Subject Code: LPCIT-104

Subject Name: Database Management System Laboratory

Programme: B.Tech. (IT)	**L: 0 T: 0 P: 2**
Semester: 4	**Teaching Hours: 26**
Theory/Practical: Practical	**Credits: 1**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE): 1.5 hr**
Total Marks: 50	**Elective Status: Compulsory**

Name of Practicals:

1. Creating and Managing Tables: Create table statement; referencing another user's tables; the DEFAULT option; data types; alter table statement; adding a column; modifying a column; dropping a column; dropping a table; truncating a table.

2. Writing Basic SQL SELECT Statements : Basic SELECT Statement; selecting - all columns, specific columns; using arithmetic operators; operator precedence; using parenthesis; defining a NULL Value; using column aliases; concatenation operator; eliminating duplicate rows; displaying table structure.

3. Restricting and Sorting Data: Limiting rows using a selection; character strings and dates; comparison conditions; using the BETWEEN condition; IN condition; LIKE condition; NULL conditions; logical conditions- AND, OR and NOT operators; rules of precedence; ORDER BY clause; sorting – ascending, descending order.

4. Manipulating Data: Data manipulation language; adding a new row to a table; inserting- new rows, rows with NULL values, specific date values; updating rows in a table; updating two columns; updating rows based on another table; removing a row from a table deleting rows from a table; deleting rows based on another table.

5. Single Row Functions: Character functions - case manipulation and character manipulation functions; number functions, date functions; using arithmetic operators with dates; date functions, conversion functions.

6. Displaying Data from Multiple Tables: Cartesian products; different types of joins specific to the software package; SQL compliant joins

7. Aggregating Data Using Group Functions: Group functions for various statistical metrics; creating groups of data by GROUP BY clause; grouping by more than one column; excluding group results- HAVING Clause.

8. Subqueries: Single-row subqueries; multiple-row subqueries; using group function in a subquery; HAVING clause with subqueries; usage of operators in multiple-row subqueries

9. Creating Views: Simple views and complex views; creating a view; retrieving data from view; querying a view; modifying a view; removing a view; inline views.

10. Overview of MongoDB: A NoSQL database: Create and drop-database, collection; data types; insert document; query document; logical operators; update document; delete document; projection; limit records; sort documents.

Experiment No. 1:

Creating and Managing Tables

1: Creating a Table

```
CREATE TABLE Students (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT,
    Course VARCHAR(50)
);
```

Output:

```
Table STUDENTS created.
```

2: Creating a Table with DEFAULT Option

```
CREATE TABLE Library (
    BookID INT PRIMARY KEY,
    Title VARCHAR(100),
    Author VARCHAR(50),
    AvailabilityStatus VARCHAR(20) DEFAULT 'Available'
);
```

Output:

```
Table LIBRARY created.
```

3: Creating a Table by Referencing Another User’s Table

```
CREATE TABLE BorrowedBooks (
    BorrowID INT PRIMARY KEY,
    StudentID INT,
    BookID INT,
    FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    FOREIGN KEY (BookID) REFERENCES Library(BookID)
);
```

Output:

```
Table BORROWEDBOOKS created.
```

4: Altering a Table - Adding a Column

```
ALTER TABLE Students ADD Email VARCHAR(100);
```

Output:

```
Table STUDENTS altered.
```

5: Altering a Table - Modifying a Column

```
ALTER TABLE Students MODIFY Age INT NOT NULL;
```

Output:

```
Table STUDENTS altered.
```

6: Altering a Table - Dropping a Column

```
ALTER TABLE Students DROP COLUMN Email;
```

Output:

```
Table STUDENTS altered.
```

7: Dropping a Table

```
DROP TABLE BorrowedBooks;
```

Output:

```
Table BORROWEDBOOKS dropped.
```

8: Truncating a Table (Deleting All Data Without Dropping the Structure)

```
TRUNCATE TABLE Library;
```

Output:

```
Table LIBRARY truncated.
```

Experiment No. 2:

Writing Basic SQL SELECT Statements

Table: Students

StudentID	Name	Age	Course
1	Alice	20	CS
2	Bob	22	IT
3	Charlie	21	CS
4	David	23	IT
5	Eva	20	CS

1: Selecting All Columns from a Table

```
SELECT * FROM Students;
```

Output:

StudentID	Name	Age	Course
1	Alice	20	CS
2	Bob	22	IT
3	Charlie	21	CS
4	David	23	IT
5	Eva	20	CS

2: Selecting Specific Columns from a Table

```
SELECT Name, Course FROM Students;
```

Output:

Name	Course
Alice	CS
Bob	IT
Charlie	CS
David	IT
Eva	CS

3: Using Arithmetic Operators in SELECT Statement

s

```
SELECT StudentID, Name, Age + 5 AS AgeAfter5Years FROM Students;
```

Output:

StudentID	Name	AgeAfter5Years
1	Alice	25
2	Bob	27
3	Charlie	26
4	David	28
5	Eva	25

4: Operator Precedence Example

```
SELECT 10 + 5 * 2 AS Result;
```

Output:

Result
20

5: Using Parentheses to Change Precedence

```
SELECT (10 + 5) * 2 AS Result;
```

Output:

Result
30

6: Defining a NULL Value in SELECT Statement

```
SELECT Name, Age, Course FROM Students WHERE Course IS NULL;
```

Output:

```
Empty set (0.00 sec)
```

(If NULL values exist, they will be displayed.)

7: Using Column Aliases

```
SELECT Name AS StudentName, Course AS Program FROM Students;
```

Output:

StudentName	Program
Alice	CS
Bob	IT
Charlie	CS
David	IT
Eva	CS

8: Using Concatenation Operator *(MySQL uses CONCAT function)*

```
SELECT CONCAT(Name, ' is in ', Course) AS StudentInfo FROM Students;
```

Output:

StudentInfo
Alice is in CS
Bob is in IT
Charlie is in CS
David is in IT
Eva is in CS

9: Eliminating Duplicate Rows

```
SELECT DISTINCT Course FROM Students;
```

Output:

Course
CS
IT

10: Displaying Table Structure

```
DESC Students;
```

Output:

Field	Type	Null	Key	Default	Extra
StudentID	INT	NO	PRI	NULL	
Name	VARCHAR(50)	YES		NULL	
Age	INT	YES		NULL	
Course	VARCHAR(50)	YES		NULL	

Experiment No. 3:

Restricting and Sorting Data

Table: Students

StudentID	Name	Age	Course	JoiningDate
1	Alice	20	CS	2023-01-15
2	Bob	22	IT	2022-12-10
3	Charlie	21	CS	2023-02-05
4	David	23	IT	2021-11-20
5	Eva	20	CS	2023-05-12

1: Limiting Rows Using a Selection (WHERE Clause)

```
SELECT * FROM Students WHERE Age > 21;
```

Output:

StudentID	Name	Age	Course	JoiningDate
2	Bob	22	IT	2022-12-10
4	David	23	IT	2021-11-20

2: Selecting Character Strings and Dates

```
SELECT Name, JoiningDate FROM Students WHERE Course = 'CS';
```

Output:

Name	JoiningDate
Alice	2023-01-15
Charlie	2023-02-05
Eva	2023-05-12

3: Using the BETWEEN Condition

```
SELECT * FROM Students WHERE Age BETWEEN 20 AND 22;
```

Output:

StudentID	Name	Age	Course	JoiningDate
1	Alice	20	CS	2023-01-15
2	Bob	22	IT	2022-12-10
3	Charlie	21	CS	2023-02-05
5	Eva	20	CS	2023-05-12

4: Using the IN Condition

```
SELECT * FROM Students WHERE Course IN ('CS', 'IT');
```

Output: *(Same as full table because all students are in CS or IT)*

StudentID	Name	Age	Course	JoiningDate
1	Alice	20	CS	2023-01-15
2	Bob	22	IT	2022-12-10
3	Charlie	21	CS	2023-02-05
4	David	23	IT	2021-11-20
5	Eva	20	CS	2023-05-12

5: Using the LIKE Condition (Pattern Matching)

```
SELECT * FROM Students WHERE Name LIKE 'A%';
```

Output: *(Names starting with 'A')*

StudentID	Name	Age	Course	JoiningDate
1	Alice	20	CS	2023-01-15

6: Using the NULL Condition

```
SELECT * FROM Students WHERE Course IS NULL;
```

Output:

```
Empty set (0.00 sec)
```

(No NULL values in Course column.)

7: Using Logical Conditions (AND, OR, NOT Operators)

```
SELECT * FROM Students WHERE Age > 20 AND Course = 'CS';
```

Output:

StudentID	Name	Age	Course	JoiningDate
3	Charlie	21	CS	2023-02-05

```
SELECT * FROM Students WHERE Age > 22 OR Course = 'IT';
```

Output:

StudentID	Name	Age	Course	JoiningDate
2	Bob	22	IT	2022-12-10
4	David	23	IT	2021-11-20

```
SELECT * FROM Students WHERE NOT Course = 'IT';
```

Output:

StudentID	Name	Age	Course	JoiningDate
1	Alice	20	CS	2023-01-15
3	Charlie	21	CS	2023-02-05
5	Eva	20	CS	2023-05-12

8: Rules of Precedence in Logical Operators

```
SELECT * FROM Students WHERE Age > 20 OR Course = 'IT' AND Name LIKE 'B%';
```

Output:

StudentID	Name	Age	Course	JoiningDate
2	Bob	22	IT	2022-12-10
3	Charlie	21	CS	2023-02-05
4	David	23	IT	2021-11-20

(Since `AND` has higher precedence than `OR`, `Course = 'IT' AND Name LIKE 'B%'` is evaluated first, then OR condition is applied.)

9: Sorting Data Using ORDER BY Clause

Ascending Order (Default):

```
SELECT * FROM Students ORDER BY Age;
```

Output:

StudentID	Name	Age	Course	JoiningDate
1	Alice	20	CS	2023-01-15
5	Eva	20	CS	2023-05-12
3	Charlie	21	CS	2023-02-05
2	Bob	22	IT	2022-12-10
4	David	23	IT	2021-11-20

Descending Order:

```
SELECT * FROM Students ORDER BY Age DESC;
```

Output:

StudentID	Name	Age	Course	JoiningDate
4	David	23	IT	2021-11-20
2	Bob	22	IT	2022-12-10
3	Charlie	21	CS	2023-02-05
1	Alice	20	CS	2023-01-15
5	Eva	20	CS	2023-05-12

Experiment No. 4:

Manipulating Data

Table: Employees

EmpID	Name	Age	Department	Salary	JoiningDate
101	Alice	30	HR	50000	2022-01-15
102	Bob	28	IT	60000	2021-12-10
103	Charlie	35	Finance	70000	2020-05-20
104	David	29	IT	65000	2022-03-10
105	Eva	32	HR	52000	2019-07-25

1: Adding a New Row to a Table

```
INSERT INTO Employees (EmpID, Name, Age, Department, Salary, JoiningDate)
VALUES (106, 'Frank', 27, 'Marketing', 55000, '2023-02-18');
```

Output:

```
1 row inserted.
```

Updated Table:

EmpID	Name	Age	Department	Salary	JoiningDate
101	Alice	30	HR	50000	2022-01-15
102	Bob	28	IT	60000	2021-12-10
103	Charlie	35	Finance	70000	2020-05-20
104	David	29	IT	65000	2022-03-10
105	Eva	32	HR	52000	2019-07-25
106	Frank	27	Marketing	55000	2023-02-18

2: Inserting Rows with NULL Values

```
INSERT INTO Employees (EmpID, Name, Age, Department, Salary, JoiningDate)
VALUES (107, 'Grace', NULL, 'Finance', NULL, '2023-06-15');
```

Output:

```
1 row inserted.
```

Updated Table:

EmpID	Name	Age	Department	Salary	JoiningDate
107	Grace	NULL	Finance	NULL	2023-06-15

3: Inserting Specific Date Values

```
INSERT INTO Employees (EmpID, Name, Age, Department, Salary, JoiningDate)
VALUES (108, 'Henry', 31, 'Sales', 58000, '2022-10-05');
```

Output:

```
1 row inserted.
```

4: Updating Rows in a Table

```
UPDATE Employees SET Salary = 62000 WHERE EmpID = 102;
```

Output:

```
1 row updated.
```

Updated Table (for EmpID 102):

EmpID	Name	Age	Department	Salary	JoiningDate
102	Bob	28	IT	62000	2021-12-10

5: Updating Two Columns

```
UPDATE Employees SET Age = 36, Salary = 75000 WHERE EmpID = 103;
```

Output:

```
1 row updated.
```

Updated Table (for EmpID 103):

EmpID	Name	Age	Department	Salary	JoiningDate
103	Charlie	36	Finance	75000	2020-05-20

6: Updating Rows Based on Another Table

Assume a second table called `SalaryUpdates`:

EmpID	NewSalary
101	52000
104	68000

```
UPDATE Employees
SET Salary = (SELECT NewSalary FROM SalaryUpdates WHERE Employees.EmpID =
SalaryUpdates.EmpID)
WHERE EmpID IN (SELECT EmpID FROM SalaryUpdates);
```

Output:

```
2 rows updated.
```

Updated Table (for EmpIDs 101 and 104):

EmpID	Name	Age	Department	Salary	JoiningDate
101	Alice	30	HR	52000	2022-01-15
104	David	29	IT	68000	2022-03-10

7: Removing a Row from a Table

```
DELETE FROM Employees WHERE EmpID = 106;
```

Output:

```
1 row deleted.
```

Updated Table: *(Row with EmpID 106 removed.)*

8: Deleting Rows from a Table

```
DELETE FROM Employees WHERE Age > 35;
```

Output:

```
1 row deleted.
```

Updated Table (for EmpID 103, assuming Charlie is above 35):

EmpID	Name	Age	Department	Salary	JoiningDate
101	Alice	30	HR	52000	2022-01-15
102	Bob	28	IT	62000	2021-12-10
104	David	29	IT	68000	2022-03-10
105	Eva	32	HR	52000	2019-07-25
107	Grace	NULL	Finance	NULL	2023-06-15
108	Henry	31	Sales	58000	2022-10-05

9: Deleting Rows Based on Another Table

Assume another table `TerminatedEmployees`:

EmpID
105
108

```
DELETE FROM Employees WHERE EmpID IN (SELECT EmpID FROM TerminatedEmployees);
```

Output:

```
2 rows deleted.
```

Updated Table: *(EmpID 105 and 108 removed.)*

Experiment No. 5:

Single Row Functions

Table: Employees

EmpID	Name	Department	Salary	JoiningDate
101	Alice	HR	50000	2022-01-15
102	Bob	IT	60000	2021-12-10
103	Charlie	Finance	70000	2020-05-20
104	David	IT	65000	2022-03-10
105	Eva	HR	52000	2019-07-25

1: Character Functions - Case Manipulation

```
SELECT Name, UPPER(Name) AS UpperCase, LOWER(Name) AS LowerCase, INITCAP(Name)
AS InitCap FROM Employees;
```

Output:

Name	UpperCase	LowerCase	InitCap
Alice	ALICE	alice	Alice
Bob	BOB	bob	Bob
Charlie	CHARLIE	charlie	Charlie
David	DAVID	david	David
Eva	EVA	eva	Eva

2: Character Functions - String Manipulation

```
SELECT Name, LENGTH(Name) AS Length, SUBSTR(Name, 1, 3) AS FirstThree,
INSTR(Name, 'a') AS PositionA, REPLACE(Name, 'a', '*') AS ReplacedA FROM
Employees;
```

Output:

Name	Length	FirstThree	PositionA	ReplacedA
Alice	5	Ali	2	Alice
Bob	3	Bob	0	Bob
Charlie	7	Cha	3	Ch*rlie
David	5	Dav	2	D*vid
Eva	3	Eva	2	Ev*

3: Number Functions

```
SELECT Salary, ROUND(Salary, -3) AS Rounded, TRUNC(Salary, -3) AS Truncated,
MOD(Salary, 5000) AS Modulus FROM Employees;
```

Output:

Salary	Rounded	Truncated	Modulus
50000	50000	50000	0
60000	60000	60000	0
70000	70000	70000	0
65000	65000	65000	0
52000	52000	52000	0

4: Date Functions

```
SELECT JoiningDate, SYSDATE AS Today, MONTHS_BETWEEN(SYSDATE, JoiningDate) AS
MonthsWorked, ADD_MONTHS(JoiningDate, 6) AS After6Months, NEXT_DAY(JoiningDate,
'Monday') AS NextMonday FROM Employees;
```

(Assume today's date is **2025-03-27**.)

Output:

JoiningDate	Today	MonthsWorked	After6Months	NextMonday
2022-01-15	2025-03-27	38.4	2022-07-15	2022-01-17
2021-12-10	2025-03-27	39.6	2022-06-10	2021-12-13
2020-05-20	2025-03-27	58.2	2020-11-20	2020-05-25
2022-03-10	2025-03-27	36.5	2022-09-10	2022-03-14
2019-07-25	2025-03-27	68.1	2020-01-25	2019-07-29

5: Arithmetic Operators with Dates

```
SELECT JoiningDate, JoiningDate + 30 AS After30Days, JoiningDate - 15 AS
Before15Days FROM Employees;
```

Output:

JoiningDate	After30Days	Before15Days
2022-01-15	2022-02-14	2021-12-31
2021-12-10	2022-01-09	2021-11-25
2020-05-20	2020-06-19	2020-05-05
2022-03-10	2022-04-09	2022-02-23
2019-07-25	2019-08-24	2019-07-10

6: Conversion Functions

```
SELECT TO_CHAR(SYSDATE, 'DD-MON-YYYY') AS TodayFormatted, TO_NUMBER('12345') +
10 AS NumberConversion, TO_DATE('15-APR-2023', 'DD-MON-YYYY') AS ConvertedDate
FROM DUAL;
```

Output:

TodayFormatted	NumberConversion	ConvertedDate
27-MAR-2025	12355	2023-04-15

Experiment No. 6:

Displaying Data from Multiple Tables

Tables Used

Employees Table

EmpID	Name	DeptID	Salary
101	Alice	1	50000
102	Bob	2	60000
103	Charlie	1	70000
104	David	3	65000
105	Eva	2	52000

Departments Table

DeptID	DeptName	Location
1	HR	New York
2	IT	San Francisco
3	Finance	Chicago

1: Cartesian Product

```
SELECT Employees.Name, Employees.Salary, Departments.DeptName,
Departments.Location
FROM Employees, Departments;
```

Output:

Name	Salary	DeptName	Location
Alice	50000	HR	New York
Alice	50000	IT	San Francisco
Alice	50000	Finance	Chicago
Bob	60000	HR	New York
Bob	60000	IT	San Francisco
Bob	60000	Finance	Chicago
Charlie	70000	HR	New York
Charlie	70000	IT	San Francisco
Charlie	70000	Finance	Chicago
...	...	...	...

(Total: 5 × 3 = 15 rows, showing only a few for brevity.)

2: INNER JOIN (SQL Compliant Join)

```
SELECT Employees.Name, Employees.Salary, Departments.DeptName,
Departments.Location
FROM Employees
INNER JOIN Departments ON Employees.DeptID = Departments.DeptID;
```

Output:

Name	Salary	DeptName	Location
Alice	50000	HR	New York
Charlie	70000	HR	New York
Bob	60000	IT	San Francisco
Eva	52000	IT	San Francisco
David	65000	Finance	Chicago

3: LEFT JOIN

```
SELECT Employees.Name, Employees.Salary, Departments.DeptName,
Departments.Location
FROM Employees
LEFT JOIN Departments ON Employees.DeptID = Departments.DeptID;
```

Output:

Name	Salary	DeptName	Location
Alice	50000	HR	New York
Charlie	70000	HR	New York
Bob	60000	IT	San Francisco
Eva	52000	IT	San Francisco
David	65000	Finance	Chicago

(Same output as INNER JOIN because all employees have matching departments.)

4: RIGHT JOIN

```
SELECT Employees.Name, Employees.Salary, Departments.DeptName,
Departments.Location
FROM Employees
RIGHT JOIN Departments ON Employees.DeptID = Departments.DeptID;
```

Output:

Name	Salary	DeptName	Location
Alice	50000	HR	New York
Charlie	70000	HR	New York
Bob	60000	IT	San Francisco
Eva	52000	IT	San Francisco
David	65000	Finance	Chicago

(Same output because all departments are assigned to employees.)

5: FULL OUTER JOIN

```
SELECT Employees.Name, Employees.Salary, Departments.DeptName,
Departments.Location
FROM Employees
FULL OUTER JOIN Departments ON Employees.DeptID = Departments.DeptID;
```

Output:

Name	Salary	DeptName	Location
Alice	50000	HR	New York
Charlie	70000	HR	New York
Bob	60000	IT	San Francisco
Eva	52000	IT	San Francisco
David	65000	Finance	Chicago

(Since there are no unmatched departments or employees, the result is identical to INNER JOIN.)

6: SELF JOIN (Example for Employees Table)

```
SELECT E1.Name AS Employee, E1.Salary, E2.Name AS Manager, E2.Salary AS
ManagerSalary
FROM Employees E1
INNER JOIN Employees E2 ON E1.Salary < E2.Salary;
```

Output:

Employee	Salary	Manager	ManagerSalary
Alice	50000	Bob	60000
Alice	50000	Charlie	70000
Alice	50000	David	65000
Alice	50000	Eva	52000
Bob	60000	Charlie	70000
Bob	60000	David	65000
Bob	60000	Eva	52000
...	...	...	...

Experiment No. 7:

Aggregating Data Using Group Functions

Tables Used

Sales Table

SaleID	Product	Category	Amount	SalesDate
1	Laptop	Electronics	80000	2024-03-01
2	Phone	Electronics	50000	2024-03-05
3	TV	Electronics	60000	2024-03-10
4	Shirt	Clothing	2000	2024-03-12
5	Jeans	Clothing	3000	2024-03-15
6	Fan	Home	4000	2024-03-20
7	Fridge	Home	25000	2024-03-25
8	Shoes	Clothing	4000	2024-03-28

1: Using Aggregate Functions

```
SELECT COUNT(*) AS Total_Sales,
       SUM(Amount) AS Total_Amount,
       AVG(Amount) AS Average_Sale,
       MAX(Amount) AS Max_Sale,
       MIN(Amount) AS Min_Sale
FROM Sales;
```

Output:

Total_Sales	Total_Amount	Average_Sale	Max_Sale	Min_Sale
8	265000	33125.00	80000	2000

2: Grouping Data by Category (GROUP BY Clause)

```
SELECT Category,
       COUNT(*) AS Total_Sales,
       SUM(Amount) AS Total_Amount,
       AVG(Amount) AS Average_Sale
FROM Sales
GROUP BY Category;
```

Output:

Category	Total_Sales	Total_Amount	Average_Sale
Electronics	3	190000	63333.33
Clothing	3	9000	3000.00
Home	2	29000	14500.00

3: Grouping by Multiple Columns (Category and SalesDate)

```
SELECT Category, SalesDate,
       COUNT(*) AS Sales_Count,
       SUM(Amount) AS Total_Amount
FROM Sales
```

```
GROUP BY Category, SalesDate;
```

Output:

Category	SalesDate	Sales_Count	Total_Amount
Electronics	2024-03-01	1	80000
Electronics	2024-03-05	1	50000
Electronics	2024-03-10	1	60000
Clothing	2024-03-12	1	2000
Clothing	2024-03-15	1	3000
Clothing	2024-03-28	1	4000
Home	2024-03-20	1	4000
Home	2024-03-25	1	25000

4: Filtering Grouped Data Using HAVING Clause

```
SELECT Category,
       SUM(Amount) AS Total_Amount
FROM Sales
GROUP BY Category
HAVING SUM(Amount) > 10000;
```

Output:

Category	Total_Amount
Electronics	190000
Home	29000

(Only groups where `Total_Amount > 10000` are displayed.)

Experiment No. 8:

Subqueries

Tables Used

Employees Table

EmpID	Name	DepartmentID	Salary
101	Alice	1	50000
102	Bob	2	60000
103	Charlie	3	70000
104	David	2	65000
105	Eva	1	52000

Departments Table

DepartmentID	DepartmentName	Location
1	HR	New York
2	IT	San Francisco
3	Finance	Chicago

1: Single-Row Subquery

Find employees who earn more than the average salary.

```
SELECT Name, Salary
FROM Employees
WHERE Salary > (SELECT AVG(Salary) FROM Employees);
```

Output:

Name	Salary
Charlie	70000
David	65000

🔹 **Explanation:** The subquery calculates the average salary, and the main query retrieves employees earning above that.

2: Multiple-Row Subquery Using IN

Find all employees working in the IT department.

```
SELECT Name, Salary
FROM Employees
WHERE DepartmentID IN (SELECT DepartmentID FROM Departments WHERE DepartmentName
= 'IT');
```

Output:

Name	Salary
Bob	60000
David	65000

🔹 **Explanation:** The subquery retrieves the `DepartmentID` for "IT", and the main query fetches employees from that department.

3: Using GROUP FUNCTION in a Subquery

Find the employee with the highest salary.

```
SELECT Name, Salary
FROM Employees
WHERE Salary = (SELECT MAX(Salary) FROM Employees);
```

Output:

Name	Salary
Charlie	70000

🔹 **Explanation:** The subquery finds the maximum salary, and the main query retrieves the employee with that salary.

4: Using HAVING Clause with a Subquery

Find departments where the total salary of employees exceeds 100,000.

```
SELECT DepartmentID, SUM(Salary) AS TotalSalary
FROM Employees
GROUP BY DepartmentID
HAVING SUM(Salary) > (SELECT SUM(Salary)/2 FROM Employees);
```

Output:

DepartmentID	TotalSalary
2	125000

🔹 **Explanation:** The subquery calculates half of the total salary across departments, and the main query retrieves departments exceeding that.

5: Using Operators in Multiple-Row Subqueries

Find employees who earn more than the minimum salary in the IT department.

```
SELECT Name, Salary
FROM Employees
WHERE Salary > ANY (SELECT Salary FROM Employees WHERE DepartmentID = 2);
```

Output:

Name	Salary
Charlie	70000
David	65000

🔹 **Explanation:** The subquery retrieves all salaries from the IT department, and the main query finds employees earning more than the lowest IT salary.

Experiment No. 9:

Creating Views

Tables Used

Employees Table

EmpID	Name	DepartmentID	Salary
101	Alice	1	50000
102	Bob	2	60000
103	Charlie	3	70000
104	David	2	65000
105	Eva	1	52000

Departments Table

DepartmentID	DepartmentName	Location
1	HR	New York
2	IT	San Francisco
3	Finance	Chicago

1: Creating a Simple View

Create a view that displays only employee names and salaries.

```
CREATE VIEW Employee_View AS
SELECT Name, Salary
FROM Employees;
```

Output:
View **Employee_View** created successfully.

2: Retrieving Data from a View

```
SELECT * FROM Employee_View;
```

Output:

Name	Salary
Alice	50000
Bob	60000
Charlie	70000
David	65000
Eva	52000

3: Creating a Complex View (Using JOINs)

Create a view that displays employee names, salaries, and department names.

```
CREATE VIEW Employee_Details AS
SELECT E.Name, E.Salary, D.DepartmentName
```

```
FROM Employees E
JOIN Departments D ON E.DepartmentID = D.DepartmentID;
```

Output:
View **Employee_Details** created successfully.

4: Querying a View

```
SELECT * FROM Employee_Details;
```

Output:

Name	Salary	DepartmentName
Alice	50000	HR
Bob	60000	IT
Charlie	70000	Finance
David	65000	IT
Eva	52000	HR

5: Modifying a View

Modify the `Employee_View` to also include `DepartmentID`.

```
CREATE OR REPLACE VIEW Employee_View AS
SELECT Name, Salary, DepartmentID
FROM Employees;
```

Output:
View **Employee_View** modified successfully.

6: Removing a View

```
DROP VIEW Employee_View;
```

Output:
View **Employee_View** dropped successfully.

7: Creating an Inline View (Subquery in FROM Clause)

Find the highest-paid employee in each department using an inline view.

```
SELECT DepartmentID, Name, Salary
FROM (SELECT E.*, RANK() OVER (PARTITION BY DepartmentID ORDER BY Salary DESC)
AS Rank
      FROM Employees E)
WHERE Rank = 1;
```

Output:

DepartmentID	Name	Salary
1	Eva	52000
2	David	65000
3	Charlie	70000

Experiment No. 10:

Overview of MongoDB – A NoSQL Database

1: Creating and Dropping a Database

Create a new database named `LibraryDB`

```
use LibraryDB
```

Output:

```
switched to db LibraryDB
```

Drop the database

```
db.dropDatabase()
```

Output:

```
{ "dropped" : "LibraryDB", "ok" : 1 }
```

2: Creating and Dropping a Collection

Create a collection named `Books`

```
db.createCollection("Books")
```

Output:

```
{ "ok" : 1 }
```

Drop the collection

```
db.Books.drop()
```

Output:

```
true
```

3: Data Types in MongoDB

MongoDB supports various data types such as:

- **String**: `"title": "MongoDB Guide"`
- **Number**: `"price": 500`
- **Boolean**: `"available": true`
- **Array**: `"authors": ["John", "Alice"]`

- **Object (Embedded Document)**: `"publisher": { "name": "TechBooks", "year": 2024 }`
- **Date**: `"publishedDate": ISODate("2024-03-27")`
- **Null**: `"review": null`

4: Insert Document

Insert a single document into the `Books` collection.

```
db.Books.insertOne({
    "title": "MongoDB Basics",
    "author": "John Doe",
    "price": 599,
    "available": true,
    "publisher": { "name": "TechBooks", "year": 2023 }
})
```

Output:

```
{
  "acknowledged" : true,
  "insertedId" : ObjectId("65f3d1e4b8c5c3a7eae12345")
}
```

Insert multiple documents.

```
db.Books.insertMany([
    { "title": "NoSQL Essentials", "author": "Alice", "price": 699, "available": 
true },
    { "title": "Advanced MongoDB", "author": "Bob", "price": 799, "available": 
false }
])
```

Output:

```
{
  "acknowledged" : true,
  "insertedIds" : [
      ObjectId("65f3d1e4b8c5c3a7eae12346"),
      ObjectId("65f3d1e4b8c5c3a7eae12347")
  ]
}
```

5: Query Documents

Retrieve all books.

```
db.Books.find()
```

Output:

```
[
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12345"), "title": "MongoDB Basics",
"author": "John Doe", "price": 599, "available": true, "publisher": { "name":
"TechBooks", "year": 2023 } },
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12346"), "title": "NoSQL Essentials",
"author": "Alice", "price": 699, "available": true },
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12347"), "title": "Advanced MongoDB",
"author": "Bob", "price": 799, "available": false }
]
```

Query books with `price > 600`.

```
db.Books.find({ "price": { $gt: 600 } })
```

Output:

```
[
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12346"), "title": "NoSQL Essentials",
"author": "Alice", "price": 699, "available": true },
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12347"), "title": "Advanced MongoDB",
"author": "Bob", "price": 799, "available": false }
]
```

6: Logical Operators

Find books where the price is greater than 600 and available is `true`.

```
db.Books.find({ $and: [{ "price": { $gt: 600 } }, { "available": true }] })
```

Output:

```
[
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12346"), "title": "NoSQL Essentials",
"author": "Alice", "price": 699, "available": true }
]
```

7: Update Document

Update the price of "MongoDB Basics" to 550.

```
db.Books.updateOne({ "title": "MongoDB Basics" }, { $set: { "price": 550 } })
```

Output:

```
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

8: Delete Document

Delete the book "Advanced MongoDB".

```
db.Books.deleteOne({ "title": "Advanced MongoDB" })
```

Output:

```
{ "acknowledged" : true, "deletedCount" : 1 }
```

9: Projection (Selecting Specific Fields)

Retrieve only `title` and `price` of all books.

```
db.Books.find({}, { "title": 1, "price": 1, "_id": 0 })
```

Output:

```
[
  { "title": "MongoDB Basics", "price": 550 },
  { "title": "NoSQL Essentials", "price": 699 }
]
```

10: Limit Records

Retrieve only the first two books.

```
db.Books.find().limit(2)
```

Output:

```
[
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12345"), "title": "MongoDB Basics",
"author": "John Doe", "price": 550, "available": true, "publisher": { "name":
"TechBooks", "year": 2023 } },
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12346"), "title": "NoSQL Essentials",
"author": "Alice", "price": 699, "available": true }
]
```

11: Sorting Documents

Sort books by price in descending order.

```
db.Books.find().sort({ "price": -1 })
```

Output:

```
[
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12346"), "title": "NoSQL Essentials",
"author": "Alice", "price": 699, "available": true },
  { "_id" : ObjectId("65f3d1e4b8c5c3a7eae12345"), "title": "MongoDB Basics",
"author": "John Doe", "price": 550, "available": true }
]
```

Subject Code: PCIT-105

Subject Name: Python Programming Laboratory

Programme: B.Tech. (IT)	**L: 0 T: 0 P: 2**
Semester: 4	**Teaching Hours: 26**
Theory/Practical: Practical	**Credits: 1**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE): 1.5 hr**
Total Marks: 50	**Elective Status: Compulsory**

Name of practicals:

1. Data types, Operators and Expressions:

a. The tax calculator program of the case study outputs a floating-point number that might show more than two digits of precision. Use the round function to modify the program to display at most two digits of precision in the output number.

b. You can calculate the surface area of a cube if you know the length of an edge. Write a program that takes the length of an edge (an integer) as input and prints the cube's surface area as output.

2. Loops and Selection Statements :

a. Write a program that accepts the lengths of three sides of a triangle as inputs. The program output should indicate whether or not the triangle is an equilateral triangle.

b. Write a program that accepts the lengths of three sides of a triangle as inputs. The program output should indicate whether or not the triangle is a right triangle. Recall from the Pythagorean theorem that in a right triangle, the square of one side equals the sum of the squares of the other two sides.

3. Strings and Text Files:

a. Write a script that inputs a line of plaintext and a distance value and outputs an encrypted text using a Caesar cipher. The script should work for any printable characters.

b. Write a script that inputs a line of encrypted text and a distance value and outputs plaintext using a Caesar cipher. The script should work for any printable characters.

4. Lists and Dictionaries :

a. Write a program that allows the user to navigate the lines of text in a file. The program should prompt the user for a filename and input the lines of text into a list. The program then enters a loop in which it prints the number of lines in the file and prompts the user for a line number. Actual line numbers range from 1 to the number of lines in the file. If the input is 0, the program quits. Otherwise, the program prints the line associated with that number.

b. Make the following modifications to the original sentence-generator program:

> a. The prepositional phrase is optional. (It can appear with a certain probability.)
>
> b. A conjunction and a second independent clause are optional: The boy took a drink and the girl played baseball.
>
> c. An adjective is optional: The girl kicked the red ball with a sore foot.

5. Design with Functions:

a. A list is sorted in ascending order if it is empty or each item except the last one is less than or equal to its successor. Define a predicate isSorted that expects a list as an argument and returns True if the list is sorted, or returns False otherwise. (Hint: For a list of length 2 or greater, loop through the list and compare pairs of items, from left to right, and return False if the first item in a pair is greater.)

b. Add a command to this chapter's case study program that allows the user to view the contents of a file in the current working directory. When the command is selected, the program should display a list of filenames and a prompt for the name of the file to be viewed. Be sure to include error recovery

6. Graphical User Interfaces:

a. Write a GUI-based program that allows the user to convert temperature values between degrees Fahrenheit and degrees Celsius. The interface should have labeled entry fields for these two values. These components should be arranged in a grid where the labels occupy the first row and the corresponding fields occupy the second row. At start-up, the Fahrenheit field should contain 32.0, and the Celsius field should contain 0.0. The third row in the window contains two command buttons, labeled >>>> and <<<<. When the user presses the first button, the program should use the data in the Fahrenheit field to compute the Celsius value, which should then be output to the Celsius field. The second button should perform the inverse function

b. Modify the temperature conversion program so that it responds to the user's press of the return or enter key. If the user presses this key when the insertion point is in a given field, the action which uses that field for input is triggered

7. Design with Classes:

a. Add three methods to the Student class that compare two Student objects. One method should test for equality. A second method should test for less than. The third method should test for greater than or equal to. In each case, the method returns the result of the comparison of the two students' names. Include a main function that tests all of the comparison operators.

b. This project assumes that you have completed Project 1. Place several Student objects into a list and shuffle it. Then run the sort method with this list and display all of the students' information.

Experiment 1a:

Tax Calculator with Rounded Output

Code:

```
# Constants
TAX_RATE = 0.20  # 20% tax rate

# Input: Getting income from user
income = float(input("Enter your income: "))

# Calculate tax
tax = income * TAX_RATE

# Output tax amount rounded to two decimal places
print("The tax amount is:", round(tax, 2))
```

Output:

```
Enter your income: 50000
The tax amount is: 10000.0
```

Experiment 1b:

Cube Surface Area Calculation

Code:

```
# Input: Getting the edge length of the cube
edge_length = int(input("Enter the edge length of the cube: "))

# Calculate surface area (6 * edge_length^2)
surface_area = 6 * (edge_length ** 2)

# Output the surface area
print("The surface area of the cube is:", surface_area)
```

Output:

```
Enter the edge length of the cube: 5
The surface area of the cube is: 150
```

Experiment 2a:

Checking for an Equilateral Triangle

Code:

```
# Input: Getting the lengths of three sides of a triangle
side1 = float(input("Enter the length of first side: "))
side2 = float(input("Enter the length of second side: "))
side3 = float(input("Enter the length of third side: "))


# Check if all sides are equal
if side1 == side2 == side3:
    print("The given triangle is an Equilateral Triangle.")
else:
    print("The given triangle is NOT an Equilateral Triangle.")
```

Output:

```
Enter the length of first side: 5
Enter the length of second side: 5
Enter the length of third side: 5
The given triangle is an Equilateral Triangle.
```

Experiment 2b:

Checking for a Right Triangle

Code:

```
# Input: Getting the lengths of three sides of a triangle

a = float(input("Enter the length of first side: "))

b = float(input("Enter the length of second side: "))

c = float(input("Enter the length of third side: "))


# Sorting the sides to ensure the largest side is considered as
hypotenuse

sides = sorted([a, b, c])


# Check if it satisfies the Pythagorean theorem

if sides[2]**2 == sides[0]**2 + sides[1]**2:

    print("The given triangle is a Right Triangle.")

else:

    print("The given triangle is NOT a Right Triangle.")
```

Output:

```
Enter the length of first side: 3

Enter the length of second side: 4

Enter the length of third side: 5

The given triangle is a Right Triangle.
```

Experiment 3a:

Caesar Cipher Encryption

Code:

```
# Input: Getting plaintext and distance value
plaintext = input("Enter the plaintext: ")
distance = int(input("Enter the distance value: "))

# Encrypt the text using Caesar cipher
encrypted_text = ""
for char in plaintext:
    encrypted_text += chr(ord(char) + distance)

# Output the encrypted text
print("Encrypted text:", encrypted_text)
```

Output:

```
Enter the plaintext: Hello123
Enter the distance value: 3
Encrypted text: Khoor456
```

Experiment 3b:

Caesar Cipher Decryption

Code:

```
# Input: Getting encrypted text and distance value
encrypted_text = input("Enter the encrypted text: ")
distance = int(input("Enter the distance value: "))

# Decrypt the text using Caesar cipher
decrypted_text = ""
for char in encrypted_text:
    decrypted_text += chr(ord(char) - distance)

# Output the decrypted text
print("Decrypted text:", decrypted_text)
```

Output:

```
Enter the encrypted text: Khoor456
Enter the distance value: 3
Decrypted text: Hello123
```

Experiment 4a:

Navigating Lines in a File

Code:

```
# Prompt the user for a filename
filename = input("Enter the filename: ")

try:
    # Read file contents into a list
    with open(filename, "r") as file:
        lines = file.readlines()

    # Loop for navigating through the file
    while True:
        print(f"\nThe file has {len(lines)} lines.")
        line_number = int(input("Enter a line number (0 to quit): "))

        if line_number == 0:
            print("Exiting program.")
            break
        elif 1 <= line_number <= len(lines):
            print(f"Line {line_number}: {lines[line_number - 1].strip()}")
        else:
            print("Invalid line number. Please enter a valid number.")

except FileNotFoundError:
    print("Error: File not found. Please check the filename and try again.")
```

Output:

```
Enter the filename: sample.txt


The file has 5 lines.
Enter a line number (0 to quit): 3
Line 3: This is the third line of the file.


Enter a line number (0 to quit): 0
Exiting program.
```

Experiment 4b:

Enhanced Sentence Generator

Code:

```
import random

# Define word lists
nouns = ["boy", "girl", "dog", "cat", "man", "woman"]
verbs = ["runs", "jumps", "plays", "eats", "sleeps", "kicks"]
adjectives = ["red", "big", "small", "happy", "sad", "sore"]
prepositions = ["with", "in", "on", "under", "over", "beside"]
articles = ["the", "a", "an"]
conjunctions = ["and", "but", "or", "so"]

# Function to generate a random sentence
def generate_sentence():
    noun1 = random.choice(nouns)
    verb1 = random.choice(verbs)
    article1 = random.choice(articles)
    sentence = f"{article1.capitalize()} {noun1} {verb1}"

    # Optionally add an adjective
    if random.random() < 0.5:
        adjective = random.choice(adjectives)
        sentence = f"{article1.capitalize()} {adjective} {noun1}
{verb1}"

    # Optionally add a prepositional phrase
    if random.random() < 0.5:
        preposition = random.choice(prepositions)
        article2 = random.choice(articles)
        noun2 = random.choice(nouns)
        sentence += f" {preposition} {article2} {noun2}"

    # Optionally add a conjunction and another clause
    if random.random() < 0.5:
        conjunction = random.choice(conjunctions)
        article3 = random.choice(articles)
        noun3 = random.choice(nouns)
        verb2 = random.choice(verbs)
        sentence += f" {conjunction} {article3} {noun3} {verb2}"

    return sentence + "."

# Generate and display a random sentence
print(generate_sentence())
```

Output Example 1:

```
The happy dog jumps over a boy.
```

Output Example 2:

```
A cat sleeps and the girl plays.
```

Experiment 5a:

Checking if a List is Sorted

Code:

```
# Function to check if a list is sorted in ascending order
def isSorted(lst):
    for i in range(len(lst) - 1):
        if lst[i] > lst[i + 1]:  # Compare adjacent elements
            return False
    return True


# Example usage
numbers = list(map(int, input("Enter numbers separated by space: 
").split()))
print("Is the list sorted?", isSorted(numbers))
```

Output Example 1 (Sorted List):

```
Enter numbers separated by space: 1 2 3 4 5
Is the list sorted? True
```

Output Example 2 (Unsorted List):

```
Enter numbers separated by space: 5 3 7 1
Is the list sorted? False
```

Experiment 5b:

Viewing Files in the Current Directory

Code:

```
import os

# Function to list and view files
def view_file():
    files = [f for f in os.listdir() if os.path.isfile(f)]  # List
all files in the current directory

    if not files:
        print("No files available in the current directory.")
        return

    print("\nAvailable files:")
    for idx, file in enumerate(files, start=1):
        print(f"{idx}. {file}")

    filename = input("\nEnter the filename to view (or 'exit' to
quit): ")

    if filename.lower() == 'exit':
        return

    if filename in files:
        try:
            with open(filename, 'r') as file:
                print("\nFile Contents:\n")
                print(file.read())
        except Exception as e:
            print("Error reading file:", e)
    else:
        print("Error: File not found.")

# Run the function
view_file()
```

Output Example:

```
Available files:
1. sample.txt
2. data.csv
3. report.docx

Enter the filename to view (or 'exit' to quit): sample.txt

File Contents:

This is a sample text file.
It contains multiple lines.
```

Experiment 6a:

GUI-based Temperature Converter

Code:

```
import tkinter as tk

# Function to convert Fahrenheit to Celsius
def fahrenheit_to_celsius():
    try:
        fahrenheit = float(fahrenheit_entry.get())
        celsius = (fahrenheit - 32) * 5 / 9
        celsius_entry.delete(0, tk.END)
        celsius_entry.insert(0, f"{celsius:.2f}")
    except ValueError:
        celsius_entry.delete(0, tk.END)
        celsius_entry.insert(0, "Invalid input")

# Function to convert Celsius to Fahrenheit
def celsius_to_fahrenheit():
    try:
        celsius = float(celsius_entry.get())
        fahrenheit = (celsius * 9 / 5) + 32
        fahrenheit_entry.delete(0, tk.END)
        fahrenheit_entry.insert(0, f"{fahrenheit:.2f}")
    except ValueError:
        fahrenheit_entry.delete(0, tk.END)
        fahrenheit_entry.insert(0, "Invalid input")

# Create main window
root = tk.Tk()
root.title("Temperature Converter")
```

```
# Labels
tk.Label(root, text="Fahrenheit").grid(row=0, column=0)
tk.Label(root, text="Celsius").grid(row=0, column=1)

# Entry fields with default values
fahrenheit_entry = tk.Entry(root)
fahrenheit_entry.grid(row=1, column=0)
fahrenheit_entry.insert(0, "32.0")

celsius_entry = tk.Entry(root)
celsius_entry.grid(row=1, column=1)
celsius_entry.insert(0, "0.0")

# Buttons for conversion
btn_f_to_c = tk.Button(root, text=">>>>",
command=fahrenheit_to_celsius)
btn_f_to_c.grid(row=2, column=0)

btn_c_to_f = tk.Button(root, text="<<<<",
command=celsius_to_fahrenheit)
btn_c_to_f.grid(row=2, column=1)

# Run the GUI
root.mainloop()
```

Output:

- The GUI opens with two entry fields: **Fahrenheit (32.0)** and **Celsius (0.0)**.
- Clicking **>>>>** converts Fahrenheit to Celsius.
- Clicking **<<<<** converts Celsius to Fahrenheit.

Experiment 6b:

Modifying the Program to Respond to the Enter Key

Code:

```
import tkinter as tk

# Function to convert Fahrenheit to Celsius
def fahrenheit_to_celsius(event=None):
    try:
        fahrenheit = float(fahrenheit_entry.get())
        celsius = (fahrenheit - 32) * 5 / 9
        celsius_entry.delete(0, tk.END)
        celsius_entry.insert(0, f"{celsius:.2f}")
    except ValueError:
        celsius_entry.delete(0, tk.END)
        celsius_entry.insert(0, "Invalid input")

# Function to convert Celsius to Fahrenheit
def celsius_to_fahrenheit(event=None):
    try:
        celsius = float(celsius_entry.get())
        fahrenheit = (celsius * 9 / 5) + 32
        fahrenheit_entry.delete(0, tk.END)
        fahrenheit_entry.insert(0, f"{fahrenheit:.2f}")
    except ValueError:
        fahrenheit_entry.delete(0, tk.END)
        fahrenheit_entry.insert(0, "Invalid input")

# Create main window
root = tk.Tk()
root.title("Temperature Converter")
```

```
# Labels
tk.Label(root, text="Fahrenheit").grid(row=0, column=0)
tk.Label(root, text="Celsius").grid(row=0, column=1)

# Entry fields with default values
fahrenheit_entry = tk.Entry(root)
fahrenheit_entry.grid(row=1, column=0)
fahrenheit_entry.insert(0, "32.0")
fahrenheit_entry.bind("<Return>", fahrenheit_to_celsius)  # Bind
Enter key

celsius_entry = tk.Entry(root)
celsius_entry.grid(row=1, column=1)
celsius_entry.insert(0, "0.0")
celsius_entry.bind("<Return>", celsius_to_fahrenheit)  # Bind
Enter key

# Buttons for conversion
btn_f_to_c = tk.Button(root, text=">>>>",
command=fahrenheit_to_celsius)
btn_f_to_c.grid(row=2, column=0)

btn_c_to_f = tk.Button(root, text="<<<<",
command=celsius_to_fahrenheit)
btn_c_to_f.grid(row=2, column=1)

# Run the GUI
root.mainloop()
```

Modifications in Experiment 6b:

- Pressing **Enter** in the **Fahrenheit field** triggers conversion to Celsius.
- Pressing **Enter** in the **Celsius field** triggers conversion to Fahrenheit.

Experiment 7a:

Comparing Two Student Objects

Code:

```
import random


class Student:
    def __init__(self, name, roll_no, marks):
        self.name = name
        self.roll_no = roll_no
        self.marks = marks

    # Method to check equality based on name
    def __eq__(self, other):
        return self.name == other.name

    # Method to check less than based on name
    def __lt__(self, other):
        return self.name < other.name

    # Method to check greater than or equal to based on name
    def __ge__(self, other):
        return self.name >= other.name

    def __str__(self):
        return f"Student(Name: {self.name}, Roll No:
{self.roll_no}, Marks: {self.marks})"

# Main function to test comparison methods
def main():
    student1 = Student("Alice", 101, 89)
    student2 = Student("Bob", 102, 92)
    print("Comparing students:")
    print(f"Is {student1.name} equal to {student2.name}? {student1 == student2}")
    print(f"Is {student1.name} less than {student2.name}? {student1 < student2}")
    print(f"Is {student1.name} greater than or equal to {student2.name}? {student1 >= student2}")


if __name__ == "__main__":
    main()
```

Output Example:

```
Comparing students:
Is Alice equal to Bob? False
Is Alice less than Bob? True
Is Alice greater than or equal to Bob? False
```

Experiment 7b:

Sorting a List of Students

Code:

```
import random


class Student:
    def __init__(self, name, roll_no, marks):
        self.name = name
        self.roll_no = roll_no
        self.marks = marks


    def __lt__(self, other):
        return self.name < other.name


    def __str__(self):
        return f"Student(Name: {self.name}, Roll No:
{self.roll_no}, Marks: {self.marks})"


# Creating a list of students
students = [
    Student("Charlie", 103, 75),
    Student("Alice", 101, 89),
    Student("Eve", 105, 95),
    Student("Bob", 102, 92),
    Student("David", 104, 85)
]


# Shuffle the student list
random.shuffle(students)


print("Shuffled student list:")
```

```
for student in students:
    print(student)

# Sorting the students by name
students.sort()

print("\nSorted student list:")
for student in students:
    print(student)
```

Output Example:

```
Shuffled student list:
Student(Name: Bob, Roll No: 102, Marks: 92)
Student(Name: Charlie, Roll No: 103, Marks: 75)
Student(Name: Alice, Roll No: 101, Marks: 89)
Student(Name: Eve, Roll No: 105, Marks: 95)
Student(Name: David, Roll No: 104, Marks: 85)

Sorted student list:
Student(Name: Alice, Roll No: 101, Marks: 89)
Student(Name: Bob, Roll No: 102, Marks: 92)
Student(Name: Charlie, Roll No: 103, Marks: 75)
Student(Name: David, Roll No: 104, Marks: 85)
Student(Name: Eve, Roll No: 105, Marks: 95)
```

Subject Code: LPCIT-106

Subject Name: Operating System and Microprocessors Lab

Programme: B.Tech. (IT)	**L: 0 T: 0 P: 2**
Semester: 4	**Teaching Hours: 26**
Theory/Practical: Practical	**Credits: 1**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE):** **2 hr**
Total Marks: 50	**Elective Status: Compulsory**

Name of Practicals:

1. Installation Process of various Operating Systems.

2. Virtualization, Installation of Virtual Machine Software and installation of Operating System on Virtual Machine.

3. Basics of Shell programming and various types of shells.

4. Implementation of shell variables and shell keywords.

5. Implement conditional statements, looping statement and case statement in Shell programming.

6. Study and usage of vi Editor.

7. To learn and Perform addition operation for two 8-bit; sum is 8-bit and 16 bit respectively

8. Perform subtraction operation of two 8-bit and 16-bit numbers.

9. Find 1’s complement of 8 bit and 16 bit number.

10. Find 2’s complement of 8 bit and 16 bit number.

11. Find sum of series of 8 bit numbers

12. Addition and subtraction of 8-bit number using 8051

Experiment 1:

Installation Process of Various Operating Systems

Installation Steps for Windows 10/11:

1. **Create Bootable Media:**
    - Download the Windows ISO file from Microsoft's official site.
    - Use **Rufus** or the **Windows Media Creation Tool** to create a bootable USB.
2. **Boot from USB:**
    - Restart your computer and enter BIOS/UEFI (press **F2, F12, DEL, or ESC**, depending on your system).
    - Change the boot order to prioritize the USB drive.
3. **Begin Installation:**
    - Select language, time, and keyboard layout.
    - Click **Install Now** and enter the product key (or choose "I don't have a product key").
4. **Select Installation Type:**
    - Choose **Custom Installation** (for a fresh install).
    - Select the drive and format it if necessary.
5. **Complete Setup:**
    - Windows will copy files and restart multiple times.
    - Set up user accounts, preferences, and connect to Wi-Fi.

Installation Steps for Ubuntu (Linux-based OS):

1. **Download Ubuntu ISO:**
    - Get it from Ubuntu's official site.
2. **Create Bootable USB:**
    - Use **Rufus** (Windows) or `dd` command (Linux/macOS) to make a bootable drive.
3. **Boot from USB:**
    - Restart the system and enter BIOS/UEFI.
    - Set the USB as the first boot device.
4. **Install Ubuntu:**
    - Select **"Try Ubuntu"** or **"Install Ubuntu"**.
    - Choose keyboard layout and configure network settings.

5. **Partitioning:**
    - Select **"Erase Disk and Install Ubuntu"** (or manual partitioning for dual boot).
6. **Create User and Finalize Installation:**
    - Set up a username, password, and time zone.
    - Wait for installation to complete and restart.

Installation Steps for macOS (on Mac computers):

1. **Download macOS:**
    - Get the latest macOS version from the **App Store** or **Apple's website**.
2. **Create a Bootable USB (if required):**
    - Use **Terminal** with the `createinstallmedia` command.
3. **Boot into Recovery Mode:**
    - Restart and hold **Command (⌘) + R** until the Apple logo appears.
4. **Erase Disk (If Needed):**
    - Open **Disk Utility**, select your disk, and format it as **APFS** or **Mac OS Extended (Journaled)**.
5. **Install macOS:**
    - Select **"Reinstall macOS"** and follow on-screen instructions.
6. **Setup macOS:**
    - Create a user account, configure Wi-Fi, and restore files if needed.

Experiment 2:

Virtualization, Installation of Virtual Machine Software, and Installation of OS on a Virtual Machine

1. Introduction to Virtualization

Virtualization allows multiple operating systems to run on a single physical machine using software like **VMware Workstation**, **Oracle VirtualBox**, or **Microsoft Hyper-V**. It creates virtual machines (VMs) that act as independent computers.

2. Installation of Virtual Machine Software

Using Oracle VirtualBox (Example)

1. **Download and Install VirtualBox:**
    - Visit VirtualBox Official Site and download the latest version.
    - Run the installer and follow the setup wizard.
2. **Install Extension Pack (Optional but Recommended):**
    - Download and install the VirtualBox Extension Pack for added features like USB support and remote desktop.

3. Installing an Operating System on a Virtual Machine

Step 1: Creating a Virtual Machine (VM)

1. Open **VirtualBox** and click **New**.
2. Enter a name for the VM (e.g., "UbuntuVM") and select the OS type and version.
3. Allocate **RAM** (recommended: **2GB for Linux, 4GB+ for Windows**).
4. Create a **virtual hard disk** (recommended: **20GB or more, VDI format, dynamically allocated**).

Step 2: Installing the OS

1. Select the created VM and click **Start**.
2. Choose the **ISO file** of the OS (e.g., Ubuntu ISO or Windows ISO).
3. Follow the standard OS installation process (as covered in Experiment 1).
4. After installation, remove the ISO and restart the VM.

Experiment 3:

Basics of Shell Programming and Various Types of Shells

1. Introduction to Shell Programming

A shell is a command-line interface that allows users to interact with the operating system. Shell scripting is a way to automate tasks using shell commands.

2. Various Types of Shells

Shell Type	Command	Description
Bourne Shell (sh)	`/bin/sh`	Original Unix shell, fast and lightweight.
Bash (Bourne Again Shell)	`/bin/bash`	Default shell in most Linux distributions, supports scripting.
C Shell (csh, tcsh)	`/bin/csh`	Syntax similar to C programming language.
Korn Shell (ksh)	`/bin/ksh`	Combines features of Bourne and C shell.
Z Shell (zsh)	`/bin/zsh`	Advanced shell with better auto-completion and customization.

3. Basics of Shell Scripting

Example 1: A Simple Shell Script

```
#!/bin/bash
echo "Hello, this is my first shell script!"
```

Steps to Run:

1. Save the script as `script.sh`.
2. Give execute permission:

   ```
   chmod +x script.sh
   ```

3. Run the script:

   ```
   ./script.sh
   ```

Output:

```
Hello, this is my first shell script!
```

Example 2: A Script to Add Two Numbers

```
#!/bin/bash
echo "Enter two numbers:"
read a b
sum=$((a + b))
echo "Sum is: $sum"
```

Output Example:

```
Enter two numbers:
5 10
Sum is: 15
```

Experiment 4:

Implementation of Shell Variables and Shell Keywords

1. Shell Variables

Shell variables store values that can be used in scripts or commands.
There are two types:

1. **User-defined variables** (created by users).
2. **System-defined variables** (predefined by the shell).

Example: User-Defined Variables

```
#!/bin/bash
name="Abhishek"
age=21
echo "My name is $name and I am $age years old."
```

Output:

```
My name is Abhishek and I am 21 years old.
```

Example: System-Defined Variables

```
#!/bin/bash
echo "Current User: $USER"
echo "Home Directory: $HOME"
echo "Shell Type: $SHELL"
echo "Present Working Directory: $PWD"
```

Output:

```
Current User: abhishek
Home Directory: /home/abhishek
Shell Type: /bin/bash
Present Working Directory: /home/abhishek/projects
```

2. Shell Keywords

Shell keywords are reserved words used in scripting for specific operations.
Some common shell keywords:

Keyword	Description
`if`	Conditional statement
`then`	Used with `if`
`else`	Alternative condition
`elif`	Else-if condition
`for`	Looping structure
`while`	Looping structure
`case`	Switch-case condition
`do`	Executes a loop
`done`	Marks the end of a loop

Keyword	Description
`function`	Defines a function
`exit`	Terminates a script

Example: Using Shell Keywords in a Script

```
#!/bin/bash
echo "Enter a number:"
read num
if [ $num -gt 10 ]; then
    echo "The number is greater than 10."
else
    echo "The number is 10 or less."
fi
```

Output Example:

```
Enter a number: 15
The number is greater than 10.
```

Experiment 5:

Implementation of Conditional Statements, Looping Statements, and Case Statements in Shell Programming

1. Conditional Statements in Shell

Conditional statements are used to execute a block of code based on conditions.

Example: If-Else Statement

```
#!/bin/bash
echo "Enter a number:"
read num

if [ $num -gt 0 ]; then
    echo "The number is positive."
elif [ $num -lt 0 ]; then
    echo "The number is negative."
else
    echo "The number is zero."
fi
```

Output Example:

```
Enter a number: 5
The number is positive.
```

2. Looping Statements in Shell

Loops allow execution of a block of code multiple times.

Example: For Loop

```
#!/bin/bash
for i in 1 2 3 4 5
do
    echo "Number: $i"
done
```

Output:

```
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
```

Example: While Loop

```
#!/bin/bash
count=1
while [ $count -le 5 ]
do
    echo "Count is: $count"
    count=$((count + 1))
done
```

Output:

```
Count is: 1
Count is: 2
Count is: 3
Count is: 4
Count is: 5
```

Example: Until Loop

```
#!/bin/bash
num=1
until [ $num -gt 5 ]
do
    echo "Number: $num"
    num=$((num + 1))
done
```

Output:

```
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
```

3. Case Statement in Shell

The `case` statement is used for multiple conditional checks.

Example: Case Statement

```
#!/bin/bash
echo "Enter a choice (1-3):"
read choice

case $choice in
    1) echo "You selected Option 1" ;;
    2) echo "You selected Option 2" ;;
    3) echo "You selected Option 3" ;;
    *) echo "Invalid choice!" ;;
esac
```

Output Example:

```
Enter a choice (1-3): 2
You selected Option 2
```

Experiment 6:

Study and Usage of vi Editor

Introduction to vi Editor

The `vi` editor (Visual Editor) is a powerful text editor available in Unix/Linux systems. It has two main modes:

1. **Command Mode** – Used for navigation and commands.
2. **Insert Mode** – Used for inserting and editing text.

Opening vi Editor

To open a file in `vi`, use:

```
vi filename
```

- If `filename` exists, it will be opened.
- If `filename` does not exist, a new file will be created.

Modes in vi Editor

1. **Command Mode (Default Mode)**
   - Press `Esc` to enter command mode.
   - Commands are executed here.
2. **Insert Mode (For Editing)**
   - Press `i` → Insert before cursor.
   - Press `a` → Insert after cursor.
   - Press `o` → Open a new line below.
   - Press `O` → Open a new line above.
3. **Visual Mode (For Selecting Text)**
   - Press `v` → Select text character by character.
   - Press `V` → Select text line by line.

Basic Navigation Commands in vi

Command	Description
`h`	Move left
`l`	Move right
`k`	Move up
`j`	Move down
`0`	Move to the beginning of the line

Command	Description
`$`	Move to the end of the line
`w`	Move to the next word
`b`	Move to the previous word
`G`	Move to the last line of the file
`gg`	Move to the first line of the file

Editing Commands in vi

Command	Description
`i`	Insert before cursor
`a`	Insert after cursor
`o`	Open a new line below
`O`	Open a new line above
`dd`	Delete the entire line
`dw`	Delete a word
`x`	Delete a character
`u`	Undo the last change
`Ctrl + r`	Redo the last undone change
`yy`	Copy (yank) a line
`p`	Paste after cursor
`P`	Paste before cursor

Saving and Exiting in vi

Command	Description
`:w`	Save file
`:q`	Quit editor
`:wq` or `ZZ`	Save and quit
`:q!`	Quit without saving
`:x`	Save and quit (if changes are made)

Search and Replace in vi

Command	Description
`/word`	Search for `word` forward
`?word`	Search for `word` backward
`n`	Repeat last search forward
`N`	Repeat last search backward
`:%s/old/new/g`	Replace `old` with `new` in the entire file
`:5,10s/old/new/g`	Replace `old` with `new` in lines 5 to 10

Example Usage

1. Open a new file:

   ```
   vi example.txt
   ```

2. Press `i` to enter Insert mode and type:

   ```
   vbnet
   CopyEdit
   This is an example text file.
   ```

3. Press `Esc` to switch to Command mode.
4. Save the file and exit using `:wq`.

Experiment 7:

Addition Operation for Two 8-bit Numbers (8-bit and 16-bit Sum)

Objective:

To perform the addition of two 8-bit numbers and store the result in:

1. **8-bit register (without carry).**
2. **16-bit register (with carry).**

Case 1: 8-bit Sum (Without Carry)

- If the sum fits within 8 bits, the carry flag remains clear.

Assembly Code (8085)

```
MVI A, 25H    ; Load first 8-bit number (e.g., 25H) into accumulator
MVI B, 34H    ; Load second 8-bit number (e.g., 34H) into register B
ADD B         ; Add register B to accumulator (A = A + B)
MOV C, A      ; Store result in register C
HLT           ; Halt execution
```

Output:

- **Accumulator (A) = 59H** (8-bit sum).
- **Carry Flag (CY) = 0** (No carry).

Case 2: 16-bit Sum (With Carry Handling)

- If the sum exceeds 8 bits, the carry is stored in another register.

Assembly Code (8085)

```
MVI A, 85H    ; Load first 8-bit number (e.g., 85H) into accumulator
MVI B, 96H    ; Load second 8-bit number (e.g., 96H) into register B
ADD B         ; Add register B to accumulator (A = A + B)
MOV L, A      ; Store lower byte of result in register L
MVI H, 00H    ; Initialize H register to 00H
JNC SKIP      ; If no carry, skip next instruction
INR H         ; Increment H register (store carry)
SKIP: HLT     ; Halt execution
```

Output:

- **Lower Byte (L) = 1BH**
- **Higher Byte (H) = 01H** (Stores carry bit)
- **Final Result = 011BH (16-bit sum).**

Experiment 8:

Subtraction Operation for Two 8-bit and 16-bit Numbers

Objective:

To perform the subtraction of two numbers using the **8085 microprocessor** and store the result. The result can be either:

1. **8-bit result (without borrow).**
2. **16-bit result (with borrow handling).**

Case 1: 8-bit Subtraction (Without Borrow)

- If the result is positive, the borrow flag remains clear.

Assembly Code (8085)

```
MVI A, 45H    ; Load first 8-bit number (e.g., 45H) into accumulator
MVI B, 23H    ; Load second 8-bit number (e.g., 23H) into register B
SUB B         ; Subtract B from A (A = A - B)
MOV C, A      ; Store result in register C
HLT           ; Halt execution
```

Output:

- **Accumulator (A) = 22H** (8-bit result).
- **Borrow Flag (CY) = 0** (No borrow).

Case 2: 8-bit Subtraction (With Borrow Handling)

- If the second number is larger, the borrow flag is set, and the result is stored as **2's complement**.

Assembly Code (8085)

```
MVI A, 23H    ; Load first 8-bit number (e.g., 23H) into accumulator
MVI B, 45H    ; Load second 8-bit number (e.g., 45H) into register B
SUB B         ; Subtract B from A (A = A - B)
MOV C, A      ; Store result in register C
HLT           ; Halt execution
```

Output:

- **Accumulator (A) = DBH** (2's complement representation of negative value).
- **Borrow Flag (CY) = 1** (Borrow occurred).

Case 3: 16-bit Subtraction (With Borrow Handling)

- Subtraction is performed in **two parts**: Lower byte first, then higher byte (including borrow).

Assembly Code (8085)

```
LXI H, 5678H  ; Load first 16-bit number (H = 56H, L = 78H)
LXI D, 3456H  ; Load second 16-bit number (D = 34H, E = 56H)
MOV A, L      ; Move lower byte of HL to accumulator
SUB E         ; Subtract lower byte of DE from A
MOV L, A      ; Store result in L
MOV A, H      ; Move higher byte of HL to accumulator
SBB D         ; Subtract higher byte of DE with borrow
MOV H, A      ; Store result in H
HLT           ; Halt execution
```

Output:

- **Result (HL) = 2222H** (16-bit difference).
- **Borrow Flag (CY) = 0** (No borrow).

Experiment 9:

Finding 1’s Complement of an 8-bit and 16-bit Number

Objective:

To find the **1’s complement** of a given **8-bit** and **16-bit** number using the **8085 microprocessor**.

1. **1’s complement** is obtained by inverting each bit (changing `0` to `1` and `1` to `0`).

Case 1: 1’s Complement of an 8-bit Number

- The **CMA (Complement Accumulator)** instruction is used to find the **1’s complement** of an 8-bit number.

Assembly Code (8085)

```
MVI A, 5AH    ; Load 8-bit number (e.g., 5AH) into accumulator
CMA           ; Complement all bits of A (1’s complement)
MOV B, A      ; Store result in register B
HLT           ; Halt execution
```

Output:

- **Given Number (A) = 5AH (01011010 in binary)**
- **1’s Complement (B) = A5H (10100101 in binary)**

Case 2: 1’s Complement of a 16-bit Number

- **1’s complement** is performed separately on **higher** and **lower** bytes.

Assembly Code (8085)

```
LXI H, 3C9FH  ; Load 16-bit number (H = 3CH, L = 9FH)
MOV A, L      ; Move lower byte to accumulator
CMA           ; Complement lower byte
MOV L, A      ; Store complemented value back in L
MOV A, H      ; Move higher byte to accumulator
CMA           ; Complement higher byte
MOV H, A      ; Store complemented value back in H
HLT           ; Halt execution
```

Output:

- **Given Number (HL) = 3C9FH (00111100 10011111 in binary)**
- **1’s Complement (HL) = C360H (11000011 01100000 in binary)**

Experiment 10:

Finding 2’s Complement of an 8-bit and 16-bit Number

Objective:

To find the **2’s complement** of a given **8-bit** and **16-bit** number using the **8085 microprocessor**.

- **2’s complement** is obtained by taking the **1’s complement** of the number and then **adding 1** to the result.
- It is used to represent negative numbers in binary.

Case 1: 2’s Complement of an 8-bit Number

- Steps:
    1. Take the **1’s complement** of the number.
    2. Add **1** to the result.

Assembly Code (8085)

```
MVI A, 5AH     ; Load 8-bit number (e.g., 5AH) into accumulator
CMA            ; Take 1’s complement of A
ADI 01H        ; Add 1 to get 2’s complement
MOV B, A       ; Store result in register B
HLT            ; Halt execution
```

Output:

- **Given Number (A) = 5AH (01011010 in binary)**
- **1’s Complement = A5H (10100101 in binary)**
- **2’s Complement = A6H (10100110 in binary)**

Case 2: 2’s Complement of a 16-bit Number

- Steps:
    1. Take the **1’s complement** of the **higher** and **lower** bytes.
    2. Add **1** to the **lower byte**.
    3. If there is a **carry**, add it to the **higher byte**.

Assembly Code (8085)

```
LXI H, 3C9FH   ; Load 16-bit number (H = 3CH, L = 9FH)
MOV A, L       ; Move lower byte to accumulator
CMA            ; Take 1’s complement of lower byte
ADI 01H        ; Add 1 to get 2’s complement
MOV L, A       ; Store result in L
MOV A, H       ; Move higher byte to accumulator
CMA            ; Take 1’s complement of higher byte
ACI 00H        ; Add carry (if any) from lower byte addition
MOV H, A       ; Store result in H
HLT            ; Halt execution
```

Output:

- **Given Number (HL) = 3C9FH (00111100 10011111 in binary)**
- **1’s Complement = C360H (11000011 01100000 in binary)**
- **2’s Complement = C361H (11000011 01100001 in binary)**

Experiment 11:

Find Sum of a Series of 8-bit Numbers

Objective:

To compute the **sum** of a given series of **8-bit numbers** using the **8085 microprocessor**.

Algorithm:

1. Load the **number of elements** in the series.
2. Initialize the sum to **zero**.
3. Add each number in the series to the sum.
4. Store the final result.

Assembly Code (8085)

```
LXI H, 2050H  ; Load memory address of the series
MOV C, M      ; Load count of numbers in register C
INX H         ; Point to first number
MVI A, 00H    ; Clear accumulator (SUM = 0)

LOOP: ADD M   ; Add the current number to A
INX H         ; Move to the next number
DCR C         ; Decrease count
JNZ LOOP      ; Repeat until all numbers are added

STA 2060H     ; Store sum at memory location 2060H
HLT           ; Halt execution
```

Example Input:

📌 **Memory Location (2050H) Contains:**

- **2050H** → `04H` (Number of elements = 4)
- **2051H** → `12H`
- **2052H** → `15H`
- **2053H** → `0AH`
- **2054H** → `08H`

Output:

📌 **Stored at Memory Location (2060H):**

- **Sum = 3FH (00111111 in binary)**
- **Decimal Equivalent = 63**

Experiment 12:

Addition and Subtraction of 8-bit Numbers Using 8051 Microcontroller

Objective:

To perform **addition** and **subtraction** of two **8-bit numbers** using the **8051 microcontroller**.

Part A: Addition of Two 8-bit Numbers

Algorithm:

1. Load the first number into **Accumulator (A)**.
2. Load the second number into **Register B**.
3. Perform **addition (ADD A, B)**.
4. Store the result in **memory (30H)**.
5. If there is a carry, store it separately.

Assembly Code (8051)

```
MOV A, #25H     ; Load first number (25H) into A
MOV B, #17H     ; Load second number (17H) into B
ADD A, B        ; Perform A = A + B
MOV 30H, A      ; Store result at memory location 30H
HLT             ; Halt execution
```

Example Input:

- **First Number = 25H (Decimal 37)**
- **Second Number = 17H (Decimal 23)**

Output:

- **Sum = 3CH (Decimal 60) → Stored at 30H**

Part B: Subtraction of Two 8-bit Numbers

Algorithm:

1. Load the first number into **Accumulator (A)**.
2. Load the second number into **Register B**.
3. Perform **subtraction (SUBB A, B)**.
4. Store the result in **memory (31H)**.
5. If there is a borrow, handle it accordingly.

Assembly Code (8051)

```
MOV A, #25H    ; Load first number (25H) into A
MOV B, #17H    ; Load second number (17H) into B
SUBB A, B      ; Perform A = A - B
MOV 31H, A     ; Store result at memory location 31H
HLT            ; Halt execution
```

Example Input:

- **First Number = 25H (Decimal 37)**
- **Second Number = 17H (Decimal 23)**

Output:

- **Difference = 0EH (Decimal 14) → Stored at 31H**

Subject Code: LPCIT-107

Subject Name: Web Technologies Laboratory

Programme: B.Tech. (IT)	**L: 0 T: 0 P: 2**
Semester: 4	**Teaching Hours: 26**
Theory/Practical: Practical	**Credits: 1**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE): 1.5 hr**
Total Marks: 50	**Elective Status: Compulsory**

Name of Practical:

1. Create a simple web page by writing HTML using a simple text editor, Notepad. Demonstrate the following components of the web page: Page titles and Headings Paragraphs and Inline images ****Note: Validation of your HTML can be done at: http://validator.w3.org/
2. Demonstrate the use of Links, Lists and Tables in HTML. You should be able to link separate pages and create named links within a document, using them to build a "table of contents".
3. Create simple Forms in HTML and demonstrate the use of various form elements like input box, textarea, submit and radio buttons etc.
4. Demonstrate the use of cascading style sheets (CSS) (inline, internal and external) to specify various aspects of style, such as colours and text fonts and sizes, in HTML document.
5. Create an html file to implement the concept of document object model, different operations and event handling using JavaScript.
6. Demonstrate the use of various selectors, filters and event handling in jQuery.
7. Demonstrate the use of AJAX to retrieve and manipulate the web page content
8. Demonstrate the use of GET and POST methods of AJAX.
9. Creation of Web pages using HTML5 and CSS3.
10. Demonstrate the use of Bootstrap Framework.
11. Setup of development server like XAMP/ WAMP in Windows and Linux.
12. Creating web pages using PHP.
13. Handling database queries with PHP.
14. Setup of CodeIgniter framework and to study its different components.

Experiment 1:

Create a Simple Web Page Using HTML

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>My First Web Page</title>
</head>
<body>

    <h1>Welcome to My Web Page</h1>

    <h2>About This Page</h2>
    <p>This is a simple web page created using HTML. It
demonstrates the use of headings, paragraphs, and images.</p>

    <h3>Image Example</h3>
    <p>Below is an example of an inline image:</p>
    <img src="https://via.placeholder.com/300" alt="Sample Image"
width="300">

    <h3>Paragraph Example</h3>
    <p>This is an example of a paragraph. You can write multiple
lines of text here.</p>

</body>
</html>
```

Output:

The web page displays the following elements:

1. **Page Title:** "My First Web Page" (Appears in the browser tab).
2. **Headings:** H1, H2, and H3 are properly displayed.
3. **Paragraphs:** Informational text is correctly formatted.
4. **Inline Image:** Displays an image with a placeholder.

Experiment 2:

Demonstrating Links, Lists, and Tables in HTML

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>HTML Links, Lists, and Tables</title>

</head>

<body>


    <h1>HTML Links, Lists, and Tables</h1>


    <!-- Table of Contents using Named Links -->

    <h2>Table of Contents</h2>

    <ul>

        <li><a href="#links">Links in HTML</a></li>

        <li><a href="#lists">Lists in HTML</a></li>

        <li><a href="#tables">Tables in HTML</a></li>

        <li><a href="page2.html">Go to Page 2</a></li> <!--
External Page Link -->

    </ul>


    <!-- Links in HTML -->

    <h2 id="links">Links in HTML</h2>

    <p>Click the link below to visit an external website:</p>

    <a href="https://www.w3schools.com" target="_blank">Visit
W3Schools</a>


    <p>Click <a href="#top">here</a> to go back to the top.</p>
```

```html
<!-- Lists in HTML -->
<h2 id="lists">Lists in HTML</h2>

<h3>Ordered List</h3>
<ol>
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScript</li>
</ol>

<h3>Unordered List</h3>
<ul>
    <li>Apple</li>
    <li>Banana</li>
    <li>Cherry</li>
</ul>

<p>Click <a href="#top">here</a> to go back to the top.</p>

<!-- Tables in HTML -->
<h2 id="tables">Tables in HTML</h2>
<table border="1">
    <tr>
        <th>Name</th>
        <th>Age</th>
        <th>City</th>
    </tr>
    <tr>
        <td>John</td>
        <td>25</td>
        <td>New York</td>
        </tr>
        <tr>
            <td>Jane</td>
            <td>30</td>
            <td>Los Angeles</td>
        </tr>
    </table>

    <p>Click <a href="#top">here</a> to go back to the top.</p>

</body>
</html>
```

Output:

The web page successfully demonstrates:

1. **Links:**
    - External link to W3Schools.
    - Internal navigation using named links (Table of Contents).
    - Link to a second page (page2.html).
2. **Lists:**
    - Ordered list (numbered).
    - Unordered list (bullet points).
3. **Tables:**
    - Displays a table with three columns: **Name, Age, City**.

Experiment 3:

Creating Simple Forms in HTML

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>HTML Forms Example</title>

</head>

<body>


    <h1>Simple HTML Form Demonstration</h1>


    <form action="submit_form.php" method="post">


        <!-- Text Input -->

        <label for="name">Name:</label>

        <input type="text" id="name" name="name" required>

        <br><br>


        <!-- Email Input -->

        <label for="email">Email:</label>

        <input type="email" id="email" name="email" required>

        <br><br>


        <!-- Password Input -->

        <label for="password">Password:</label>

        <input type="password" id="password" name="password"
required>

        <br><br>
        <!-- Radio Buttons -->
        <label>Gender:</label>
        <input type="radio" id="male" name="gender" value="Male">
        <label for="male">Male</label>
        <input type="radio" id="female" name="gender"
value="Female">
        <label for="female">Female</label>
        <br><br>

        <!-- Checkboxes -->
        <label>Hobbies:</label>
        <input type="checkbox" id="reading" name="hobbies"
value="Reading">
        <label for="reading">Reading</label>
        <input type="checkbox" id="sports" name="hobbies"
value="Sports">
        <label for="sports">Sports</label>
        <input type="checkbox" id="music" name="hobbies"
value="Music">
        <label for="music">Music</label>
        <br><br>


        <!-- Dropdown (Select Box) -->
        <label for="country">Country:</label>
        <select id="country" name="country">
            <option value="India">India</option>
            <option value="USA">USA</option>
            <option value="UK">UK</option>
            <option value="Canada">Canada</option>
        </select>
        <br><br>


        <!-- Textarea -->
        <label for="comments">Comments:</label>
        <textarea id="comments" name="comments" rows="4"
cols="30"></textarea>
        <br><br>

        <!-- Submit and Reset Buttons -->
        <input type="submit" value="Submit">
        <input type="reset" value="Reset">

    </form>

</body>
</html>
```

Output:

The form successfully demonstrates the use of various form elements:

1. **Text Input Box** for name.
2. **Email Input** for user email.
3. **Password Input** (masked characters).
4. **Radio Buttons** to select gender.
5. **Checkboxes** for hobbies selection.
6. **Dropdown Menu** to select a country.
7. **Textarea** for user comments.
8. **Submit Button** to send data.
9. **Reset Button** to clear all inputs.

Experiment 4:

Demonstrating the Use of Cascading Style Sheets (CSS)

This experiment demonstrates **Inline, Internal, and External CSS** to style an HTML page, specifying aspects like **colors, fonts, and sizes**.

Code:

1. Inline CSS Example

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>CSS Demonstration</title>

</head>

<body>


    <h1 style="color: blue; font-size: 30px; text-align:
center;">Inline CSS Example</h1>

    <p style="color: green; font-size: 18px;">This paragraph is
styled using Inline CSS.</p>


</body>

</html>
```

2. Internal CSS Example

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>CSS Demonstration</title>

    <style>
```

```
        body {
            background-color: lightgray;
            font-family: Arial, sans-serif;
        }
        h1 {
            color: darkred;
            font-size: 32px;
            text-align: center;
        }
        p {
            color: navy;
            font-size: 20px;
            font-weight: bold;
        }
    </style>
</head>
<body>

    <h1>Internal CSS Example</h1>
    <p>This paragraph is styled using Internal CSS.</p>

</body>
</html>
```

3. External CSS Example

(Create a separate CSS file named `styles.css`)

```
body {
    background-color: lightyellow;
    font-family: Verdana, sans-serif;
}
h1 {
    color: purple;
    font-size: 28px;
    text-align: center;
}
p {
    color: darkblue;
    font-size: 18px;
}
```

(Link the External CSS in an HTML file)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Demonstration</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <h1>External CSS Example</h1>
    <p>This paragraph is styled using External CSS.</p>

</body>
</html>
```

Output:

The experiment successfully demonstrates the use of Inline, Internal, and External CSS to style an HTML document.

- **Inline CSS:** Applies styles directly inside the HTML tag (affects only that element).
- **Internal CSS:** Defined inside the `<style>` tag within the HTML document (affects the whole page).
- **External CSS:** Linked from an external file (`styles.css`), allowing multiple pages to share the same styles.

Experiment 5:

Implementing the Document Object Model (DOM) and Event Handling Using JavaScript

This experiment demonstrates the **Document Object Model (DOM) operations** and **event handling** using JavaScript.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>DOM and Event Handling</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 50px;
        }
        #box {
            width: 200px;
            height: 200px;
            background-color: lightblue;
            margin: 20px auto;
            line-height: 200px;
            font-size: 20px;
            font-weight: bold;
        }
        button {
            padding: 10px;
            font-size: 16px;
            margin: 5px;
            cursor: pointer;
        }
    </style>
</head>
<body>

    <h1>Document Object Model (DOM) and Event Handling</h1>

    <p id="message">Click the buttons to see DOM manipulation and
event handling in action.</p>

    <div id="box">Hello</div>

    <button onclick="changeText()">Change Text</button>
    <button onclick="changeColor()">Change Color</button>
    <button onclick="hideBox()">Hide</button>
    <button onclick="showBox()">Show</button>

    <script>
        function changeText() {
            document.getElementById("box").innerText = "DOM
Updated!";
        }

        function changeColor() {
            let box = document.getElementById("box");
            let colors = ["lightblue", "lightgreen", "lightcoral",
"lightgoldenrodyellow", "lightpink"];
            let randomColor = colors[Math.floor(Math.random() *
colors.length)];
            box.style.backgroundColor = randomColor;
        }
        function hideBox() {
            document.getElementById("box").style.display = "none";
        }

        function showBox() {
            document.getElementById("box").style.display =
"block";
        }
    </script>

</body>
</html>
```

Output:

The experiment successfully demonstrates the Document Object Model (DOM) operations and event handling in JavaScript.

- **DOM Manipulation:** Changes the text content inside a `<div>` using `innerText`.
- **Style Manipulation:** Changes the background color dynamically.
- **Event Handling:** Uses `onclick` event listeners to trigger changes on button clicks.
- **Visibility Control:** Hides and shows the `<div>` element using JavaScript.

Experiment 6:

Demonstrating Various Selectors, Filters, and Event Handling in jQuery

This experiment demonstrates the use of **jQuery selectors**, **filters**, and **event handling**.

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>jQuery Selectors, Filters, and Event Handling</title>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <style>

        body {

            font-family: Arial, sans-serif;

            text-align: center;

            margin: 50px;

        }

        .box {

            width: 150px;

            height: 150px;

            margin: 10px auto;

            line-height: 150px;

            font-size: 18px;

            font-weight: bold;

            color: white;

            text-align: center;

        }

        #box1 { background-color: red; }

        #box2 { background-color: green; }
        #box3 { background-color: blue; }
        .highlight { border: 5px solid yellow; }
        button {
            padding: 10px;
            font-size: 16px;
            margin: 5px;
            cursor: pointer;
        }
    </style>
</head>
<body>

    <h1>jQuery Selectors, Filters, and Event Handling</h1>

    <div id="box1" class="box">Box 1</div>
    <div id="box2" class="box">Box 2</div>
    <div id="box3" class="box">Box 3</div>

    <button id="changeColor">Change Colors</button>
    <button id="toggleVisibility">Toggle Visibility</button>
    <button id="highlightEven">Highlight Even Box</button>

    <script>
        $(document).ready(function () {
            // Event: Change background color of all boxes
            $("#changeColor").click(function () {
                $(".box").each(function () {
                    let colors = ["red", "green", "blue",
"orange", "purple"];
                    let randomColor =
colors[Math.floor(Math.random() * colors.length)];
                    $(this).css("background-color", randomColor);
                });
            });

            // Event: Toggle visibility of boxes
            $("#toggleVisibility").click(function () {
                $(".box").toggle();
            });

            // Filter: Highlight even-indexed boxes
            $("#highlightEven").click(function () {
                $(".box").removeClass("highlight"); // Remove
highlight from all
                $(".box:even").addClass("highlight"); // Highlight
only even-indexed boxes
            });

            // Additional: Change text on hover
            $(".box").hover(
                function () {
                    $(this).text("Hovered!");
                },
                function () {
                    $(this).text($
(this).attr("id").toUpperCase());
                }
            );
        });
    </script>

</body>
</html>
```

Output:

The experiment successfully demonstrates jQuery selectors, filters, and event handling.

- **Selectors Used:** `$(“.box”)`, `$(“#box1”)`, `$(“.box:even”)`
- **Filters Applied:** `:even` selector highlights even-indexed boxes
- **Event Handling:**
    - **Button Clicks:** Changes colors, toggles visibility, and highlights elements
    - **Hover Event:** Changes text inside the box when hovered

Experiment 7:

Demonstrating the Use of AJAX to Retrieve and Manipulate Web Page Content

This experiment demonstrates how to use **AJAX (Asynchronous JavaScript and XML)** to dynamically fetch data from a server and update web page content without reloading the page.

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>AJAX Example</title>

    <script
src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <style>

        body {

            font-family: Arial, sans-serif;

            text-align: center;

            margin: 50px;

        }

        #content {

            border: 2px solid black;

            padding: 20px;

            margin-top: 20px;

            width: 50%;

            display: inline-block;

        }

        button {

            padding: 10px;

            font-size: 16px;

            margin: 10px;
            cursor: pointer;
        }
    </style>
</head>
<body>

    <h1>AJAX Data Retrieval Example</h1>
    <button id="loadData">Load Data</button>
    <div id="content">Click the button to load content 
dynamically.</div>

    <script>
        $(document).ready(function () {
            $("#loadData").click(function () {
                $.ajax({
                    url: "data.txt", // Dummy text file (You can 
replace this with a JSON API)
                    type: "GET",
                    success: function (response) {
                        $("#content").html(response);
                    },
                    error: function () {
                        $("#content").html("Error loading 
content.");
                    }
                });
            });
        });
    </script>

</body>
</html>
```

Output:

The experiment successfully demonstrates AJAX-based data retrieval and web page content manipulation.

- Clicking the **"Load Data"** button triggers an AJAX request to fetch content from `"data.txt"`.
- The content inside `#content` is dynamically updated without refreshing the page.
- If the request fails, an error message is displayed.

Experiment 8:

Demonstrating the Use of GET and POST Methods in AJAX

This experiment demonstrates how to use **AJAX GET and POST methods** to send and receive data from a server.

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>AJAX GET and POST Example</title>

    <script
src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <style>

        body {

            font-family: Arial, sans-serif;

            text-align: center;

            margin: 50px;

        }

        #content {

            border: 2px solid black;

            padding: 20px;

            margin-top: 20px;

            width: 50%;

            display: inline-block;

        }

        button {

            padding: 10px;

            font-size: 16px;

            margin: 10px;

            cursor: pointer;
        }
    </style>
</head>
<body>

    <h1>AJAX GET and POST Example</h1>

    <!-- GET Request -->
    <button id="getData">Use GET Method</button>
    <div id="getResponse">Click the button to fetch data using
GET.</div>

    <!-- POST Request -->
    <h2>Submit Data Using POST</h2>
    <input type="text" id="name" placeholder="Enter your name">
    <button id="postData">Submit</button>
    <div id="postResponse">Submit your name using POST.</div>

    <script>
        $(document).ready(function () {
            // AJAX GET Request
            $("#getData").click(function () {
                $.ajax({
                    url: "data.txt",  // Dummy text file (Replace
with an actual API)
                    type: "GET",
                    success: function (response) {
                        $("#getResponse").html("GET Response: " +
response);
                    },
                    error: function () {
                        $("#getResponse").html("Error fetching
data.");
```

```
                    }
                });
            });

            // AJAX POST Request
            $("#postData").click(function () {
                let userName = $("#name").val();
                $.ajax({
                    url: "server.php",  // Dummy PHP script
(Replace with an actual API)
                    type: "POST",
                    data: { name: userName },
                    success: function (response) {
                        $("#postResponse").html("POST Response: "
+ response);
                    },
                    error: function () {
                        $("#postResponse").html("Error sending
data.");
                    }
                });
            });
        });
    </script>

</body>
</html>
```

Output:

The experiment successfully demonstrates AJAX GET and POST methods.

- Clicking **"Use GET Method"** fetches data from `data.txt` and displays it inside `#getResponse`.
- Entering a name and clicking **"Submit"** sends data using the POST method to `"server.php"` (or a real API), and the response is displayed in `#postResponse`.

Experiment 9:

Creation of Web Pages Using HTML5 and CSS3

This experiment demonstrates how to create a **responsive web page** using **HTML5** and **CSS3**, showcasing various elements like **semantic tags, multimedia, and modern styling**.

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>HTML5 and CSS3 Web Page</title>

    <style>

        /* Global Styles */

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 0;

            background-color: #f4f4f4;

            color: #333;

        }


        /* Header Section */

        header {

            background: #333;

            color: white;

            text-align: center;

            padding: 20px;

            font-size: 24px;

        }


        /* Navigation Bar */
```

```
nav {
    text-align: center;
    background: #444;
    padding: 10px;
}

nav a {
    color: white;
    text-decoration: none;
    margin: 0 15px;
    font-size: 18px;
}

/* Main Content */
section {
    padding: 20px;
    text-align: center;
}

/* Image */
.image-container {
    margin: 20px auto;
    width: 80%;
}

img {
    max-width: 100%;
    border-radius: 10px;
}

/* Video */
video {
```

```
            width: 100%;
            max-width: 600px;
            border-radius: 10px;
        }

        /* Footer */
        footer {
            background: #333;
            color: white;
            text-align: center;
            padding: 10px;
            position: relative;
            bottom: 0;
            width: 100%;
        }

        /* Responsive Design */
        @media (max-width: 600px) {
            header {
                font-size: 20px;
            }
            nav a {
                font-size: 16px;
            }
        }
    </style>
</head>
<body>

    <header>
        Welcome to My HTML5 & CSS3 Web Page
    </header>
```

```html
    <nav>
        <a href="#">Home</a>
        <a href="#">About</a>
        <a href="#">Services</a>
        <a href="#">Contact</a>
    </nav>

    <section>
        <h2>HTML5 and CSS3 Demonstration</h2>
        <p>This is a simple web page using **HTML5** and **CSS3**,
featuring modern web elements.</p>

        <!-- Image -->
        <div class="image-container">
            <img src="https://via.placeholder.com/800x400"
alt="Sample Image">
        </div>

        <!-- Video -->
        <h3>Sample Video</h3>
        <video controls>
            <source src="sample.mp4" type="video/mp4">
            Your browser does not support the video tag.
        </video>
    </section>

    <footer>
        &copy; 2025 My Website | All Rights Reserved
    </footer>

</body>
</html>
```

Output:

Successfully created a modern web page using HTML5 and CSS3.

- **Header & Footer**: Uses `header` and `footer` elements for structured layout.
- **Navigation Bar**: Built using `nav` and styled for better user experience.
- **Multimedia Elements**: Includes an **image and a video** demonstrating HTML5 features.
- **Responsive Design**: Uses **media queries** to ensure proper display on different screen sizes.

Experiment 10:

Demonstrating the Use of Bootstrap Framework

This experiment demonstrates how to create a **responsive webpage** using **Bootstrap**, showcasing **grid systems, navigation bars, buttons, forms, and cards**.

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <title>Bootstrap Demonstration</title>


    <!-- Bootstrap CSS CDN -->

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst
rap.min.css" rel="stylesheet">


</head>

<body>


    <!-- Navigation Bar -->

    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

        <div class="container-fluid">

            <a class="navbar-brand" href="#">My Website</a>

            <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav ms-auto">

                    <li class="nav-item"><a class="nav-link"
href="#">Home</a></li>
```

```
                    <li class="nav-item"><a class="nav-link" href="#">About</a></li>
                    <li class="nav-item"><a class="nav-link" href="#">Services</a></li>
                    <li class="nav-item"><a class="nav-link" href="#">Contact</a></li>
                </ul>
            </div>
        </div>
    </nav>

    <!-- Hero Section -->
    <div class="container text-center mt-4">
        <h1 class="display-4">Welcome to Bootstrap Framework</h1>
        <p class="lead">A responsive webpage using Bootstrap 5.</p>
        <button class="btn btn-primary">Get Started</button>
    </div>

    <!-- Grid System -->
    <div class="container mt-5">
        <div class="row">
            <div class="col-md-4">
                <div class="card">
                    <img src="https://via.placeholder.com/300" class="card-img-top" alt="Image">
                    <div class="card-body">
                        <h5 class="card-title">Card 1</h5>
                        <p class="card-text">Bootstrap provides an easy-to-use grid system.</p>
                        <a href="#" class="btn btn-success">Learn More</a>
                    </div>
                </div>
            </div>
            <div class="col-md-4">
                <div class="card">
                    <img src="https://via.placeholder.com/300"
class="card-img-top" alt="Image">
                    <div class="card-body">
                        <h5 class="card-title">Card 2</h5>
                        <p class="card-text">It is mobile-friendly
and responsive.</p>
                        <a href="#" class="btn btn-success">Learn
More</a>
                    </div>
                </div>
            </div>
            <div class="col-md-4">
                <div class="card">
                    <img src="https://via.placeholder.com/300"
class="card-img-top" alt="Image">
                    <div class="card-body">
                        <h5 class="card-title">Card 3</h5>
                        <p class="card-text">Easily create layouts
using Bootstrap components.</p>
                        <a href="#" class="btn btn-success">Learn
More</a>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Contact Form -->
    <div class="container mt-5">
        <h2>Contact Us</h2>
        <form>
```

```
            <div class="mb-3">
                <label class="form-label">Name</label>
                <input type="text" class="form-control" 
placeholder="Enter your name">
            </div>
            <div class="mb-3">
                <label class="form-label">Email</label>
                <input type="email" class="form-control" 
placeholder="Enter your email">
            </div>
            <button type="submit" class="btn btn-
warning">Submit</button>
        </form>
    </div>

    <!-- Footer -->
    <footer class="bg-dark text-white text-center p-3 mt-4">
        &copy; 2025 My Website | Bootstrap Example
    </footer>

    <!-- Bootstrap JS CDN -->
    <script 
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstra
p.bundle.min.js"></script>

</body>
</html>
```

Output:

Successfully created a responsive webpage using Bootstrap.

- **Navigation Bar**: A responsive navbar with dropdown menu.
- **Grid System**: Three equal-width columns with cards.
- **Buttons & Forms**: Uses Bootstrap buttons and form controls.
- **Styling & Responsiveness**: Automatically adjusts layout for different screen sizes.

Experiment 11:

Setup of Development Server like XAMPP/WAMP in Windows and Linux

Installation of XAMPP in Windows

1. **Download XAMPP:**
   - Visit Apache Friends and download XAMPP for Windows.
2. **Install XAMPP:**
   - Run the downloaded `.exe` file and follow the installation wizard.
   - Select components like Apache, MySQL, PHP, and phpMyAdmin.
3. **Start the Services:**
   - Open the **XAMPP Control Panel** and start **Apache** and **MySQL**.
4. **Verify Installation:**
   - Open a browser and go to `http://localhost/`.
   - If you see the XAMPP welcome page, the installation is successful.

Installation of WAMP in Windows

1. **Download WAMP:**
   - Visit WAMP Official Website and download WAMP for Windows.
2. **Install WAMP:**
   - Run the downloaded `.exe` file and complete the installation.
3. **Start WAMP Server:**
   - Open **WAMP Server** from the start menu.
   - Click on the **green WAMP icon** in the system tray and select **Start All Services**.
4. **Verify Installation:**
   - Open a browser and go to `http://localhost/` to see the WAMP welcome page.

Installation of XAMPP in Linux (Ubuntu/Debian-based)

1. **Download XAMPP:**
   - Open a terminal and run:

     ```
     wget https://www.apachefriends.org/xampp-files/7.4.30/xampp-linux-
     x64-7.4.30-0-installer.run
     ```

2. **Make the File Executable:**
    - Run:

      ```
      chmod +x xampp-linux-x64-7.4.30-0-installer.run
      ```

3. **Run the Installer:**
    - Execute:

      ```
      sudo ./xampp-linux-x64-7.4.30-0-installer.run
      ```

    - Follow the installation steps.
4. **Start XAMPP:**
    - Run:

      ```
      sudo /opt/lampp/lampp start
      ```

5. **Verify Installation:**
    - Open a browser and visit `http://localhost/`.

Installation of LAMP (Apache, MySQL, PHP) in Linux (Ubuntu/Debian-based)

1. **Update Package List:**

   ```
   sudo apt update
   ```

2. **Install Apache:**

   ```
   sudo apt install apache2
   ```

    - Verify by visiting `http://localhost/`.
3. **Install MySQL:**

   ```
   sudo apt install mysql-server
   ```

    - Secure installation:

      ```
      bash
      CopyEdit
      sudo mysql_secure_installation
      ```

4. **Install PHP:**

   ```
   sudo apt install php libapache2-mod-php php-mysql
   ```

5. **Restart Apache:**

   ```
   sudo systemctl restart apache2
   ```

6. **Verify Installation:**

- Create a PHP file:

```
sudo nano /var/www/html/info.php
```

- Add the following code:

```
<?php
phpinfo();
?>
```

- Save and visit `http://localhost/info.php`.

Output:

Successfully installed and set up XAMPP, WAMP, and LAMP on Windows and Linux.

- Able to access `http://localhost/` and verify the installation.
- Web server is up and running. 🚀

Experiment 12:

Creating Web Pages Using PHP

Step 1: Setting Up PHP Environment

Before writing PHP code, ensure you have a web server (XAMPP, WAMP, or LAMP) installed and running.

- **Windows Users:** Start Apache and MySQL in **XAMPP/WAMP**.
- **Linux Users:** Use the command:

  ```
  sudo systemctl start apache2
  sudo systemctl start mysql
  ```

Step 2: Writing a Basic PHP Web Page

📌 **Create a new PHP file:**

1. Open the `htdocs` folder (for XAMPP) or `www` folder (for WAMP).
2. Create a file named `index.php`.
3. Write the following PHP code:

index.php

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>PHP Web Page</title>

</head>

<body>

    <h1>Welcome to My PHP Web Page</h1>

    <?php

        echo "<p>This is a simple PHP script running on a web
server.</p>";

        echo "<p>Current Date and Time: " . date("Y-m-d H:i:s") .
"</p>";

    ?>

</body>
```

```
</html>
```

Step 3: Running the PHP Web Page

1. Start your **XAMPP/WAMP** server.
2. Open a browser and go to:
   `http://localhost/index.php`
3. You should see:
   - A **welcome message**.
   - The **current date and time**, dynamically generated by PHP.

Step 4: Connecting PHP to MySQL Database (Optional)

To extend the PHP page with database functionality, create a MySQL connection.

database.php

```php
<?php
$servername = "localhost";
$username = "root"; // Default user in XAMPP/WAMP
$password = "";
$dbname = "test_db"; // Create this database in phpMyAdmin

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully!";
?>
```

Output:

PHP web page successfully created and running on a local server.
Displays dynamic content like the current date and time.
Optional database connection tested successfully. 🚀

Experiment 13:

Handling Database Queries with PHP

Step 1: Setting Up MySQL Database

1. Open **phpMyAdmin** from your browser:

   `http://localhost/phpmyadmin/`

2. Create a new database named **`student_db`**.

3. Inside `student_db`, create a table named **`students`** with the following columns:

   - `id` (INT, Primary Key, Auto Increment)
   - `name` (VARCHAR(50))
   - `email` (VARCHAR(50))
   - `age` (INT)

SQL Query to Create Table

```
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(50),
    age INT
);
```

Step 2: Establishing Database Connection in PHP

📌 Create a new PHP file named **`db_connect.php`** to handle the database connection.

db_connect.php

```
<?php
$servername = "localhost";
$username = "root"; // Default user in XAMPP/WAMP
$password = "";
$dbname = "student_db";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
```

```
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully!";
?>
```

Step 3: Writing PHP Queries for CRUD Operations

Create a file named `database_operations.php` and include the connection file.

database_operations.php

php

CopyEdit

```
<?php
include 'db_connect.php';

// Insert Data
$sql_insert = "INSERT INTO students (name, email, age) VALUES ('John Doe', 'john@example.com', 22)";
if ($conn->query($sql_insert) === TRUE) {
    echo "New record created successfully.<br>";
} else {
    echo "Error: " . $sql_insert . "<br>" . $conn->error;
}

// Read Data
$sql_select = "SELECT * FROM students";
$result = $conn->query($sql_select);

if ($result->num_rows > 0) {
    echo "<h3>Student Records:</h3>";
    while($row = $result->fetch_assoc()) {
        echo "ID: " . $row["id"]. " - Name: " . $row["name"]. " - Email: " . $row["email"]. " - Age: " . $row["age"]. "<br>";
    }
} else {
    echo "No records found.";
}

// Update Data
$sql_update = "UPDATE students SET age=23 WHERE name='John Doe'";
if ($conn->query($sql_update) === TRUE) {
    echo "Record updated successfully.<br>";
} else {
    echo "Error updating record: " . $conn->error;
}

// Delete Data
$sql_delete = "DELETE FROM students WHERE name='John Doe'";
if ($conn->query($sql_delete) === TRUE) {
    echo "Record deleted successfully.<br>";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

Step 4: Running the PHP Script

1. Save the file as **`database_operations.php`** in the `htdocs` (XAMPP) or `www` (WAMP) folder.
2. Open a browser and run:

   ```
   http://localhost/database_operations.php
   ```

3. The page will execute the following operations:
   Insert a new student record
   Retrieve and display all records

Update a student record
Delete a student record

Output:

```
Connected successfully!

New record created successfully.

Student Records:

ID: 1 - Name: John Doe - Email: john@example.com - Age: 22

Record updated successfully.

Record deleted successfully.
```

Experiment 14:

Setup of CodeIgniter Framework and Study of Its Different Components

Step 1: Download and Install CodeIgniter

1. **Download CodeIgniter**
   - Visit the official CodeIgniter website: https://codeigniter.com/
   - Download the latest stable version of **CodeIgniter 4**.
2. **Extract CodeIgniter**
   - Extract the downloaded ZIP file.
   - Move the extracted folder to `htdocs` (for XAMPP) or `www` (for WAMP).
   - Rename the folder to `codeigniter_app`.
3. **Run CodeIgniter**
   - Open a terminal or command prompt.
   - Navigate to the project directory:

     ```
     cd C:\xampp\htdocs\codeigniter_app
     ```
   - Start the built-in PHP server:

     ```
     php spark serve
     ```
   - Open a browser and go to:

     ```
     http://localhost:8080
     ```
   - If you see the **Welcome to CodeIgniter!** page, the setup is successful.

Step 2: Understanding CodeIgniter Folder Structure

Main directories in CodeIgniter:

1. **`app/`** → Contains the main application files (controllers, models, views).
2. **`public/`** → The entry point of the application (index.php).
3. **`system/`** → The core CodeIgniter framework files.
4. **`writable/`** → Storage for logs, cache, and session data.

Step 3: Creating a Simple Controller

Controllers handle requests and return responses.

1. Open `app/Controllers/` and create a new file **`Hello.php`**.

2. Add the following code:

```php
<?php
namespace App\Controllers;

class Hello extends BaseController {
    public function index() {
        return "Hello, CodeIgniter!";
    }
}
?>
```

3. Open your browser and visit:

   ```
   http://localhost:8080/hello
   ```

4. Output:

   ```
   Hello, CodeIgniter!
   ```

Step 4: Creating a View

Views handle HTML content.

1. Create a file **`hello_view.php`** inside `app/Views/`:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Hello Page</title>
</head>
<body>
    <h1>Welcome to CodeIgniter</h1>
</body>
</html>
```

2. Modify the `Hello` controller to load the view:

php

CopyEdit

```php
<?php
namespace App\Controllers;
```

```
class Hello extends BaseController {
    public function index() {
        return view('hello_view');
    }
}
?>
```

3. Visit:
   `http://localhost:8080/hello`
4. Output:
   `A webpage displaying "Welcome to CodeIgniter"`

Step 5: Creating a Model

Models interact with the database.

1. Create a file **`StudentModel.php`** inside `app/Models/`:

php

CopyEdit

```
<?php
namespace App\Models;
use CodeIgniter\Model;

class StudentModel extends Model {
    protected $table = 'students';
    protected $allowedFields = ['name', 'email', 'age'];
}
?>
```

2. Modify the `Hello` controller to fetch student data:

php

CopyEdit

```
<?php
namespace App\Controllers;
use App\Models\StudentModel;
```

```
class Hello extends BaseController {
    public function students() {
        $model = new StudentModel();
        $data['students'] = $model->findAll();
        return view('students_view', $data);
    }
}
?>
```

3. Create a file **`students_view.php`** inside `app/Views/`:

html

CopyEdit

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Students List</title>
</head>
<body>
    <h1>Students List</h1>
    <ul>
        <?php foreach ($students as $student): ?>
            <li><?= $student['name']; ?> - <?=
$student['email']; ?></li>
        <?php endforeach; ?>
    </ul>
</body>
</html>
```

4. Visit:

   ```
   http://localhost:8080/hello/students
   ```

5. Output:

   ```
   List of students fetched from the database.
   ```

Output Summary

Successfully set up CodeIgniter
Created a Controller (`Hello.php`)
Created a View (`hello_view.php`)
Created a Model (`StudentModel.php`)
Fetched data from the database