective:

To write a Python Program to Create a binary file with roll number, name, mark and update/modify the mark for a given roll number.

```
1 import pickle
2
3 def write_binary_file(file_path, data):
4     pickle.dump(data, open(file_path, 'wb'))
5
6 def update_mark(file_path, roll_to_update, new_marks):
7     data = [(r, n, new_marks) if r == roll_to_update else (r, n, m) for r, n, m in pickle.load(open(file_path, 'rb'))]
8     pickle.dump(data, open(file_path, 'wb'))
9
10 # Example usage:
11 file_path, data_to_write = "student_data.pkl", [(101, 'Chandrash', 85), (102, 'Chunnilal', 92), (103, 'KattarSunni', 78)]
12 write_binary_file(file_path, data_to_write)
13
14 roll_to_update, new_marks = int(input("Enter the roll number to update marks: ")), int(input("Enter the new marks: "))
15 update_mark(file_path, roll_to_update, new_marks)
16
17 print(f"Updated marks for Roll Number {roll_to_update}.")
18
19
```

#### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

```
Enter the roll number to update marks: 101
Enter the new marks: 1001
```

## Exercise 13:

### CREATING A PYTHON PROGRAM TO CREATE AND SEARCH EMPLOYEE'S RECORD IN CSV FILE.

## Objective:

To write a Python program Create a CSV file to store Empno, Name, Salary and search any Empno and display Name, Salary and if not found display appropriate message

```
1 import csv
2
3 def write_csv(file_path, data):
4     with open(file_path, 'w', newline='') as file:
5         csv.writer(file).writerows([["Empno", "Name", "Salary"]] + data)
6
7 def search_by_empno(file_path, search_empno):
8     return next(((name, salary) for empno, name, salary in csv.reader(open(file_path)) if empno == search_empno), (None, None))
9
10 # Example usage:
11 file_path, data_to_write = "employee_data.csv", [(["E101", "BhupendraJogi", "50000"), ("E102", "Rekha", "60000"), ("E103", "LordPuneet", "75000")]
12 write_csv(file_path, data_to_write)
13
14 search_empno = input("Enter the Empno to search: ")
15 name, salary = search_by_empno(file_path, search_empno)
16
17 print(f"Name: {name}\nSalary: {salary}" if name is not None else f"Empno {search_empno} not found.")
```

## Outcome:

Thus, the above Python program has been executed and the output is verified successfully

```
Enter the Empno to search: E102
Name: KattarSunni
```

## Exercise 14:

### : CREATING A PYTHON PROGRAM TO IMPLEMENT STACK OPERATIONS(LIST)

## Objective:

To write a Python program to implement Stack using a list data-structure, to perform the following operations:

- (i) To Push an object containing Doc\_ID and Doc\_name of doctors who specialize in "ENT" to the stack.
- (ii) To Pop the objects from the stack and display them.

(iii) To display the elements of the stack (after performing PUSH or POP)

```
1  stack = []
2
3  def push(doc_id, doc_name, specialization):
4      if specialization == "ENT": stack.append({"Doc_ID": doc_id, "Doc_Name": doc_name})
5
6  def pop():
7      if stack:
8          popped_doc = stack.pop()
9          print(f"Doctor with ID {popped_doc['Doc_ID']} and Name {popped_doc['Doc_Name']} popped from the stack.")
10     else:
11         print("Stack is empty. Cannot pop.")
12
13  def display_stack():
14      print("Elements of the stack:" if stack else "Stack is empty.")
15      [print(f"Doc_ID: {doc['Doc_ID']}, Doc_Name: {doc['Doc_Name']}") for doc in stack]
16
17  # Example usage:
18  push(1, "KattarSunni", "ENT")
19  push(2, "Migga", "Cardiology")
20  push(3, "BediChai", "ENT")
21
22  display_stack()
23
24  pop()
25  pop()
26
27  display_stack()
```

### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

```
Elements of the stack:
Doc_ID: 1, Doc_Name: KattarSunni
Doc_ID: 3, Doc_Name: BediChai
Doctor with ID 3 and Name BediChai popped from the stack.
Doctor with ID 1 and Name KattarSunni popped from the stack.
Stack is empty.
```

## Exercise 15:

# CREATING A PYTHON PROGRAM TO IMPLEMENT STACK OPERATIONS(Dictionary)

### Objective:

To Write a program, with separate user-defined functions to perform the following operations:

(i) To Create a function Push(Stk,D) Where Stack is an empty list and D is Dictionary of Items. from this Dictionary Push the keys (name of the student) into a stack, where the corresponding value (marks) is greater than 70.

(ii) To Create a Function Pop(Stk) , where Stk is a Stack implemented by a list of student names. The function returns the items deleted from the stack.

(iii) To display the elements of the stack (after performing PUSH or POP).

```
1 def push(stack, data):
2     stack.extend(name for name, marks in data.items() if marks > 70)
3     print("Push operation completed.")
4
5 def pop(stack):
6     popped_items = stack[-5:]
7     del stack[-5:]
8     return popped_items
9
10 def display_stack(stack):
11     print("Elements of the stack:" if stack else "Stack is empty.")
12     print(stack)
13
14 # Example usage:
15 stack = []
16 student_data = {"KattarSunni": 80, "BediChai": 65, "Ligma": 90, "AdityaKumar": 75, "Chunnilal": 68, "Duryodhan": 82}
17
18 push(stack, student_data)
19 display_stack(stack)
20
21 popped_items = pop(stack)
22 print("Popped items:", popped_items)
23 display_stack(stack)
```

## Outcome:

Thus, the above Python program has been executed and the output is verified successfully

```
Push operation completed.
Elements of the stack:
['KattarSunni', 'Ligma', 'AdityaKumar', 'Duryodhan']
Popped items: ['KattarSunni', 'Ligma', 'AdityaKumar', 'Duryodhan']
Stack is empty.
[]
```

## Exercise 16:

### CREATING A PYTHON PROGRAM TO INTEGRATE MYSQL WITH PYTHON (CREATING DATABASE AND TABLE)

#### Objective:

To write a Python Program to integrate MYSQL with Python to create Database and Table to store the details of employees.

```
1 import mysql.connector
2
3 def execute_query(cursor, query):
4     cursor.execute(query)
5     print("Operation successful.")
6
7 def insert_employee(cursor, table_name, employee):
8     execute_query(cursor, f"INSERT INTO {table_name} (name, age, salary) VALUES (%s, %s, %s)", employee)
9
10 # Example usage:
11 host, user, password = "192.168.0.1:6969", "Chandrash", "nigowplease"
12 database_name, table_name = "employees", "employee_details"
13
14 connection = mysql.connector.connect(host=host, user=user, password=password)
15 cursor = connection.cursor()
16
17 execute_query(cursor, f"CREATE DATABASE IF NOT
```

#### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

## Exercise 17:

### CREATING A PYTHON PROGRAM TO INTEGRATE MYSQL WITH PYTHON (INSERTING RECORDS AND DISPLAYING RECORDS)

## Objective:

To write a Python Program to integrate MYSQL with Python by inserting records to Emp table and display the records

```
1 import mysql.connector
2
3 def execute_query(cursor, query, values=None):
4     cursor.execute(query, values)
5     print("Operation successful.")
6
7 def insert_employee(cursor, emp_data):
8     execute_query(cursor, "INSERT INTO Emp (Emp_ID, Emp_Name, Salary) VALUES (%s, %s, %s)", emp_data)
9
10 def display_records(cursor):
11     execute_query(cursor, "SELECT * FROM Emp")
12     records = cursor.fetchall()
13
14     print("Records in Emp table:" if records else "No records found in Emp table.")
15     [print(record) for record in records]
16
17 # Example usage:
18 host, user, password, database = "your_mysql_host", "your_username", "your_password", "your_database"
19 table_name = "Emp"
20
21 connection = mysql.connector.connect(host=host, user=user, password=password, database=database)
22 cursor = connection.cursor()
23
24 execute_query(cursor, f"CREATE TABLE IF NOT EXISTS {table_name} (Emp_ID INT PRIMARY KEY, Emp_Name VARCHAR(255), Salary FLOAT)")
25
26 insert_employee(cursor, (101, "BediChai", 50000.0))
27 insert_employee(cursor, (102, "KattarSunni786", 60000.0))
28
29 display_records(cursor)
30
31 connection.commit()
32 connection.close()
```

## Outcome:

Thus, the above Python program has been executed and the output is verified successfully

## Exercise 18:

# CREATING A PYTHON PROGRAM TO INTEGRATE MYSQL WITH PYTHON (SEARCHING AND DISPLAYING RECORDS)

## Objective:



To write a Python Program to integrate MYSQL with Python to search an Employee using EMPID and display the record if present in already existing table EMP, if not display the appropriate message.

```
1 import mysql.connector
2
3 def execute_query(cursor, query, values=None):
4     cursor.execute(query, values)
5     print("Operation successful.")
6
7 def search_employee(cursor, emp_id):
8     query = "SELECT * FROM EMP WHERE EMPID = %s"
9     cursor.execute(query, (emp_id,))
10    return cursor.fetchone()
11
12 # Example usage:
13 host, user, password, database = "your_mysql_host", "your_username", "your_password", "your_database"
14 table_name = "EMP"
15
16 connection = mysql.connector.connect(host=host, user=user, password=password, database=database)
17 cursor = connection.cursor()
18
19 execute_query(cursor, f"CREATE TABLE IF NOT EXISTS {table_name} (EMPID INT PRIMARY KEY, EMPNAME VARCHAR(255), SALARY FLOAT)")
20
21 execute_query(cursor, f"INSERT INTO {table_name} (EMPID, EMPNAME, SALARY) VALUES (101, 'John Doe', 50000.0)")
22 execute_query(cursor, f"INSERT INTO {table_name} (EMPID, EMPNAME, SALARY) VALUES (102, 'Jane Smith', 60000.0)")
23
24 search_emp_id = int(input("Enter EMPID to search: "))
25 result = search_employee(cursor, search_emp_id)
26
27 print(f"Record found: {result}") if result else print(f"No record found for EMPID {search_emp_id}.")
28
29 connection.close()
```

## Outcome:

Thus, the above Python program has been executed and the output is verified successfully

## Exercise 19:

# CREATING A PYTHON PROGRAM TO INTEGRATE MYSQL WITH PYTHON (UPDATING RECORDS)

### Objective:

To write a Python Program to integrate MYSQL with Python to search an Employee using EMPID and update the Salary of an employee if present in already existing table EMP, if not display the appropriate message.

```
1 import mysql.connector
2
3 def execute_query(cursor, query, values=None):
4     cursor.execute(query, values)
5     print("Operation successful.")
6
7 def search_employee(cursor, emp_id):
8     cursor.execute("SELECT * FROM EMP WHERE EMPID = %s", (emp_id,))
9     return cursor.fetchone()
10
11 def update_salary(cursor, emp_id, new_salary):
12     execute_query(cursor, "UPDATE EMP SET SALARY = %s WHERE EMPID = %s", (new_salary, emp_id))
13
14 # Example usage:
15 host, user, password, database = "your_mysql_host", "your_username", "your_password", "your_database"
16 table_name = "EMP"
17
18 connection = mysql.connector.connect(host=host, user=user, password=password, database=database)
19 cursor = connection.cursor()
20
21 execute_query(cursor, f"CREATE TABLE IF NOT EXISTS {table_name} (EMPID INT PRIMARY KEY, EMPNAME VARCHAR(255), SALARY FLOAT)")
22
23 execute_query(cursor, f"INSERT INTO {table_name} (EMPID, EMPNAME, SALARY) VALUES (101, 'John Doe', 50000.0)")
24 execute_query(cursor, f"INSERT INTO {table_name} (EMPID, EMPNAME, SALARY) VALUES (102, 'Jane Smith', 60000.0)")
25
26 search_emp_id = int(input("Enter EMPID to search: "))
27 result = search_employee(cursor, search_emp_id)
28
29 if result:
30     new_salary = float(input("Enter new salary: "))
31     update_salary(cursor, search_emp_id, new_salary)
32     print(f"Salary updated for EMPID {search_emp_id}.")
33 else:
34     print(f"No record found for EMPID {search_emp_id}.")
35
36 connection.commit()
37 connection.close()
```

### Outcome:

Thus, the above Python program has been executed and the output is verified successfully

## SQL COMMANDS EXERCISE - 1

**Ex.No: 20**

**DATE:**

**AIM:**

To write Queries for the following Questions based on the given table:

Rollno	Name	Gender	Age	Dept	DOA	Fees
1	Arun	M	24	COMPUTER	1997-01-10	120
2	Ankit	M	21	HISTORY	1998-03-24	200
3	Anu	F	20	HINDI	1996-12-12	300
4	Bala	M	19	NULL	1999-07-01	400
5	Charan	M	18	HINDI	1997-09-05	250
6	Deepa	F	19	HISTORY	1997-06-27	300
7	Dinesh	M	22	COMPUTER	1997-02-25	210
8	Usha	F	23	NULL	1997-07-31	200

(a) Write a Query to Create a new database in the name of "**STUDENTS**".

**CREATE DATABASE STUDENTS;**

(b) Write a Query to Open the database "**STUDENTS**".

**USE STUDENTS;**

(c) Write a Query to create the above table called: "**STU**"

**CREATE TABLE STU(ROLLNO INT PRIMARY KEY,NAME VARCHAR(10),  
GENDER VARCHAR(3), AGE INT,DEPT VARCHAR(15),  
DOA DATE,FEES INT);**

(d) Write a Query to list all the existing database names.

**SHOW DATABASES;**

```
+-----+
| Database |
+-----+
| employees |
| hospital  |
| hospitals |
| information_schema |
| ip_practicals |
| mysql     |
| performance_schema |
| school    |
| students  |
| sys       |
+-----+
```

(e) Write a Query to List all the tables that exists in the current database.

**SHOW TABLES;**

**Output:**

```
+-----+
| Tables_in_students |
+-----+
| stu                 |
+-----+
```

## SQL COMMANDS EXERCISE - 2

**Ex.No: 21**

**DATE:**

**AIM:**

To write Queries for the following Questions based on the given table:

Rollno	Name	Gender	Age	Dept	DOA	Fees
1	Arun	M	24	COMPUTER	1997-01-10	120
2	Ankit	M	21	HISTORY	1998-03-24	200
3	Anu	F	20	HINDI	1996-12-12	300
4	Bala	M	19	NULL	1999-07-01	400
5	Charan	M	18	HINDI	1997-09-05	250
6	Deepa	F	19	HISTORY	1997-06-27	300
7	Dinesh	M	22	COMPUTER	1997-02-25	210
8	Usha	F	23	NULL	1997-07-31	200

(a) Write a Query to insert all the rows of above table into Info table.

**INSERT INTO STU VALUES (1,'Arun','M', 24,'COMPUTER','1997-01-10', 120);**

**INSERT INTO STU VALUES (2,'Ankit','M', 21,'HISTORY','1998-03-24', 200);**

**INSERT INTO STU VALUES (3,'Anu','F', 20,'HINDI','1996-12-12', 300);**

**INSERT INTO STU VALUES (4,'Bala','M', 19, NULL,'1999-07-01', 400);**

**INSERT INTO STU VALUES (5,'Charan','M', 18,'HINDI','1997-06-27', 250);**

**INSERT INTO STU VALUES (6,'Deepa','F', 19,'HISTORY','1997-06-27', 300);**

**INSERT INTO STU VALUES (7,'Dinesh','M', 22,'COMPUTER','1997-02-25', 210);**

**INSERT INTO STU VALUES (8,'Usha','F', 23, NULL,'1997-07-31', 200);**

(b) Write a Query to display all the details of the Employees from the above table 'STU'.

**SELECT \* FROM STU;**

**Output:**

Rollno	Name	Gender	Age	Dept	DOA	Fees
1	Arun	M	24	COMPUTER	1997-01-10	120
2	Ankit	M	21	HISTORY	1998-03-24	200
3	Anu	F	20	HINDI	1996-12-12	300
4	Bala	M	19	NULL	1999-07-01	400
5	Charan	M	18	HINDI	1997-06-27	250
6	Deepa	F	19	HISTORY	1997-06-27	300
7	Dinesh	M	22	COMPUTER	1997-02-25	210
8	Usha	F	23	NULL	1997-07-31	200

(c) Write a query to Rollno, Name and Department of the students from **STU** table.

**SELECT ROLLNO,NAME,DEPT FROM STU;**

**Output:**

ROLLNO	NAME	DEPT
1	Arun	COMPUTER
2	Ankit	HISTORY
3	Anu	HINDI
4	Bala	NULL
5	Charan	HINDI
6	Deepa	HISTORY
7	Dinesh	COMPUTER
8	Usha	NULL

(d) Write a Query to select distinct Department from **STU** table.

**SELECT DISTICT(DEPT) FROM STU;**

**Output:**

DEPT
COMPUTER
HISTORY
HINDI
NULL

(e) To show all information about students of History department.

**SELECT \* FROM STU WHERE DEPT='HISTORY';**

**Output:**

Rollno	Name	Gender	Age	Dept	DOA	Fees
2	Ankit	M	21	HISTORY	1998-03-24	200
6	Deepa	F	19	HISTORY	1997-06-27	300

\*\*\*\*\*

**DATE:** \_\_\_\_\_**AIM:**

To write Queries for the following Questions based on the given table:

Rollno	Name	Gender	Age	Dept	DOA	Fees
1	Arun	M	24	COMPUTER	1997-01-10	120
2	Ankit	M	21	HISTORY	1998-03-24	200
3	Anu	F	20	HINDI	1996-12-12	300
4	Bala	M	19	NULL	1999-07-01	400
5	Charan	M	18	HINDI	1997-09-05	250
6	Deepa	F	19	HISTORY	1997-06-27	300
7	Dinesh	M	22	COMPUTER	1997-02-25	210
8	Usha	F	23	NULL	1997-07-31	200

**(a)** Write a Query to list name of female students in Hindi Department.**SELECT NAME FROM STU WHERE DEPT='HINDI' AND GENDER='F';****Output:**

```

+-----+
|  NAME  |
+-----+
|  Anu   |
+-----+

```

**(b)** Write a Query to list name of the students whose ages are between 18 to 20.**SELECT NAME FROM STU WHERE AGE BETWEEN 18 AND 20;****Output:**

```

+-----+
|  NAME  |
+-----+
|  Anu   |
|  Bala  |
|  Charan|
|  Deepa |
+-----+

```

(c) Write a Query to display the name of the students whose name is starting with 'A'.

**SELECT NAME FROM STU WHERE NAME LIKE 'A%';**

**Output:**

NAME
Arun
Ankit
Anu

(d) Write a query to list the names of those students whose name have second alphabet 'n' in their names.

**SELECT NAME FROM STU WHERE NAME LIKE '\_N%';**

**Output:**

NAME
Ankit
Anu

\*\*\*\*\*

**DATE:****AIM:**

To write Queries for the following Questions based on the given table:

Rollno	Name	Gender	Age	Dept	DOA	Fees
1	Arun	M	24	COMPUTER	1997-01-10	120
2	Ankit	M	21	HISTORY	1998-03-24	200
3	Anu	F	20	HINDI	1996-12-12	300
4	Bala	M	19	NULL	1999-07-01	400
5	Charan	M	18	HINDI	1997-09-05	250
6	Deepa	F	19	HISTORY	1997-06-27	300
7	Dinesh	M	22	COMPUTER	1997-02-25	210
8	Usha	F	23	NULL	1997-07-31	200

**(a)** Write a Query to delete the details of Roll number is 8.**DELETE FROM STU WHERE ROLLNO=8;****Output (After Deletion):**

Rollno	Name	Gender	Age	Dept	DOA	Fees
1	Arun	M	24	COMPUTER	1997-01-10	120
2	Ankit	M	21	HISTORY	1998-03-24	200
3	Anu	F	20	HINDI	1996-12-12	300
4	Bala	M	19	NULL	1999-07-01	400
5	Charan	M	18	HINDI	1997-06-27	250
6	Deepa	F	19	HISTORY	1997-06-27	300
7	Dinesh	M	22	COMPUTER	1997-02-25	210

**(b)** Write a Query to change the fess of Student to 170 whose Roll number is 1, if the existing fess is less than 130.**UPDATE STU SET FEES=170 WHERE ROLLNO=1 AND FEES<130;****Output(After Update):**

Rollno	Name	Gender	Age	Dept	DOA	Fees
1	Arun	M	24	COMPUTER	1997-01-10	170



(c) Write a Query to add a new column **Area** of type varchar in table STU.

**ALTER TABLE STU ADD AREA VARCHAR(20);**

**Output:**

```
Query OK, 0 rows affected
Records: 0 Duplicates: 0
```

```
mysql> SELECT * FROM STU;
```

ROLLNO	NAME	GENDER	AGE	DEPT	DOA	FEES	AREA
1	Arun	M	24	COMPUTER	1997-01-10	120	NULL
2	Ankit	M	21	HISTORY	1998-03-24	200	NULL
3	Anu	F	20	HINDI	1996-12-12	300	NULL
4	Bala	M	19	NULL	1999-07-01	400	NULL
5	Charan	M	18	HINDI	1997-09-05	250	NULL
6	Deepa	F	19	HISTORY	1997-06-27	300	NULL
7	Dinesh	M	22	COMPUTER	1997-02-25	210	NULL
8	Usha	F	23	NULL	1997-07-31	200	NULL

(d) Write a Query to Display Name of all students whose Area Contains NULL.

**SELECT NAME FROM STU WHERE AREA IS NULL;**

**Output:**

NAME
Arun
Ankit
Anu
Bala
Charan
Deepa
Dinesh
Usha

(e) Write a Query to delete Area Column from the table STU.

**ALTER TABLE STU DROP AREA;**

**Output:**

```
Query OK, 0 rows affected
Records: 0 Duplicates: 0
```

(f) Write a Query to delete table from Database.

**DROP TABLE STU;**

**Output:**

```
Query OK, 0 rows affected
```

\*\*\*\*\*

**Ex.No: 24**

**DATE:**

**SQL COMMANDS EXERCISE - 5**

**AIM:**

To write Queries for the following Questions based on the given table:

**TABLE: UNIFORM**

Ucode	Uname	Ucolor	StockDate
1	Shirt	White	2021-03-31
2	Pant	Black	2020-01-01
3	Skirt	Grey	2021-02-18
4	Tie	Blue	2019-01-01
5	Socks	Blue	2019-03-19
6	Belt	Black	2017-12-09

**TABLE: COST**

Ucode	Size	Price	Company
1	M	500	Raymond
1	L	580	Mattex
2	XL	620	Mattex
2	M	810	Yasin
2	L	940	Raymond
3	M	770	Yasin
3	L	830	Galin
4	S	150	Mattex

**(a)** To Display the average price of all the Uniform of Raymond Company from table COST.

**SELECT AVG(PRICE) FROM COST WHERE COMPANY='RAYMOND';**

**Output:**

AVG(PRICE)
720.0000

**(b)** To display details of all the Uniform in the Uniform table in descending order of Stock date.

**SELECT \* FROM UNIFORM ORDER BY STOCKDATE DESC;**

**Output:**

Ucode	Uname	Ucolor	StockDate
1	Shirt	White	2021-03-31
3	Skirt	Grey	2021-02-18
2	Pant	Black	2020-01-01
5	Socks	Blue	2019-03-19
4	Tie	Blue	2019-01-01
6	Belt	Black	2017-12-09

(c) To Display max price and min price of each company.

**SELECT COMPANY,MAX(PRICE),MIN(PRICE) FROM COST GROUP BY COMPANY;**

**Output:**

COMPANY	MAX(PRICE)	MIN(PRICE)
RAYMOND	940	500
MATTEX	620	150
YASIN	810	770
GALIN	830	830

(d) To display the company where the number of uniforms size is more than 2.

**SELECT COMPANY, COUNT(\*) FROM COST GROUP BY COMPANY HAVING COUNT(\*)>2;**

**Output:**

COMPANY	COUNT(*)
MATTEX	3

(e) To display the Ucode, Uname, Ucolor, Size and Company of tables uniform and cost.

**SELECT U.UCODE,UNAME,UCOLOR,SIZE,COMPANY FROM UNIFORM U,COST C WHERE U.UCODE=C.UCODE;**

**Output:**

UCODE	UNAME	UCOLOR	SIZE	COMPANY
1	Shirt	White	M	RAYMOND
1	Shirt	White	L	MATTEX
2	Pant	Black	XL	MATTEX
2	Pant	Black	M	YASIN
2	Pant	Black	L	RAYMOND
3	Skirt	Grey	M	YASIN
3	Skirt	Grey	L	GALIN
4	Tie	Blue	S	MATTEX

\*\*\*\*\*