UG – Single Major Syllabus

(ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION)

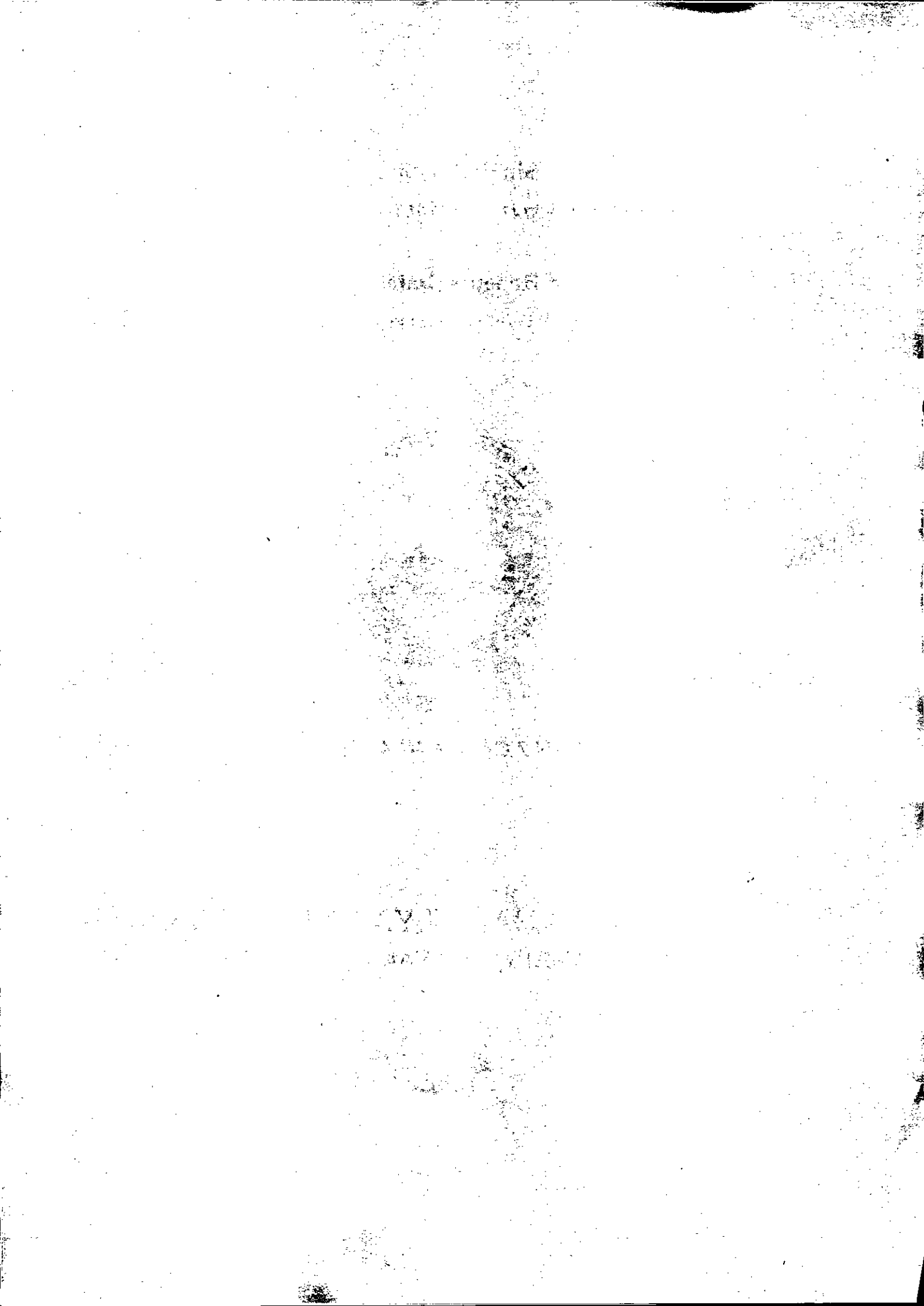
BCA Honours (Data Science)

(W.E.F. 2025-26 Admitted Batch)

DEAN ACADEMIC AFFAIRS

ADIKAVI NANNAYA UNIVERSITY

RAJAMAHENDRAVARAM, AP-533296



BCA (Data Science)- Major

Sl.No	Semester	Course	Hrs. per/week	Credits
1	I	C1: Computer Fundamentals and Office Automation	3	3
		C1P: Computer Fundamentals and Office Automation Lab	2	1
2		C2: Problem Solving Using C	3	3
		C2P: Problem Solving Using C Lab	2	1
3	II	C3: Python Programming and Data Structures	3	3
		C3P: Python programming and Data Structures lab	2	1
4		C4: Statistical Foundations for Data Science	3	3
		C4P: Statistical Foundations for Data Science lab	2	1
5	III	C5: Database Management Systems	3	3
		C5P: Database Management Systems Lab	2	1
6		C6: Data Science with R	3	3
		C6P: Data Science With R lab	2	1
7		C7: Object oriented programming using Java	3	3
		C7P: Object oriented programming using Java Lab	2	1
8	IV	C8: Data Mining & Data warehousing	3	3
		C8P: Data Mining & Data warehousing Lab	2	1
		C9: Exploratory Data Analysis and Visualization	3	3

9		C9P: Exploratory Data Analysis and Visualization lab	2	1
		C10: Software Engineering	3	3
10		C10P: Software Engineering Lab	2	1
		C11: Business Intelligence Tools	3	3
11	V	C11P: Business Intelligence Tools Lab	2	1
SEA : Skill Elective Track A (Application Development)				
		*C12: SEA1: Web Design & Development Fundamentals	3	3
12		*C12P: SEAP1: Web Design & Development Fundamentals Lab	2	1
	V-Track A	*C13: SEA2: Web Application Development using PHP & MySQL	3	3
13		*C13P: SEAP2: Web Application Development using PHP & MySQL Lab	2	1
		*C14: SEA3: Mobile Application Development	3	3
14		*C14P: SEAP3: Mobile Application Development lab	2	1
	VI - Track A	*C15: SEA4: MERN Stack	3	3
15		*C15P: SEAP4: MERN Stack Lab	2	1
SEB : Skill Elective Track B (AIML & Data Engineering)				
		*C12: SEB1: Artificial Intelligence	3	3
12		*C12P: SEBP1: Artificial Intelligence Lab	2	1
	V - Track B	*C13: SEB2: Big Data Technologies	3	3
13		*C13P: SEBP2: Big Data Technologies lab	2	1

14	VI - Track B	*C14: SEB3: Machine Learning	3	3
		*C14P: SEBP3: Machine Learning Lab	2	1
15		*C15: SEB4: Cloud computing for Data Science	3	3
		*C15P: SEBP4: Cloud computing for Data Science Lab	2	1

Semester-I

C1: Computer fundamentals and Office Automation

Course Objectives

1. **Understand foundational computing concepts,** including number systems, the evolution of computers, block diagrams, and generational progress.
2. **Develop knowledge of computer architecture,** focusing on system organization and networking fundamentals.
3. **Acquire practical skills in document creation,** formatting, and digital presentations using word processing tools.
4. **Gain proficiency in spreadsheet operations,** such as data entry, formulas, functions, and charting techniques.
5. **Introduce data visualization and basic modelling principles,** fostering analytical thinking in structuring and interpreting data sets.

Course Outcomes

1. At the End of the Course, The Students will be able to **explain different number systems,** the historical evolution of computers, and identify key components in a block diagram.
2. Learners will demonstrate **basic blocks of a computer and fundamental networking knowledge.**

Approved by the Vice-Chancellor

Dated: 17/11/2025

DEAN
Academic Affairs
Adikavi Nannaya University
RAJAHMUNDRY-533 296.

3. Learners will create professional-level documents and **design visually appealing presentations** using word processing software and presentation software.
4. Learners will manipulate data within spreadsheets, apply formulas, and **generate accurate summaries and visualizations.**
5. Learners will apply data modelling techniques to **analyze, organize, and represent data effectively** in various scenarios.

Unit 1. Number Systems, Evolution , Block Diagram and Generations:

Number Systems: Binary, Decimal, Octal, Hexadecimal; conversions between number systems.

Evolution of Computers: History from early mechanical devices to modern-day systems.

Block Diagram of a Computer: Components like Input Unit, Output Unit, Memory, CPU (ALU + CU).

Generations of Computers: First to Fifth Generation – technologies, characteristics, examples.

Unit 2. Basic organization and N/W fundamentals:

Computer Organization: Functional components – Input/Output devices, Storage types, Memory hierarchy.

Types of Computers: Micro, Mini, Mainframe, and Supercomputers.

Networking Fundamentals: Definition, need for networks, types (LAN, WAN, MAN), topology (Star, Ring, Bus).

Internet Basics: IP Address, Domain Name, Web Browser, Email, WWW.

Unit 3. Word Processing and presentations:

Word Processing Basics: Using MS Word/Google Docs – formatting, styles, tables, mail merge.

Presentation Tools: Using PowerPoint/Google Slides – slide design, animations, transitions.

Applications: Creating resumes, reports, brochures, and presentations.

Keyboard Shortcuts

Unit 4. Spreadsheet Basics:

Spreadsheet Concepts: Understanding rows, columns, cells in tools like MS Excel/Google Sheets, cell referencing.

Functions and Formulae: SUM, AVERAGE, IF, COUNT.

Charts and Graphs: Creating visual representations

Data Handling: Sorting, filtering, conditional formatting.

Text Functions: LEFT, RIGHT, MID, LEN, TRIM, CONCAT, TEXTJOIN

Advanced Functions: Logical: IF, AND, OR, IFERROR, **Lookup**: VLOOKUP, HLOOKUP, XLOOKUP, INDEX, MATCH

Unit 5. Data Analysis and Visualization:

Conditional Formatting: Custom rules, Color scales, Icon sets, Data bars

Data Analysis Tools: Pivot Tables and Pivot Charts, Data Validation (Drop-downs, Input Messages, Error Alerts), What-If Analysis: Goal Seek, Scenario Manager, Data Tables

Charts and Dashboards: Creating Interactive Dashboards, Using slicers with Pivot Tables,Combo Charts and Sparklines

Productivity Tips: Using Named Ranges, Freeze Panes, Split View

Textbooks:

1. **Fundamentals of Computers,** Reema Thareja, Oxford University Press, Second Edition
2. **Fundamentals of Computers**, V. Rajaraman – PHI Learning
3. **Introduction to Computers** by Peter Norton – McGraw Hill
4. **Microsoft Office 365 In Practice** by Randy Nordell – McGraw Hill Education

References:

1. **Excel 2021 Bible** by Michael Alexander, Richard Kusleika – Wiley
2. **Networking All-in-One For Dummies** by Doug Lowe – Wiley
3. **Microsoft Official Docs and Training**: https://learn.microsoft.com
4. **Google Workspace Learning Center**: https://support.google.com/a/users/

Activities:

Outcome: At the End of the Course, The Students will be able to **explain different number systems**, the historical evolution of computers, and identify key components in a block diagram.
Activity: Create a digital poster or infographic comparing number systems (binary, decimal, octal, hexadecimal) and illustrating the timeline of computer generations with key innovations.
Evaluation Method: Rubric-based assessment of the poster presentation on a 10-point scale focusing on:

- Accuracy of number system conversions
- Correct identification of block diagram components
- Visual organization and creativity

Outcome: Learners will demonstrate **basic blocks of a computer and fundamental networking knowledge.**
Activity: Design a concept map showing the internal architecture of a computer and types of networks (LAN, WAN, MAN), including devices and topologies.
Evaluation Method: Checklist-based peer review and instructor validation:

- Completeness of the map
- Correctness of networking concepts
- Use of appropriate terminology
- Logical flow and structure of the map

Outcome: Learners will create professional-level documents and **design visually appealing presentations** using word processing software and presentation software.
Activity: Prepare a formal report (e.g., project proposal) in a word processor and present it using a slide deck with transitions, embedded media, and design elements.
Evaluation Method: Performance-based evaluation using a 10-point scoring scale:

- Formatting and structure of the document
- Presentation aesthetics and clarity
- Communication skills during presentation

Outcome: Learners will manipulate data within spreadsheets, apply formulas, and **generate accurate summaries and visualizations.**

Activity: Analyze a dataset (e.g., student scores or sales data) using spreadsheet software. Apply formulas (SUM, AVERAGE, IF, VLOOKUP) and create relevant charts.

Evaluation Method: Practical test with a rubric:

- Correct use of formulas
- Accuracy of data summaries

Outcome: Learners will apply data modelling techniques to **analyze, organize, and represent data effectively** in various scenarios.

Activity: Prepare an interactive dashboard for a given data set using EXCEL.

Evaluation Method: Evaluation of the dashboard on a 10-point scoring scale:

- Presentation aesthetics and clarity
- Interactiveness
- Communication skills during presentation

C1P: Computer fundamentals and Office Automation Lab

List of Experiments:

1. Demonstration of Assembling and Dessembling of Computer Systems.
2. Identify and prepare notes on the type of Network topology of your institution.
3. Prepare your resume in Word.
4. Using Word, write a letter to your higher official seeking 10-days leave.
5. Prepare a presentation that contains text, audio and video.
6. Using a spreadsheet, prepare your class Time Table.
7. Using a Spreadsheet, calculate the Gross and Net salary of employees(Min 5) considering all the allowances.
8. Generate the class-wise and subject-wise results for a class of 20 students. Also generate the highest and lowest marks in each subject.
9. Using IF, AND, OR, and IFERROR to Automate Grade Evaluation.
    a. Create a table of student scores in different subjects.
    b. Use IF to assign grades (A/B/C/Fail).
    c. Use IFERROR to handle missing scores or invalid data.

10. *Employee Database Search Using VLOOKUP, HLOOKUP, XLOOKUP, INDEX, and MATCH*
    a. Create a database of employees (Name, ID, Department, Salary).
    b. Implement VLOOKUP to search by employee ID.
    c. Use HLOOKUP to extract department heads by role.
    d. Apply XLOOKUP for more flexible searches.
    e. Use INDEX + MATCH as an alternative to VLOOKUP.
11. Sales Report Analysis Using Pivot Tables and Charts
    a. Use a dataset of product sales (Product, Region, Date, Quantity, Revenue).
    b. Create Pivot Tables to summarize data by region/product.
    c. Insert Pivot Charts for visual analysis (e.g., bar, line).
    d. Add slicers to make the dashboard interactive.
12. Designing a Data Entry Form with Drop-downs and Input Rules
    a. Create a student registration form.
    b. Add drop-down lists for course selection using Data Validation.
    c. Add input messages to guide users.
    d. Add error alerts for wrong entries.
13. Monthly Budget Planning using Goal Seek and Scenario Manager
    a. Create a simple personal budget (income, expenses, savings).
    b. Use Goal Seek to determine income needed to save a desired amount.
    c. Use Scenario Manager to compare different budgeting scenarios (best/ worst/ realistic case).
    d. Create a one-variable Data Table to analyze how different expenses affect savings.
14. Dashboard Creation Using Combo Charts, Sparklines & Slicers
    a. Use existing sales or attendance data.
    b. Insert combo charts (e.g., column + line).
    c. Add sparklines to show trends.
    d. Use slicers with Pivot Tables to control dashboard elements.
    e. Finalize and format for interactivity.

C2: Problem Solving using C

Course Objectives:

1. Understand the fundamentals of computer programming, Apply structured problem-solving approaches using algorithms, flowcharts, and C programming constructs.
2. Develop efficient logic using decision-making, loop, and jump control statements.
3. Utilize derived data types like arrays and strings for modular program design.
4. Design and implement modular solutions using functions, recursive logic, pointer operations, and dynamic memory management.
5. Handle complex data structures including structures, unions, and text file operations.

Course Outcomes:

At the end of the course, students will be able to:

1. Understand basic computing concepts, programming paradigms and write structured C programs.
2. Apply control flow statements to solve logical and repetitive tasks in C.
3. Implement arrays and string operations to manage and manipulate data efficiently.
4. Design modular code using functions, recursion, and appropriate parameter passing.
5. Utilize pointers and memory operations for effective data handling. Demonstrate competence in dynamic memory allocation and text file processing.

Unit 1. Introduction to computer programming:

Introduction, Types of software, Compiler and interpreter, Concepts of Machine level, Assembly level and high-level programming, Flowcharts and Algorithms, Fundamentals of C: History of C, Features of C, C Tokens-variables and keywords and identifiers, constants and Data types, Rules for constructing variable names, Operators, Structure of C program, Input /output statements in C-Formatted and Unformatted I/O

Unit 2. Control statements:

Decision making statements: if, if else, else if ladder, switch statements. Loop control statements: while loop, for loop and do-while loop. Jump Control statements: break,continue and goto.

Unit 3. Derived data types in C:

Arrays: One Dimensional arrays - Declaration, Initialization and Memory representation; Two Dimensional arrays -Declaration, Initialization and Memory representation. Strings: Declaring & Initializing string variables; String handling functions, Character handling functions

Unit 4. Functions:

Pointers: Pointer data type, Pointer declaration, initialization, accessing values using pointers. Pointer arithmetic, Pointers and arrays.

Function Prototype, definition and calling. Return statement. Nesting of functions. Categories of functions. Recursion (Basic Concept only). Parameter Passing by address & by value. Local and Global variables. Storage classes: automatic, external, static and register.

Unit 5. Dynamic Memory Management:

Introduction, Functions-malloc, calloc, realloc, free Structures: Basics of structure, structure members, accessing structure members, nested structures, array of structures, structure and functions, structures and pointers. Unions - Union definition; difference between Structures and Unions. Working with text files - modes: opening, reading, writing and closing text files.

Text Books:

1. Programming in ANSI C, E. Balagurusamy, Tata McGraw Hill, 6 th Edn,
2. Computer fundamentals and programming in C, Reema Theraja, Oxford University Press

Reference Books:

1. Let us C, Y Kanetkar, BPB publications
2. Head First C: A Brain-Friendly Guide, David Griffiths, Dawn Griffiths

Activities:

Outcome: Understand basic computing concepts, programming paradigms and write structured C programs.

Activity: Create a concept map of computing fundamentals and programming paradigms (procedural, structured, object-oriented). Then, they write a structured C program (e.g., a calculator or student grade system) using proper syntax, indentation, and modular design.

Evaluation Method: Rubric-based Code Review & Viva to check the

- The correctness of the concept map
- Correct use of structure (main + functions)
- Identification of paradigm used
- Code readability and documentation

Outcome: Apply control flow statements to solve logical and repetitive tasks in C.

Activity: Implement a program that solves a logic puzzle (e.g., number guessing game, pattern generation, or prime number finder) using if, switch, for, while, and do-while.

Evaluation Method: Automated Test Cases + Peer Review to check the

- Correct use of control statements
- Logical correctness of output
- Efficiency and edge case handling
- Peer feedback on clarity and logic

Outcome: Implement arrays and string operations to manage and manipulate data efficiently.

Activity: Build a program that stores and arranges student marks in ascending and descending order using arrays and performs string operations like concatenation, comparing, and formatting names.

Evaluation Method: Functional Demonstration + Code Walkthrough to check the

- Correct array and string usage
- Memory efficiency
- Handling of invalid inputs
- Explanation of sorting/searching logic

Activity:

- **Recursive Problem Solver**

  Students write a modular program to solve a recursive problem (e.g., factorial, Fibonacci, or Tower of Hanoi) using functions with parameters and return values.

Evaluation Method:

- **Code Trace + Written Quiz**
    - Correct function decomposition
    - Proper parameter passing (by value/reference)
    - Recursion depth and base case handling
    - Quiz on tracing recursive calls

Outcome: Utilize pointers and memory operations for effective data handling. Demonstrate competence in dynamic memory allocation and text file processing.

Activity: Create a program that dynamically stores user input (e.g., survey responses) using pointers and writes/reads the data to/from a text file.

Evaluation Method: Memory Debugging + File I/O Assessment to check the

- Proper use of malloc, calloc, realloc, and free
- Pointer arithmetic and dereferencing
- File creation, reading, writing, and error handling
- Use of tools like Valgrind or manual memory trace (Optional for Unix flavours)

C2P: Problem Solving using C Lab

List of Experiments:

1. Write a program to check whether the given number is Armstrong or not.
2. Write a program to find the sum of individual digits of a positive integer.
3. Write a program to generate the first n terms of the Fibonacci sequence.
4. Write a program to find both the largest and smallest number in a list of integer values
5. Write a program to demonstrate change in parameter values while swapping two integer variables using Call by Value & Call by Address
6. Write a program to perform various string operations.
7. Write a program to search an element in a given list of values.
8. Write a program that uses functions to add two matrices.
9. Write a program to calculate factorial of given integer value using recursive functions
10. Write a program for multiplication of two N X N matrices.
11. Write a program to sort a given list of integers in ascending order.

12. Write a program to calculate the salaries of all employees using the Employee (ID, Name, Designation, Basic Pay, DA, HRA, Gross Salary, Deduction, Net Salary) structure.
    a. DA is 30 % of Basic Pay
    b. HRA is 15% of Basic Pay
    c. Deduction is 10% of (Basic Pay + DA)
    d. Gross Salary = Basic Pay + DA+ HRA
    e. Net Salary = Gross Salary - Deduction
13. Write a program to read / write the data from / to a file.
14. Write a program to reverse the contents of a file and store in another file.
15. Write a program to create Book (ISBN,Title, Author, Price, Pages, Publisher)structure and store book details in a file and perform the following operations
    a. Add book details
    b. Search a book details for a given ISBN and display book details, if available
    c. Update a book details using ISBN
    d. Delete book details for a given ISBN and display list of remaining Books

Semester-II

C3: Python Programming and Data Structures

Course Objectives

1. To introduce the fundamentals of Python programming, including environment setup, syntax, and core concepts.
2. To develop problem-solving skills using control flow, functions, and modules.
3. To provide knowledge of Python data structures, file handling, and exception handling for effective programming.
4. To impart object-oriented programming concepts and GUI development skills for building applications.

Course Outcomes (COs)

After successful completion of the course, students will be able to:

1. Explain the basic features, syntax, data types, and operators of Python programming.
2. Apply control flow constructs, functions, and modules to develop structured Python programs.
3. Demonstrate the use of sequences, sets, and dictionaries for effective data handling and manipulation.
4. Implement file handling techniques and apply exception handling mechanisms for robust applications.
5. Develop object-oriented and GUI-based applications using Python.

Unit 1. Basics of Python Programming:

Introduction to Python, Features of Python, Programming Modes - Interactive Mode & Script Mode, Identifiers, Naming Conventions, Keywords (Reserved Words), Built-in Data Types, Literals - Integer, Float, Complex, Boolean, String, Variables, Operators, Expressions, Assignment Statements, Input/Output Statements, Python Syntax (Lines, Comments, Indentation)

Operators & Operands, Classification of Operators - Arithmetic Operators, Relational Operators, Logical Operators, Bitwise Operators, Assignment, Augmented Assignment, Identity Operators, Expressions & Precedence Rules

Unit 2. Control Flow, Functions & Modules:

Control Flow - if Statement, if-else, if-elif-else. Iterative Statements – while, for, Nested Loops, Loop Control Statements – break, continue, pass; else with loops

Need for Functions, Defining & Invoking User-defined Functions, Return Statement, Function Input/Output Cases, Scope of Variables - Local, Global, Nested Functions, Function Arguments - Required, Positional, Default, Variable-length, main() Function, Documentation Strings, Recursive Functions, Anonymous Functions (Lambda), Library Functions

Modules - Import, from..import, Creating & Using Modules, Namespaces

Unit 3. Sequence, Set, Mapping Types:

Strings- Representation, Indexing, Slicing, Immutability, String Operators, Traversal, Accumulation, Formatting & Methods

Lists - Overview, Indexing, Slicing, Methods, Mutability, List Operations - Add, Update, Delete, Search, Copy, Traverse, Comprehension

Tuples - Operations, Immutability, Tuple Assignment, Arrays & Operations

Sets - Overview, Methods, Mathematical Operations, Frozenset, Comprehension

Dictionaries - Overview, Methods, Operations, Traversal, Comparison

Unit 4. File Handling, Exception Handling & Object Oriented Programming:

File Handling - Types, Paths, Basic Operations on Files - Open/Close, Read/Write, CSV Files, OS/Pathlib

Error & Exception Handling - Syntax Errors, Built-in Exceptions, Catching and Handling Exceptions: try-except, raise, User-defined Exceptions, Assertions

OOP Concepts: Classes, Objects, Attributes, Methods, Constructor and Destructors

Encapsulation: Private and Public Members

Inheritance: Single, Multilevel, Multiple, Method Overriding

Unit 5: Abstract Data Structures and GUI Programming

Abstract Data Structures (ADTs): Concepts and Importance

Linked List: Definition, Types- Singly, Doubly, Circular; Node Structure, Insertion, Deletion, Traversal (Single Linked list implementation only)

Stacks: LIFO Principle, Implementation using List, Applications

Queues: FIFO Principle, Implementation using List, Priority Queues

GUI Programming with Tkinter: Widgets (Label, Button, Entry, Menu, Listbox, Canvas etc.), Event Handling, Building Simple GUI Apps

Textbooks:

1. Python Programming-An Object Oriented approach, Anita Goel, Universities Press
2. Python Programming using Problem Solving Approach Reema Thareja Oxford University Press 2020

3. Exploring Python, Budd T A, McGraw-Hill Education, 1st Edition, 2011.

Reference Book:

1. Python: The Complete Reference, Martin C. Brown, Mc Graw-Hill, 2018
2. Fundamentals of Python, Kenneth A. Lambert. (2019), First Programs,2nd Edition, CENGAGE Publication.

Activities:

Outcome: Explain the basic features, syntax, data types, and operators of Python programming.

Activity: Conduct a "Python Basics Lab" where students write small programs to demonstrate literals, variables, data types, and operators (e.g., swapping numbers, simple calculator).

Evaluation Method:

- Lab performance checklist (execution of 3 mini tasks)
- Short quiz with multiple-choice and fill-in-the-blanks on syntax, data types, and operators

Outcome: Apply control flow constructs, functions, and modules to develop structured Python programs.

Activity: Group activity - "Python Problem Solving Challenge": Students solve real-life problems (e.g., finding prime numbers, grade calculator, menu-driven calculator) using control structures, functions, and importing standard modules.

Evaluation Method:

- Code submission with proper use of functions/modules (20%)
- Viva-voce to explain logic and flow of control (40%)
- Unit test with scenario-based programming questions (40%)

Outcome: Demonstrate the use of sequences, sets, and dictionaries for effective data handling and manipulation.

Activity: Hands-on mini project – "Student Data Manager": Students create a program using lists, tuples, sets, and dictionaries to store and manipulate student records (e.g., marks, courses, hobbies).

Evaluation Method:

- Practical demo of program with at least 5 data operations (add, search, delete, update, traverse)
- Evaluation rubric for correctness, efficiency, and use of appropriate data structure

Outcome: Implement file handling techniques and apply exception handling mechanisms for robust applications.

Activity: Individual assignment - "File-Based Address Book": Students create a program to store, update, and retrieve data from files, with exception handling for invalid inputs or missing files.

Evaluation Method:

- Assessment of program correctness (file read/write, append, delete, exception handling)
- Short quiz with error-tracing and debugging questions (given code with errors, students identify and correct)

Outcome: Develop object-oriented and GUI-based applications using Python.

Activity: Mini Project – "Student Information System with GUI": Students design a simple Tkinter-based application with classes/objects for handling student data, including basic GUI widgets (Entry, Button, Listbox).

Evaluation Method:

- Project demo and presentation (50%)
- Rubric-based evaluation for OOP concepts (classes, inheritance, encapsulation) and GUI design (widgets, event handling) (30%)
- Peer review/feedback on usability (20%)

C3P: Python Programming and Data Structures Lab:

1. Basic Python Programs:
    a. Write a program to display basic details (name, roll number, department) using print() and demonstrate different literal types (int, float, string, boolean, complex).
    b. Write a program to perform arithmetic, relational, logical, bitwise, and assignment operations on given inputs.
2. Control Flow Practice
    a. Write a program to find the largest of three numbers using if-elif-else.
    b. Write a program to check whether a number is prime or not using loops.
    c. Write a program to illustrate the use of loop control statements (break, continue, pass).

3. Functions and Recursion
    a. Write a program to define a function to calculate factorial of a number (using recursion).
    b. Write a program to demonstrate different types of function arguments (default, positional, keyword, variable-length).
4. Write a program to illustrate string slicing, concatenation, repetition, and built-in methods.
5. Write a program to create a list of numbers, perform insertion, deletion, searching, sorting, and list comprehension.
6. Write a program to demonstrate tuple packing, unpacking, and immutability.
7. Write a program to implement set operations (union, intersection, difference, subset, superset).
8. Write a program to create a dictionary of student roll numbers and marks, and perform add, update, delete, and traversal operations.
9. Write a program to read and display count of vowels, consonants, digits, and spaces of a text file.
10. Write a program to copy the contents of one file into another file.
11. Write a program to read and process student marks from a CSV file (calculate average, highest, lowest).
12. Write a program to demonstrate exception handling using try-except-finally.
13. Write a program to create a class Student with attributes and methods to display details.
14. Write a program to demonstrate single and multilevel inheritance.
15. Implement stack (LIFO) and queue (FIFO) using lists and linked lists.
16. Implement singly linked lists: node creation, insertion, deletion, traversal.
17. Write a Tkinter program with Label, Entry, and Button widgets to take user input and display it.
18. Write a Tkinter program to create a simple calculator application.

C4: Statistical Foundations for Data Science

Course Objectives

1. To introduce the fundamental concepts of probability and statistics for quantifying and analyzing uncertainty in real-world problems.

2. To develop an understanding of random variables, expectations, and common probability distributions (discrete and continuous).

3. To build the ability to summarize and describe data using measures of central tendency, dispersion, correlation, and visualization techniques.

4. To equip students with statistical tools for modeling relationships using correlation and regression analysis.

5. To provide knowledge of estimation and hypothesis testing for making valid inferences from sample data about populations.

Course Outcomes

At the end of the course, students will be able to:

1. **Apply** the basic rules of probability, conditional probability, and Bayes' theorem to solve problems involving uncertainty.

2. **Compute and interpret** descriptive statistics (mean, median, mode, variance, standard deviation, correlation, covariance) and represent data effectively using histograms, bar charts, and scatter plots.

3. **Analyze** random variables and probability distributions (Binomial, Poisson, Normal, Exponential, etc.) to model real-life situations.

4. **Perform** correlation and regression analysis to identify and interpret relationships between variables.

5. **Conduct** statistical inference through confidence intervals and hypothesis testing (z-test, t-test, chi-square, F-test) for decision-making.

Unit 1: Fundamentals of Probability & Basic Statistics

Probability: Concept of Uncertainty, Axioms and rules of probability, Conditional probability and independence, Law of total probability and Bayes' theorem

Measures of central tendency: Mean, Median, Mode

Measures of dispersion: range, interquartile range, variance, standard deviation

Introduction to correlation and covariance

Data representation: histograms, bar charts, scatter plots

Unit 2: Random Variables, Expectation, and Variance

Random variables: definition, types (discrete & continuous), and properties, Probability mass function (PMF) and probability density function (PDF), Cumulative distribution function (CDF), Mathematical expectation (mean), variance, and standard deviation, Moments and moment-generating functions

Unit 3: Probability Distributions

Discrete distributions: Binomial, Poisson, Geometric, Negative Binomial distributions - definitions, properties, and examples

Continuous distributions: Uniform, Normal (Gaussian), Exponential, Gamma distributions -definitions, properties, and applications

Joint, marginal, and conditional distributions, Introduction to Central Limit Theorem

Unit 4: Correlation and Regression

Bivariate data and scatter plots

Correlation: Pearson and Spearman coefficients, interpretation

Simple linear regression: model, estimation, properties, and analysis of variance

Multiple linear regression basics (conceptual understanding)

Residuals and goodness of fit

Unit 5: Statistical Inference, Estimation, and Hypothesis Testing

Population and sample, parameters and statistics, Sampling distributions, Point and interval estimation (confidence intervals), Tests of significance: z-test, t-test, chi-square test, and F-test, p-values and errors (Type I & II), Power of a statistical test

Textbooks:

1. Probability and Statistics for Engineers and Scientists, Ronald E. Walpole, Wiley.
2. Sheldon M. Ross, Introduction to Probability and Statistics for Engineers and Scientists
3. Douglas C. Montgomery & George C. Runger, Applied Statistics and Probability for Engineers

Reference Books:

1. D.C. Agarwal, Statistics for Data Science and AI
2. Larry J. Stephens, Excel Data Analysis: Your visual blueprint for analyzing data, statistics, and AI

Activities:

Outcome: Apply probability rules, conditional probability, and Bayes' theorem

Activity: Classroom **Quiz** (MCQs & short problems on probability, conditional probability, Bayes).

Evaluation Method: Individual quiz marks (accuracy of solutions).

Outcome: Compute and interpret descriptive statistics & visualize data

Activity: Poster Presentation – students prepare posters with a dataset summary (mean, median, variance, histograms, scatter plot).

Evaluation Method: Rubric-based evaluation (clarity, correctness, creativity, interpretation).

Outcome: Analyze random variables and probability distributions

Activity: Seminar/Presentation – each student (or group) explains one distribution (Binomial, Poisson, Normal, Exponential, etc.) with real-life examples and graphs.

Evaluation Method: Seminar grading (content accuracy, explanation, use of examples, presentation skills).

Outcome: Perform correlation and regression analysis

Activity: Casé Study Assignment – students are given real-world data (e.g., sales vs advertising, study hours vs marks) to compute correlation/regression and interpret.

Evaluation Method: Submission of case study report + viva (correctness of analysis and interpretation).

Outcome: Conduct statistical inference (estimation & hypothesis testing)

Activity: Group Discussion/Debate – teams test a given hypothesis (e.g., "Male and Female students score equally") using sample data and statistical tests, then defend their conclusion.

Evaluation Method: Marks based on correct application of test, interpretation of p-value, and clarity in argument.

C5P: Statistical Foundations for Data Science Lab

Advanced Spreadsheets/Excel Lab/PSPP Open Source

1. Construct a contingency table from sales data and compute conditional probabilities. Verify independence of variables.
2. Apply Bayes' theorem on medical test data to compute the probability of disease given a positive result.
3. Calculate measures of central tendency (mean, median, mode) for student marks dataset.
4. Compute measures of dispersion (range, variance, standard deviation, IQR) for the same dataset and interpret variability.
5. Create a histogram of student marks and comment on the shape of the distribution.
6. Prepare bar charts of categorical data (e.g., Gender vs Section) and interpret group comparisons.
7. Generate scatter plots between Hours Studied and Exam Score, compute correlation and covariance, and interpret the relationship.
8. Random Variable Simulation: Simulate and visualize discrete/continuous random variables using Excel functions and Data Analysis Tool pack.
9. Expectation & Variance Calculation: Use Excel formulas to compute expected value, variance, and standard deviation from given datasets.

10. Modeling Discrete Probability Distributions: Generate and plot Binomial and Poisson distributions; analyze probabilities and mean/variance.
11. Modeling Continuous Distributions: Simulate Normal and Exponential distributions; use NORM.DIST, NORM.INV, EXPON.DIST functions.
12. Correlation Analysis: Calculate Pearson/Spearman correlation coefficients; visualize with scatter plots and trendlines.
13. Linear Regression in Excel: Fit a linear regression model, interpret coefficients, predict new values; use REGRESSION tool.
14. Statistical Inference & Estimation: Create confidence intervals using Excel formulas; visualize sampling distributions
15. Hypothesis Testing: Perform z-test, t-test, chi-square tests in Excel; interpret p-values and results.

Semester-III

C5: Database Management Systems

Course Objectives:

1. To understand the fundamentals of data, information, and the evolution from file-based systems to modern database management systems.
2. To develop the ability to design conceptual data models using Entity-Relationship (ER) and Enhanced ER diagrams.
3. To explore relational model principles, such as keys, integrity constraints and normalization.
4. To perform data definition and manipulation using SQL commands including queries, joins, subqueries, views, and set operations.
5. To apply procedural logic using PL/SQL, incorporating control structures, functions, procedures, and database triggers.

Course Outcomes:

At the end of the course, students will be able to:

1. **Describe** the fundamentals of data, database systems, and the differences between file-based and database approaches. **Compare and classify** various DBMS architectures, data models, and their components, including the three-schema architecture.
2. **Design** conceptual data models using Entity-Relationship and Enhanced ER diagrams, applying generalization, specialization, and constraints.
3. **Apply** relational model concepts, including CODD rules and normalization techniques.
4. **Construct and execute** SQL queries for data definition, manipulation, aggregation, joining, and subqueries, including views and set operations.
5. **Develop** PL/SQL programs incorporating control structures, procedures, and functions to manage database behavior effectively.

Unit 1. Overview of Database Management System:

Introduction to data, information, database, database management systems, file-based system, Drawbacks of file-Based System, database approach, Classification of Database Management Systems, advantages of database approach, Various Data Models, Components of Database Management System, three schema architecture of data base, costs and risks of database approach.

Unit 2. Entity-Relationship Model:

Introduction, the building blocks of an entity relationship diagram, classification of entity sets, attribute classification, relationship degree, relationship classification, reducing ER diagram to tables, enhanced entity-relationship model (EER model), generalization and specialization, IS A relationship and attribute inheritance, multiple inheritance, constraints on specialization and generalization, advantages of ER modeling.

Unit 3. Relational Model:

Introduction, CODD Rules, relational data model, concept of key, relational integrity, relational algebra, relational algebra operations, advantages of relational algebra, limitations of relational algebra, Functional dependencies and normal forms.

Unit 4. Structured Query Language:

Introduction, Commands in SQL, Data Types in SQL, Data Definition Language, Selection Operation, Projection Operation, Aggregate functions, Data Manipulation Language, Table Modification Commands, Join Operation, Set Operations, View, Sub Query.

Unit 5. PL/SQL:

Introduction, Shortcomings of SQL, Structure of PL/SQL, PL/SQL Language Elements,Data Types, Operators Precedence, Control Structures, Steps to Create a PL/SQL, Program, Iterative Control, Procedures, Functions.

Textbooks:

1. Database System Concepts, Avi Silberschatz, Henry F. Korth,S. Sudarshan, Seventh Edition, McGraw-Hill
2. Database Management Systems by Raghu Ramakrishnan, McGrawhill

Reference Books:

1. Fundamentals of Database Systems, Elmasri Navathe Pearson Education
2. An Introduction to Database systems, C.J. Date, A.Kannan, S.Swami Nadhan, Pearson

Activities:

Outcome: Describe the fundamentals of data, database systems, and the differences between file-based and database approaches. Compare and classify various DBMS architectures, data models, and their components, including the three-schema architecture.

Activity: Create a comparative presentation or infographic illustrating:

- File-based vs. DBMS approaches
- Types of DBMS architectures (1-tier, 2-tier, 3-tier)
- Data models and the three-schema architecture

Evaluation Method: Rubric-based assessment of the presentation covering clarity, accuracy, and depth of comparison. Include a short quiz to test conceptual understanding.

Outcome: Design conceptual data models using Entity-Relationship and Enhanced ER diagrams, applying generalization, specialization, and constraints.

Activity: Model a university or hospital database using ER and Enhanced ER diagrams that shows:

- Entity sets, relationships
- Generalization/specialization
- Participation and cardinality constraints

Evaluation Method: Diagram submission with peer review and instructor feedback. Use a checklist to assess completeness, correctness, and notation usage.

Outcome: Apply relational model concepts, including CODD rules, and normalization techniques.

Activity: Normalize a given Database upto 3NF.

Evaluation Method: Written assignment graded on:

- Correctness of normalization steps
- Short-answer questions on CODD rules

Outcome: Construct and execute SQL queries for data definition, manipulation, aggregation, joining, and subqueries, including views and set operations.

Activity: Implement a mini-project (e.g., Library or Inventory DB) using SQL. Include:

- Table creation (DDL)
- Data manipulation (DML)
- Aggregation, joins, subqueries, views, and set operations

Evaluation Method: Lab-based practical test with query execution and output validation. Include a viva to explain logic and optimization.

Outcome: Develop PL/SQL programs incorporating control structures, procedures and functions to manage database behaviour effectively.

Activity: Build a PL/SQL-based payroll or student grading system using:

- Procedures and functions
- Control structures (IF, LOOP)
- Triggers for automated updates

Evaluation Method: Code review and demonstration. Evaluate based on:

- Syntax correctness
- Logical flow

C5P: Database Management Systems Lab

Experiment 1 : Database: Inventory Management

Table 1: Products

Structure:

Column Name	Data Type	Constraints
product_id	INT	PRIMARY KEY
product_name	VARCHAR(50)	NOT NULL
price	DECIMAL(10,2)	CHECK(price > 0)
stock_qty	INT	CHECK(stock_qty >= 0)

Sample Data:

product_id	product_name	price	stock_qty
1	Pen	10.00	100
2	Notebook	50.00	200
3	Stapler	120.00	50
4	Marker	25.00	80
5	File Folder	60.00	150

Table 2: Suppliers

Structure:

Column Name	Data Type	Constraints
supplier_id	INT	PRIMARY KEY
supplier_name	VARCHAR(50)	NOT NULL
contact_no	VARCHAR(20)	UNIQUE
product_id	INT	FOREIGN KEY REFERENCES Products(product_id)

Sample Data:

supplier_id	supplier_name	contact_no	product_id
101	StationeryMart	9876543210	1

102	PaperWorld	9876500000	2
103	OfficeSupplies	9876512345	3
104	MarkerHub	9876522222	4
105	FileDepot	9876533333	5

Section A: DDL (Data Definition Language)

1. Create a database called InventoryDB.
2. Create a table Products and table Suppliers with the specified columns and constraints:

Section B: DML (Data Manipulation Language)

4. Insert at least 5 rows into the Products table.
5. Insert at least 5 rows into the Suppliers table.
6. Update the stock quantity of product 'Pen' to 120.
7. Delete a supplier with a specific supplier_id.
8. Write a query to rename 'Notebook' to 'NoteBook A4'

Section C: DQL (SELECT Queries)

9. Display all records from the Products table.
10. Display only product_name and price of all products.
11. List all products that have a stock quantity less than 100.
12. Show all products between 20 and 100 price range.
13. Find all suppliers whose contact number starts with '98765'.
14. Find the average price of products.
15. Display the total number of products in the inventory.
16. Show the maximum and minimum stock quantities.
17. Count how many suppliers supply each product.
18. Show all products where price > 50 AND stock_qty > 100.
19. Show all products where price < 20 OR stock_qty < 80.
20. Display suppliers whose supplier_name contains the word 'Mart'
21. List all suppliers along with the product they supply (use INNER JOIN).
22. Display suppliers whose name starts with 'S'.
23. Find products whose name has exactly 5 characters
24. Find suppliers who supply products costing more than 100.

Experiment 2 : ONLINE BOOKSTORE DB

An online book store wants to implement a BOOKSTORE DB for managing their online transactions by using the following tables.

Authors Table

Column Name	Data Type	Constraints
author_id	INTEGER	PRIMARY KEY

first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
nationality	VARCHAR	NULL allowed

Books Table

Column Name	Data Type	Constraints
book_id	INTEGER	PRIMARY KEY
Title	VARCHAR	NOT NULL
author_id	INTEGER	FOREIGN KEY REFERENCES Authors
publication_year	INTEGER	
Price	DECIMAL	

Customers Table

Column Name	Data Type	Constraints
customer_id	INTEGER	PRIMARY KEY
first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
Email	VARCHAR	UNIQUE, NOT NULL
Address	VARCHAR	NOT NULL

Orders Table

Column Name	Data Type	Constraints
order_id	INTEGER	PRIMARY KEY
customer_id	INTEGER	FOREIGN KEY REFERENCES Customers
book_id	INTEGER	FOREIGN KEY REFERENCES Books
order_date	DATE	NOT NULL
quantity	INTEGER	NOT NULL

SAMPLE DATA SET for BOOKSTORE DB

Authors Table

author_id	first_name	last_name	nationality
1	Jane	Austen	British
2	George	Orwell	British
3	Gabriel	Garcia Marquez	Colombian

4	Toni	Morrison	American
5	Mark	Twain	American
6	Harper	Lee	American
7	Fyodor	Dostoevsky	Russian

Books Table

book_id	Title	author_id	publication_year	price
101	Pride and Prejudice	1	1813	12.99
102	1984	2	1949	9.50
103	One Hundred Years of Solitude	3	1967	15.00
104	Beloved	4	1987	11.25
105	Animal Farm	2	1945	8.75
106	Adventures of Huckleberry Finn	5	1884	10.50
107	To Kill a Mockingbird	6	1960	14.00

Customers Table

customer_id	first_name	last_name	Email	address
201	Alice	Smith	alice.s@example.com	12 Oak St, London
202	Bob	Johnson	bob.j@example.com	45 Pine Ave, Oxford
203	Charlie	Brown	charlie.b@example.com	78 Maple Rd, Bristol
204	Diana	Prince	diana.p@example.com	34 Queen St, York
205	Edward	Norton	edward.n@example.com	22 River Ln, Leeds
206	Fiona	Hall	fiona.h@example.com	56 Lake Dr, Bath
207	Greg	Miller	greg.m@example.com	89 Park Ave, Glasgow

Orders Table

order_id	customer_id	book_id	order_date	Quantity
301	201	101	2025-07-20	1
302	202	102	2025-07-21	2
303	201	105	2025-07-22	1
304	203	103	2025-07-23	1
305	204	106	2025-07-24	1
306	205	107	2025-07-25	3
307	206	104	2025-07-26	2

Section A: DDL (Schema Design & Constraints)

1. Write SQL statements to create all 4 tables (Authors, Books, Customers, Orders) with:
    - Primary Keys
    - Foreign Keys
    - Appropriate data types
    - NOT NULL constraints where necessary.
2. Alter the Books table to add a constraint that price must be greater than 0.
3. Add a new column phone_number to the Customers table (VARCHAR(15)) and ensure it is unique.
4. Drop the phone_number column from the Customers table.

Section B: DML (Data Manipulation)

5. Insert at least 7 records for each table (use sample dataset above).
6. Update the price of the book titled *Animal Farm* by increasing it by 10%.
7. Delete all orders made before 2025-07-21.
8. Change the nationality of Gabriel Garcia Marquez to "Latino-American".

Section C: SELECT Queries (Data Querying)

9. List all books published between 1900 and 2000.
10. Find all customers whose email contains "example.com".
11. Retrieve books whose price is between 10 and 15 and published before 1950.
12. Show authors who are either 'British' or 'American'.
13. Find books that have a price less than 10 or are published after 1980.
14. Display all orders placed after 2025-07-22.
15. List all books written by author with author_id = 2.
16. Find customers whose last name starts with B.
17. Show all books with a price NOT between 9 and 13.
18. Display books whose publication_year is in (1813, 1945, 1987).
19. Find authors whose nationality is NOT 'British'.
20. List customers whose address contains the word Park.
21. Show all books sorted by price in descending order.
22. List authors in alphabetical order by last_name.
23. Display orders sorted by order_date (latest first).

Use of Date Functions

24. Show all orders placed in July 2025.
25. Show all orders with an estimated delivery date (5 days after order date).
26. Show customers who placed an order on a weekend.
27. Calculate how many days have passed since the last order was placed.

Aggregate Functions (COUNT, SUM, AVG, MIN, MAX)

28. Count the total number of books in the database.
29. Find the average price of all books.
30. Show the highest-priced book.

31. Count how many orders each customer has placed.
32. Calculate the total sales (price × quantity) for each customer.

GROUP BY and HAVING

33. Count how many books are written by each author.
34. Group orders by customer_id and display total quantity ordered.
35. Show customers who have ordered more than 2 books in total (use HAVING).
36. Find the total number of books sold per author (GROUP BY author).

Experiment 3: EMPLOYEE DB

An enterprise wants to automate its employee management process by implementing an Employee Database.The goal is to replace manual record-keeping with a centralized system that stores employee, department, and project details. Use the following table structures and data set to implement Employee DB.

EmployeeDB - Table Structures

1. Departments Table

Column	**Type**	**Constraints**
dept_id	INT	PRIMARY KEY
dept_name	VARCHAR	UNIQUE, NOT NULL
location	VARCHAR	NOT NULL

2. Employees Table

Column	Type	Constraints
emp_id	INT	PRIMARY KEY
first_name	VARCHAR	NOT NULL
last_name	VARCHAR	NOT NULL
email	VARCHAR	UNIQUE, NOT NULL
phone	VARCHAR	CHECK (phone LIKE '--____')
hire_date	DATE	NOT NULL
job_title	VARCHAR	NOT NULL
salary	DECIMAL	CHECK (salary > 0)
dept_id	INT	FOREIGN KEY REFERENCES Departments(dept_id)
manager_id	INT	FOREIGN KEY REFERENCES Employees(emp_id) (self-referential)

3. Projects Table

Column	**Type**	**Constraints**
project_id	INT	PRIMARY KEY

project_name	VARCHAR	NOT NULL
start_date	DATE	NOT NULL
end_date	DATE	NULL
dept_id	INT	FOREIGN KEY REFERENCES Departments(dept_id)

4. Employee_Project Table (Many-to-Many)

Column	Type	Constraints
emp_id	INT	FOREIGN KEY REFERENCES Employees(emp_id), PRIMARY KEY(emp_id, project_id)
project_id	INT	FOREIGN KEY REFERENCES Projects(project_id)
hours_allocated	INT	CHECK (hours_allocated > 0)

Sample Data Set

Departments Table

dept_id	dept_name	Location
1	HR	New York
2	IT	San Francisco
3	Finance	Chicago
4	Marketing	Boston
5	Operations	Seattle
6	Legal	Washington D.C.
7	Sales	Dallas
8	R&D	Austin
9	Procurement	Denver
10	Customer Care	Miami

2. Employees Table

emp_id	first_name	last_name	Email	phone	hire_date	job_title	salary	dept_id	manage d
101	Alice	Johnson	alice.j@corp.com	123-456-7890	2020-03-15	HR Manager	75000	1	NULL
102	Bob	Smith	bob.s@corp.com	234-567-8901	2019-05-20	IT Analyst	65000	2	104

103	Charlie	Brown	charlie.b@corp.com	345-678-9012	2021-01-10	Finance Executive	58000	3	106
104	Diana	Prince	diana.p@corp.com	456-789-0123	2018-07-12	IT Manager	90000	2	NULL
105	Ethan	Hunt	ethan.h@corp.com	567-890-1234	2022-02-25	Marketing Lead	62000	4	NULL
106	Fiona	Hall	fiona.h@corp.com	678-901-2345	2017-11-01	Finance Manager	85000	3	NULL
107	Greg	Miles	greg.m@corp.com	789-012-3456	2023-04-15	IT Support	45000	2	104
108	Hannah	White	hannah.w@corp.com	890-123-4567	2021-09-05	HR Executive	50000	1	101
109	Ian	Scott	ian.s@corp.com	901-234-5678	2020-11-20	Operations Analyst	56000	5	NULL
110	Julia	Adams	julia.a@corp.com	012-345-6789	2019-12-18	Legal Advisor	70000	6	NULL

3. Projects Table

project_id	project_name	start_date	end_date	dept_id
201	Payroll System	2023-01-01	NULL	3
202	Website Upgrade	2023-02-10	NULL	2
203	Recruitment Drive	2023-03-05	NULL	1
204	Ad Campaign	2023-05-20	NULL	4
205	New CRM Tool	2023-04-15	NULL	7
206	Compliance Portal	2023-06-10	NULL	6
207	Inventory System	2023-07-01	NULL	5
208	AI Research	2023-08-05	NULL	8
209	Customer Feedback	2023-09-10	NULL	10
210	Procurement System	2023-10-01	NULL	9

4. Employee_Project Table

emp_id	project_id	hours_allocated
102	202	120
104	202	80
103	201	100

106	201	150
101	203	50
105	204	70
107	202	60
109	207	90
110	206	110
108	203	40

Section A: DDL (Schema Creation & Modification)

1. Write SQL statements to create the above tables with the specified constraints
2. Alter the Employees table to add a column bonus DECIMAL(8,2) with default value0.
3. Drop the column bonus from Employees.

Section B: DML (Insert, Update, Delete)

4. Insert at least 10 rows into Departments, Employees, Projects, and Employee_Project.(use the above data set)
5. Try inserting an employee with a negative salary (should fail due to CHECK constraint).
6. Update the salary of the employee with emp_id = 103 by 15%.
7. Delete an employee record who has resigned (choose any emp_id).
8. Increase all employees' salaries in the IT department by 5%.
9. Change the department of an employee to "Research".(should fail due to FK constraint)

Section C: DQL (Select Queries)

10. List all employees and their details.
11. Show all employees in the "HR" department.
12. Find employees with salaries between 50,000 and 80,000.
13. Retrieve employees hired after 2020.
14. Show employees who are in either the IT or Finance department.
15. Find employees whose email ends with "@corp.com".
16. List all employees with salary > 60,000 AND located in "New York".
17. Display employees in descending order of salary.
18. Count the number of employees in each department.
19. Show the average salary of employees department-wise.
20. Display departments where the average salary is greater than 70,000.
21. Find the number of employees in each project.
22. Display departments with more than 3 employees.
23. Show the sum of all salaries department-wise.
24. List all distinct department IDs from the Employees table.

25. Show employee names with the year they were hired.
26. Show employees grouped by the year of hire.
27. List employees hired in the last 90 days.
28. List the no of years of experience of all the employees

Section D: Joins

29. List all employees with their department names (INNER JOIN).
30. Display all departments along with employees, including those departments without employees (LEFT JOIN).
31. Show employees and the projects they are working on (JOIN 3 tables: Employees, Employee_Project, Projects).
32. List projects along with total hours allocated by employees.
33. Write a query to find employees who are working on more than one project.
34. Show all projects handled by the 'Finance' department.

Section E: PL/SQL Programming

1. Write a procedure GetEmpInfo that takes emp_id as input and displays name, salary, and department.
2. Write a PL/SQL block that checks if an employee's salary is above 50,000.
   If yes, print "High Salary" ;Otherwise print "Standard Salary".
3. Write a PL/SQL program to display the top 10 rows in the Emp table based on their job and salary
4. Write a stored procedure GiveBonus that takes department ID and a designation as input, along with a bonus amount, and updates the salary of all employees in that department who have the specified designation by adding the bonus amount to their current salary.
5. Create a trigger to prevent inserting employees with a salary less than 30,000.
6. Create a trigger to avoid any transactions(insert, update, delete) on the EMP table on Saturday & Sunday.

C6: Data Science with R:

Course Objectives

1. Introduce the data science process, lifecycle, and applications in real-world domains.
2. Build proficiency in R programming for data manipulation, exploration, and visualization.
3. Train students in handling structured, unstructured, and time-based data effectively.
4. Familiarize with basic machine learning and statistical modeling using R.
5. Develop awareness of ethical, interpretability, and responsible use of data science.

Course Outcomes

At the end of the course, students will be able to:

1. Explain the Data Science process and perform EDA (Exploratory Data Analysis).
2. Write R programs using variables, functions, loops, and packages for basic analytics.
3. Perform data wrangling, cleaning, and visualization with R libraries (dplyr, tidyr, ggplot2).
4. Build and evaluate basic machine learning models such as regression and clustering.
5. Apply data science techniques to practical case studies.

Unit 1. Introduction to Data Science Process:

Introduction- Definition - Data Science in various fields - Examples - Impact of Data Science - Data Analytics Life Cycle - Data Science Toolkit - Data Scientist - Data Science Team, Exploratory Data Analysis (EDA), Feature Engineering & Data Transformation

Unit 2. Basics of R Programming:

Introduction to R and RStudio, Data Types, Variables, Operators, Control Structures (if, loops, apply), Functions and Packages, Data Input/Output (CSV, Excel, XML, JSON).

Unit 3. Data Handling & Visualization in R:

Data Frames, Lists, Matrices, Data Wrangling with dplyr and tidyr, Handling Missing Data, Working with Date/Time in R. Visualization with ggplot2: grammar of graphics, aesthetics, geometries, scales.

Faceting and layering techniques,Visualizing categorical and numerical data, Customizing and exporting plots

Unit 4. Applications & Case Studies in Data Science:

Simple Linear Regression, Multiple Regression

Model Evaluation Method: Accuracy, Confusion Matrix, ROC.

K-Means Clustering, Text Mining & Word Clouds, Recommender Systems Basics, Ethical Issues in Data Science

Unit 5. Advanced Topics in Data Science with R :

Introduction to Time Series Analysis in R (ARIMA basics)- Concept of time series (trend, seasonality, noise), Time series objects in R (ts, zoo, xts), Plotting and decomposing time series, Stationarity and differencing, Autocorrelation & Partial Autocorrelation (ACF/PACF), AR, MA, ARIMA model basics, Forecasting using forecast package

Creating interactive visualizations with plotly packages-Converting ggplot2 plots to interactive plots

Animations and sliders in plotly

R Shiny: Building interactive web applications-Introduction to Shiny framework, UI and server functions, Reactive expressions and reactivity in Shiny, Input and output widgets (sliders, dropdowns, text), Layouts and dashboard design

Textbooks

1. An Introduction to Statistical Learning with Applications in R, Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Springer, 2nd Edition, 2021
2. R for Data Science, Hadley Wickham and Garrett Grolemund, O'Reilly Media, 2017.

Reference Books

1. The Art of R Programming, Norman Matloff,, No Starch Press, 2011.
2. Modern Applied Statistics with S, W.N. Venables & B.D. Ripley, Springer, 2002.
3. Introduction to Data Science: Data Analysis and Prediction Algorithms with R, Rafael A. Irizarry, CRC Press, 2020.
4. Data Science from Scratch: First Principles with Python (for conceptual clarity only), Joel Grus,

Activities:

Outcome: Explain the Data Science process and perform EDA (Exploratory Data Analysis).

Activity: Use a real-world dataset (e.g., Titanic or COVID data) to:

- Outline the steps of the Data Science workflow

- Perform EDA using summary statistics and visualizations (histograms, boxplots, scatterplots)

Evaluation Method: Presentation and checklist (10-point scale):

- Clear explanation of workflow stages
- Quality of EDA insights
- Use of appropriate plots and summaries

Outcome: Write R programs using variables, functions, loops, and packages for basic analytics.

Activity: Write an R script that:

- Reads a CSV file
- Uses if, for, and while loops
- Defines and calls custom functions with arguments and return values

Evaluation Method: Code review and execution test to verify (10-point scale):

- Correctness of the syntax and logic
- Functionality of control structures
- Output accuracy and modularity

Outcome: Perform data wrangling, cleaning, and visualization with R libraries (dplyr, tidyr, ggplot2).

Activity: Clean a messy dataset using:

- dplyr for filtering, selecting, and mutating
- tidyr for reshaping and handling missing values
- Time-based operations (e.g., filling gaps, formatting dates)

Evaluation Method: Before-and-after comparison (10 point score):

- Completeness of cleaning steps
- Use of appropriate functions
- Handling of missing/time data

Outcome: Implement basic machine learning models and evaluate performance using appropriate metrics and visual tools.

Activity: Build a simple classification model (e.g., logistic regression or decision tree) using R:

- Train/test split

- Predict outcomes
- Evaluate using confusion matrix, accuracy, precision, recall

Evaluation Method: Model report and demo (10 point scale):

- Correct implementation of model
- Use of evaluation metrics

C6P: Data Science with R Lab:

List of Practicals:

1. Compute Mean, Median, Mode, Variance, and Standard Deviation
2. Visualize Binomial, Normal, and Poisson Distributions
3. Perform t-test and Chi-Square Test in R
4. Calculate Correlation and Build a Simple Linear Regression Model
5. Conduct Exploratory Data Analysis (EDA) on a Real-World Dataset
6. Apply Feature Engineering: Scaling, Normalization, and Encoding
7. Practice R Programming: Variables, Control Structures, and Functions
8. Read and Write Data from CSV, Excel, JSON, and XML Files
9. Use dplyr and tidyr for Data Wrangling Tasks
10. Handle Missing Data and Detect Outliers
11. Work with Dates and Times in R
12. Visualize Data Using ggplot2 (Bar, Scatter, Histogram, Boxplot)
13. Perform K-Means Clustering and Visualize Clusters
14. Evaluate Models Using Confusion Matrix, Accuracy, and ROC Curve
15. Perform Text Mining and Create a Word Cloud
16. Time Series Forecasting with ARIMA on a real dateset (e.g., monthly airline passengers, stock prices, or temperature data).
17. Create interactive bar, line, and scatter plots using plotly. On a real dataset (e.g., COVID-19 cases, sales data, or student marks).
18. Develop a Shiny app that lets users upload a CSV file.

C7: Object Oriented Programming Using JAVA

Course Objectives

1. Introduce core OOP principles and contrast procedural and object-oriented paradigms within the Java ecosystem.
2. Equip learners with foundational and advanced Java syntax including variables, control statements, arrays, strings, and classes.
3. Enable the use of inheritance, polymorphism, interfaces, and exception handling to create maintainable and reusable code.
4. Train students in concurrent programming and stream-based I/O operations including file management and serialization.
5. Empower learners to design, build, and manage GUI programs using Swing components, layout managers, and event handling techniques.

Course Outcomes

At the End of the Course, The Students will be able to:

1. Apply OOP principles such as encapsulation, inheritance, and polymorphism in Java applications.
2. Write, compile, and debug Java code using control statements, arrays, classes, and methods effectively.
3. Construct modular code leveraging interfaces, abstract classes, and package hierarchies.
4. Manage thread lifecycles, synchronization, and I/O streams for file handling and console interaction.
5. Design user interfaces using Swing and handle keyboard/mouse input through event-driven programming.

Unit 1. OOPs Concepts and Java Programming:

Introduction to Object-Oriented concepts, procedural and object-oriented programming paradigm Java programming: An Overview of Java, Java Environment, Data types, Variables, constants, scope and life time of variables, operators, type conversion and casting, Accepting Input from

the Keyboard, Reading Input with Java.util.Scanner Class, Displaying Output, Displaying Formatted Output with String.format(), Control Statements

Unit 2. Arrays and OOP Constructs:

Arrays, Command Line Arguments, Strings-String Class Methods
Classes & Objects: Creating Classes, declaring objects, Methods, parameter passing, static fields and methods, Constructors, and 'this' keyword, overloading methods and access Inheritance: Inheritance hierarchies, super and subclasses, member access rules, 'super' keyword, preventing inheritance: final classes and methods, the object class and its methods; Polymorphism: Dynamic binding, method overriding, abstract classes and methods;

Unit 3. Interfaces, Packages & Exception Handling:

Interfaces VS Abstract classes, defining an interface, implement interfaces, accessing implementations through interface references, extending interface;
Packages: Defining, creating and accessing a package, importing packages.
Exception Handling: Benefits of exception handling, the classification of exceptions, exception hierarchy, checked exceptions and unchecked exceptions, usage of try, catch, throw, throws and finally, rethrowing exceptions, exception specification, built in exceptions, creating own exception sub classes.

Unit 4. Multithreading & Stream based I/O:

Differences between multiple processes and multiple threads, thread states, thread life cycle, creating threads, interrupting threads, thread priorities, synchronizing threads, inter thread communication.
Stream based I/O (java.io) – The Stream classes-Byte streams and Character streams, Reading console Input and Writing Console Output, File class, Reading and writing Files, The Console class, Serialization

Unit 5. GUI Programming with Swing:

Introduction, MVC architecture, components, containers. Understanding Layout Managers - Flow Layout, Border Layout, Grid Layout, Card Layout, GridBag Layout. Event Handling- The

Delegation event model- Events, Event sources, Event Listeners, Event classes, Handling mouse and keyboard events, Adapter classes.

Text Books:

1. Java The complete reference, Herbert Schildt, 9th edition, McGraw Hill.
2. Programming in Java, S. Malhotra, S. Chudhary, 2nd edition, Oxford Univ. Press.

Reference Books:

1. Programming with JAVA - A Primer, E Balaguruswamy, 3rd Edition, McGraw Hill
2. Head First Java: A Brain-Friendly Guide , Katty Sierra, Bert Bates, 2nd Edition, O'Reilly

Activities:

Outcome: Apply OOP principles such as encapsulation, inheritance, and polymorphism in Java applications

Activity: Develop a class hierarchy for a zoo management system using inheritance and polymorphism (e.g., Animal → Mammal → Dog). Implement encapsulation through private fields and public getters/setters.

Evaluation Method: Code review and oral explanation focusing on class relationships, method overriding, and encapsulation practices.

Outcome: Write, compile, and debug Java code using control statements, arrays, classes, and methods effectively

Activity: Create a console-based student grade calculator using loops, conditionals, arrays, and modular methods.

Evaluation Method: Practical test with debugging tasks and output validation across multiple input scenarios.

Outcome: Construct modular code leveraging interfaces, abstract classes, and package hierarchies **Activity**: Design a payment processing system with abstract classes for Payment, interfaces for Taxable, and organize classes into packages (e.g., com.billing, com.tax).

Evaluation Method: Project submission assessed for modularity, interface implementation, abstraction usage, and package structure.

Outcome: Manage thread lifecycles, synchronization, and I/O streams for file handling and console interaction

Activity: Build a multithreaded logger that reads input from the console and writes to a file using synchronized threads and buffered streams.

Evaluation Method: Lab demonstration with thread state tracing and file output verification under concurrent input.

Outcome: Design user interfaces using Swing and handle keyboard/mouse input through event-driven programming

Activity: Create a GUI-based quiz application using Swing components (JFrame, JButton, JTextField) with event listeners for mouse clicks and key presses.

Evaluation Method: Live demo and rubric-based assessment of UI responsiveness, event handling accuracy, and layout design.

C7P: Object Oriented Programming Using JAVA lab

List of Experiments

1. Write a Java program to print Fibonacci series.
2. Write a Java program to calculate multiplication of 2 matrices.
3. Write a Java program for sorting a given list of names in ascending order.
4. Create a class Rectangle. The class has attributes length and width. It should have methods that calculate the perimeter and area of the rectangle. It should have readAttributes() method to read length and width from the user.
5. Write a Java program that implements method overloading.
6. Write a Java program to implement various types of inheritance

i. Single ii. Multi-Level iii. Hierarchical iv. Hybrid

7. Write a java program to implement runtime polymorphism.
8. Write a Java program which accepts withdrawal amount from the user and throws an exception In Sufficient Funds when withdrawal amount is more than available amount.

9. Write a Java program to create three threads and that displays good morning, for every one second, hello for every 2 seconds and welcome for every 3 seconds by using extending Thread class.

10. Write a Java program that creates three threads. The first thread displays OOPS, the second thread displays Through and the third thread displays JAVA by using Runnable interface.

11. Write a Java program that displays the number of characters, lines and words in a text file.

12. Implement a Java program for handling mouse events when the mouse entered, exited, clicked, pressed, released, dragged and moved in the client area.

13. Implement a Java program for handling key events when the key board is pressed, released, typed.

14. Write a Java swing program that reads two numbers from two separate text fields and displays the sum of two numbers in the third text field when the button add is pressed.

15. Write a Java program to design student registration form using Swing Controls. The form must have the following fields and button SAVE

Form Fields are: Name, RNO, Mailid, Gender, Branch, Address.

Semester-IV

C8: Data Mining and Data Warehousing

Course Objectives:

- Provide an understanding of data warehousing concepts, architecture, and OLAP operations for effective storage, modeling, and analysis.
- Develop knowledge of data mining fundamentals, tasks, and preprocessing techniques to prepare data for mining.
- Introduce students to association rule mining algorithms for discovering hidden patterns and relationships in large datasets.
- Enable learners to apply classification techniques (decision trees, Bayesian, nearest neighbor, rule-based) for predictive modeling.

- Equip students with knowledge of clustering paradigms and algorithms (partitioning, hierarchical, density-based, categorical) for data grouping and pattern discovery.

Course Outcomes:

Upon successful completion of the course, the student will be able to:

1. Explain the architecture, schemas, and OLAP operations of data warehousing and distinguish it from traditional database systems.

2. Apply preprocessing techniques (data cleaning, dimensionality reduction, feature selection, transformation, and similarity measures) to prepare raw data for analysis.

3. Implement various association rule mining algorithms (Apriori, Partition, FP-Growth, etc.) to uncover meaningful relationships within large datasets.

4. Build and evaluate classification models using decision tree algorithms (ID3, C4.5, CART), rule-based classifiers, Bayesian classifiers, and nearest-neighbor methods.

5. Analyze and implement clustering techniques such as K-Means, K-Medoid, DBSCAN, BIRCH, and categorical clustering methods (STIRR, ROCK, CACTUS) for grouping and pattern discovery in different types of datasets.

Unit-1: Data Warehousing:

Introduction to Data Ware House, Differences between Database systems and Data Ware House, Data Ware House characteristics, Data Ware House Architecture and its components, Data Modeling, Schema Design, star and snow-Flake Schema, Fact Constellation, Fact Table, OLAP cube, OLAP Operations.

Unit-2: Data Mining:

What is Data Mining? Data Mining: Definitions, KDD vs Data Mining, Data Mining Tasks, Data Preprocessing- Data Cleaning, Missing Data, Dimensionality Reduction, Feature Subset Selection, Discretization and Binarization, Data Transformation; Measures of similarity and Dissimilarity-Basics.

Issues and Challenges in DM, DM Applications- Case Studies

Unit-3: Association Analysis:

Association Rules: What is an Association Rule?, Methods to Discover Association Rules, A Priori Algorithm, Partition Algorithm, Pincer-Search Algorithm, Dynamic Itemset Counting Algorithms, FP-Tree Growth Algorithm, Generalized Association Rule, Association Rules with Item Constraints

Unit-4: Classification:

Introduction to Classification and Predictive Modeling, Decision Tree Concepts:, Basic Terminology (Root, Node, Leaf), Tree Construction Principles (Splitting, Pruning), Information Gain, Gini Index (Basic conceptual), Decision Tree Algorithms: **CART**, Overview of Other Classifiers: Naïve Bayes, k-Nearest Neighbor (k-NN), Rule-Based Classifiers (concepts)

Introduction to Ensemble Learning (Conceptual): What is Ensemble Learning?, Motivation: Improving Accuracy through Combination, Bagging and Random Forest (concepts only), Boosting, Voting Classifiers (concept only), Model Evaluation: Confusion Matrix, Accuracy, Precision, Recall

Unit-5: Clustering Techniques:

Clustering Paradigms, Partitioning Algorithms (K-Means), k-Medoid Algorithms, Hierarchical Clustering: DBSCAN, BIRCH, Categorical Clustering Algorithms: STIRR, ROCK, CACTUS

Textbooks:

1. Data Mining Techniques, Arun K Pujari, 3rd Edition, Universities Press
2. Data Mining: Concepts and Techniques, Jiawei Han, Micheline Kamber, Jian Pei, 3rd Edition, Morgan Kaufmann Publishers

Reference Books:

1. K.P. Soman , Shyam Diwakar, V.Ajay ,2006, Insight into Data Mining Theory and Practice, Prentice Hall of India Pvt. Ltd - New Delhi.

2. Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar, 2nd edition,

Activities:

Outcome: Explain the architecture, schemas, and OLAP operations of data warehousing and distinguish it from traditional database systems.

Activity: Students will design a **conceptual schema** for a data warehouse using **star or snowflake schema** for a given business case (e.g., sales, hospital, or university records). They will also **demonstrate OLAP operations** (roll-up, drill-down, slice, dice) on a sample dataset using a spreadsheet or OLAP tool.

Evaluation Method: Assessment will be based on correctness and completeness of schema design, appropriateness of fact/dimension tables, and accuracy of OLAP operation outputs. A short viva/quiz will be used to test conceptual understanding.

Outcome 2: Apply preprocessing techniques (data cleaning, dimensionality reduction, feature selection, transformation, and similarity measures) to prepare raw data for analysis.

Activity: Students will be given a **real-world noisy dataset** (with missing values, outliers, redundant features). They will perform:

- Data cleaning (handling missing values, outliers)
- Dimensionality reduction (e.g., PCA)
- Feature selection (e.g., filter/wrapper methods)
- Similarity/dissimilarity measure calculations.

Evaluation Method: Evaluation will consider correctness of preprocessing steps, justification of chosen methods, and clarity of intermediate outputs. Students will submit a short report with before/after comparisons.

Outcome 3: Implement various association rule mining algorithms (Apriori, Partition, FP-Growth, etc.) to uncover meaningful relationships within large datasets.

Activity: Using a **transaction dataset** (e.g., market basket data), students will implement Apriori/FP-Growth in **Python (mlxtend library) or Weka**. They will generate frequent itemsets, derive association rules, and interpret them under given **support, confidence, and lift thresholds.**

Evaluation Method: Submissions will be graded on correctness of generated rules, clarity of code/parameters, quality of interpretation, and ability to link discovered rules to practical business insights.

Outcome 4: Build and evaluate classification models using decision tree algorithms (ID3, C4.5, CART), rule-based classifiers, Bayesian classifiers, and nearest-neighbor methods.

Activity: Students will implement at least **two classification algorithms** (e.g., Decision Tree + Naïve Bayes or KNN) on a dataset like **Iris, Titanic, or Student Performance**. They will evaluate models using **confusion matrix, accuracy, precision, recall, and F1-score** and compare results.

Evaluation Method: Assessment will consider correctness of implementation, clarity in performance comparison, and visualization of decision trees/rules. Students must explain why certain models performed better.

Outcome: Analyze and implement clustering techniques such as K-Means, K-Medoid, DBSCAN, BIRCH, and categorical clustering methods (STIRR, ROCK, CACTUS).

Activity: Students will implement at least two clustering algorithms (e.g., K-Means and DBSCAN or BIRCH) on real datasets (e.g., customer segmentation, text data, student groups). They will compare clusters using Silhouette Score, Davies-Bouldin Index, and visualize results using scatter plots/dendrograms.

Evaluation Method: Evaluation will be based on accuracy of clustering implementation, choice of parameters (e.g., k in K-Means, eps in DBSCAN), visualization quality, and clarity in interpretation of clusters.

C8P: Data Mining and Data Warehousing Lab

List of Experiments:

Recommended datasets: **weather.arff, iris.arff, supermarket.arff, vote.arff, contact-lenses.arff**, or custom CSV datasets.

1. Load datasets in WEKA and explore data formats (ARFF/CSV)
2. Perform data cleaning and handle missing values using filters
3. Apply normalization and discretization on numeric attributes
4. Reduce data using attribute selection and PCA
5. Summarize and visualize data using statistical tools and class-wise comparison in WEKA.
6. Generate association rules using the Apriori algorithm
7. Apply multilevel association rule mining using hierarchical attributes
8. Apply K-means clustering and interpret the cluster outputs.
9. Perform hierarchical clustering and visualize results using dendrograms.
10. Apply Expectation-Maximization (EM) clustering and analyze cluster summaries.
11. Build a decision tree classifier using J48 and evaluate its performance.
12. Perform Naive Bayes classification and compare with decision tree results.
13. Apply rule-based classification using PART or JRip algorithms.
14. Compare classifiers using confusion matrix, accuracy, and ROC curves.

C9: Exploratory Data Analysis & Data Visualization

Course Objectives

1. To introduce the concepts, importance, and workflow of exploratory data analysis.
2. To impart practical skills in data preprocessing, cleaning, and manipulation using Python.
3. To equip students with techniques in univariate, bivariate, and multivariate data analysis.
4. To develop proficiency in data visualization and interpretation using standard Python libraries.
5. To enable students to build reports and dashboards to communicate data insights effectively.

Course Outcomes

After completing this course, students will be able to:

1. Explain the importance and process of EDA in the data science pipeline.
2. Perform robust data preprocessing including missing value handling, outlier detection, transformation, and encoding.
3. Manipulate, filter, and reshape datasets using Python libraries like Pandas and NumPy.
4. Conduct data analysis and visualize results using Matplotlib, Seaborn, and Plotly.
5. Design interactive visualizations and dashboards to communicate findings in real-world scenarios.

Unit 1: Introduction to Exploratory Data Analysis (EDA)

Significance and objectives of EDA, Types and scales of data (nominal, ordinal, interval, ratio), Concepts of variability and central tendency, Role of EDA in the data science pipeline, Introduction to data quality and data issues, Differences between EDA, classical, and Bayesian analysis, Basic terminology and concepts

Unit 2: Data Preprocessing and Cleaning

Overview of data preprocessing, Handling missing types of missingness (MCAR, MAR, MNAR), Techniques for missing data imputation (mean, median, mode, forward/backward fill), Detection and treatment of duplicates, Outlier detection methods (Z-score, IQR method), Data transformation: normalization and standardization, Encoding categorical variables: label encoding, one-hot encoding, Data integration and reduction techniques, Importance of preprocessing for machine learning models

Unit 3: Data Manipulation using Python with Pandas and NumPy

NumPy Array Creation: Use `np.array()`, `np.zeros()`, `np.ones()` to create arrays of various dimensions. **Array Manipulation:** Reshape, transpose, flatten arrays for flexible data formats. Element-wise Arithmetic Operations, Statistical Functions on NumPy Arrays

Indexing & Slicing: Access array elements via slicing, masking, and fancy indexing for selective data analysis.

Pandas DataFrames: Series and Data frames, Create from dictionaries, Series, CSV, Excel, and JSON files using Dataframe or reader functions

Data Frame Manipulation: Indexing & Selection - Label-based and Position-based indexing; conditional filtering with boolean masks, Aggregation & Grouping- Group data by columns with `.groupby()`, summarize using `.sum()`, `.mean()`, `.count()`.

Data Reshaping & Pivoting and stack/unstack hierarchical index data.

Merging & Joining Data Frames

Datetime & Categorical Data: Handle and transform temporal data with `pd.to_datetime()`, categorical conversions with `.astype('category')`.

Unit 4: Basic Data Analysis and Visualization

What is visualization, what is it's importance? Types: histograms, bar charts, Boxplots and violin plots for data distribution, Scatter plots and correlation analysis (Pearson, Spearman), Cross-tabulation and contingency tables, Analyzing relationships: correlation vs causation, Visualizing categorical vs numerical data

Unit 5: Advanced Visualization and Reporting

Principles of effective graphical representation, Introduction to visualization libraries: Matplotlib, Seaborn, Plotly, Advanced plots: heatmaps, pair plots, joint plots, Interactive visualizations and dashboards, Geographical data visualization basics, Storytelling with using visualization for communication, Introduction to reporting tools and best practices

Activities

Unit 1: Introduction to EDA

Activity:

Analyze the Titanic dataset to explore data types, central tendency, and variability.

Outcome:

Develop a foundational understanding of data structures and EDA significance.

Evaluation Method:

Lab notebook submission and oral quiz.

Unit 2: Data Preprocessing and Cleaning

Activity:

Clean a health survey dataset: identify/impute missing values, standardize columns, encode categories, remove duplicates/outliers.

Outcome:

Demonstrate proficiency in data cleaning and preprocessing techniques in Python.

Evaluation Method:

Script submission and report on preprocessing steps with before/after summary.

Unit 3: Data Manipulation using Python

Activity:

Manipulate a retail transaction dataset by indexing, filtering, grouping, merging, and handling date/time columns.

Outcome:

Apply multiple Pandas/NumPy techniques for dataset transformation.

Evaluation Method:

Graded Jupyter notebook, peer code review.

Unit 4: Basic Data Analysis and Visualization

Activity:

Explore demographic data with descriptive statistics, create histograms, boxplots, scatter plots, and cross-tabulations.

Outcome:

Visualize and interpret univariate and bivariate data insights.

Evaluation Method:

Visualization notebook and critical analysis write-up.

Unit 5: Advanced Visualization and Reporting

Activity:

- Build a dashboard on sales or air quality data using Matplotlib/Seaborn/Plotly.
- Incorporate interactive and geospatial visualizations.Outcome:
- Develop professional dashboards and communicate insights through visual storytelling.

Evaluation Method:

Final project dashboard evaluation and presentation.

Preferred Textbooks

1. Suresh Kumar Mukhiya, Usman Ahmed, "Hands-On Exploratory Data Analysis with Python", Packt Publishing, 2020.
2. Jake Vander Plas, "Python Data Science Handbook: Essential Tools for Working with Data", O'Reilly, 2017.
3. Catherine Marsh, Jane Elliott, "Exploring Data: An Introduction to Data Analysis for Social Scientists", Wiley, 2008.

References

1. Eric Pimpler, "Data Visualization and Exploration with R", GeoSpatial Training Service, 2017.
2. Claus O. Wilke, "Fundamentals of Data Visualization", O'Reilly, 2019.
3. Matthew O. Ward, Georges Grinstein, Daniel Keim, "Interactive Data Visualization: Foundations, Techniques, and Applications", CRC Press, 2015.

C9P: Exploratory Data Analysis & Data Visualization Lab

Note: *This lab has to be executed using interactive computing environments like Jupyter Notebook or Google Colab.*

1. Importing Data and Basic Exploration
   - Load datasets from CSV/Excel/JSON
   - Use `.head()`, `.info()`, `.describe()`, `.shape` to examine structure and summary statistics
2. Examining Data Types and Missing Values
   - Identify data types for each column
   - Detect and quantify missing values with `.isnull()` and `.sum()`.
3. Handling Missing Data Techniques
   - Impute missing values using mean, median, mode, forward, and backward fill
   - Drop rows/columns with missing data and compare results
4. Dealing With Duplicates and Outliers

- Find and remove duplicate entries
- Detect outliers using Z-score or IQR methods
- Treat or remove outliers and compare distributions

5. Data Transformation and Scaling
   - Normalize, standardize, log-transform columns
   - Encode categorical variables (Label Encoding, One-Hot Encoding)
6. Indexing, Filtering, and Slicing DataFrames
   - Select specific rows/columns using index and Boolean filters
   - Conditional selections and advanced slicing
7. Aggregating, Grouping, and Pivoting Data
   - Use `.groupby()` for aggregate statistics
   - Create pivot tables for categorical summaries
8. Merging and Concatenating Multiple Datasets
   - Join, merge, concatenate DataFrames
   - Handle merging keys and missing arguments
9. Univariate Visualization Techniques
   - Create histograms, boxplots, violin plots for distributions
   - Visualize frequency and central tendency
10. Bivariate and Multivariate Visualization
    - Display scatter plots, correlation matrices, pair plots
    - Visualize relationships and patterns
11. Time Series Data Exploration
    - Parse and index datetime columns
    - Visualize trends and seasonality
12. Interactive Visualizations and Dashboards
    - Build simple interactive graphs with Plotly
    - Design a dashboard showing multiple aspects of a dataset

C10: SOFTWARE ENGINEERING

Course Objectives :

1. Understand the fundamental principles of software engineering, including software development life cycle models and their practical applications.
2. Analyze and document software requirements through feasibility studies and specification techniques.
3. Apply software design principles and development standards for building reliable and maintainable systems.
4. Evaluate software testing methodologies, including design and execution of test cases for various testing levels.
5. Examine software maintenance strategies, types, and metrics to sustain software performance over time.

Course Outcomes

At the End of the Course, The Students will be able to:

1. Compare and contrast software development life cycle models such as Waterfall, Spiral, and Agile, and explain their appropriate use cases.
2. Conduct requirement analysis and distinguish between functional and non-functional requirements to develop a Software Requirements Specification (SRS) document.
3. Utilize design principles like modularity, cohesion, and coupling; implement Data Flow Diagrams (DFDs), structure charts, and follow coding standards and version control practices.
4. Design and perform different types of software tests—white-box, black-box, unit, integration, system—and differentiate between manual and automated testing approaches.
5. Categorize software maintenance types and propose strategies based on software maintenance metrics for effective long-term software sustainability.

Unit 1: Software Engineering Foundations and Requirements Engineering:

Definition of Software engineering, Software life cycle models: Waterfall, prototyping, Evolutionary, Spiral and Agile models. Comparison among development life cycles.

Unit 2: Requirement Analysis and Specification

Feasibility study, Requirements gathering, Functional and Non-functional requirements, Requirements analysis and specification, design of software requirement specification (SRS).

Unit 3: Software Design Principles and Development Practices

Introduction to software design, modularity, cohesion, coupling and layering, functional design, and solution design, use of DFD and Structure chart in software design, UML in software design, user interface design. Software Development Basics, Coding standards, Version Control and Code review techniques.

Unit 4: Software Testing

Fundamentals of testing (verification and validation), White-box, and black-box testing, unit testing, integration testing, system testing, acceptance testing (alpha testing and beta testing), test scenarios and test case design, automation testing and manual testing.

Unit 5: Software Maintenance

Introduction to software maintenance, corrective maintenance, perfective maintenance, adaptive maintenance, preventive maintenance, challenges in software maintenance, metrics related to software maintenance.

Textbooks:

1. Software Engineering: A Practitioner's Approach, Roger Pressman, McGraw Hill, 6th Edition
2. Software Engineering, Sommerville, Adison Wesley, 7th edition.

Reference Books:

1. Fundamentals of Software Engineering, Mall Rajib,PHI, Fifth Edition,
2. Fundamentals of Software Engineering, Hitesh, BPB Publications

Activities

Outcome: Compare and contrast software development life cycle models such as Waterfall, Spiral, and Agile, and explain their appropriate use cases.

Activity: Create a comparison chart in groups showing key features, pros/cons, and use cases of Waterfall, Spiral, and Agile models. Include a real-world example for each.

Evaluation Method: Use a rubric to assess on a 10-point scale to check:

- Accuracy of model descriptions
- Clarity of comparison
- Relevance of examples
- Presentation skills (if shared in class)

Outcome: Conduct requirement analysis and distinguish between functional and non-functional requirements to develop a Software Requirements Specification (SRS) document.

Activity: Analyze a simple system (e.g., online library or food ordering app). Identify 5 functional and 3 non-functional requirements. Draft a mini-SRS document using a template.

Evaluation Method: Checklist-based review on a 10-point scale:

- Correct classification of requirements
- Completeness of SRS sections
- Clarity and formatting
- Use of standard terminology

Outcome: Utilize design principles like modularity, cohesion, and coupling; implement Data Flow Diagrams (DFDs), structure charts, and follow coding standards and version control practices.

Activity: Design a basic login system using DFD (Level 0 and Level 1) and a structure chart. Highlight modules and discuss cohesion and coupling in pairs.

Evaluation Method: Peer review and instructor feedback on a 10-point scale:

- Correct use of DFD symbols
- Logical flow and modularity
- Explanation of cohesion/coupling
- Neatness and labelling

Outcome: Design and perform different types of software tests—white-box, black-box, unit, integration, system—and differentiate between manual and automated testing approaches.

Activity: Write simple test cases for a calculator app (e.g., addition, division by zero). Perform manual unit testing and simulate black-box and white-box testing.

Evaluation Method: Observation and worksheet to assess on a 10-point scale:

- Correct identification of test types
- Clear test case steps and expected output
- Execution and result recording
- Understanding of manual vs automated testing

Outcome: Categorize software maintenance types and propose strategies based on software maintenance metrics for effective long-term software sustainability.

Activity: Categorize sample maintenance tasks (e.g., fixing a bug, adding a feature) into corrective, adaptive, perfective, or preventive. Discuss sustainability strategies in small groups.

Evaluation Method: Group activity score:

- Correct classification of maintenance types
- Participation in discussion
- Suggestions for sustainability (e.g., documentation, modular design)
- Use of basic metrics (e.g., number of bugs fixed)

C10P: SOFTWARE ENGINEERING

Select domain of interest (e.g. College Management System) and identify multi-tier software applications to work on (e.g. Online Fee Collection). Analyze, design and develop this application:

1. Develop an SRS document. Also develop risk management and project plan (Gantt chart).
2. Understanding of System modeling: Data modeling using ER – Diagram-Draw an ER Diagram which also includes generalization, specialization and aggregation of specified problem statements.

3. Understanding of System modeling: Functional modeling: DFD Context and draw it
4. Understanding of System modeling: UML and draw it.
5. Develop a sample calculator program and perform Black-box Testing:
    a. Identify test scenarios without viewing the internal code.
    b. Write test cases for valid and invalid inputs.
    c. Execute the test cases and record outcomes.
6. Develop a sample login authentication page and perform White--box Testing:
    a. Analyze the code for all possible paths, conditions, and loops.
    b. Apply statement coverage and branch coverage techniques.
    c. Test internal functions with known inputs.
7. For the above login authentication page, perform the following:
    a. Simulate the following maintenance activities:
        - Corrective Maintenance: Fix an existing bug (e.g., wrong output, crash, miscalculation).
        - Perfective Maintenance: Add a new user-requested feature (e.g., sorting, filter, or improved UI).
        - Adaptive Maintenance: Modify the code to adapt to a new platform or library version.
        - Preventive Maintenance: Refactor the code to improve readability, performance, or security.
    b. Document each change with:
        - Problem description
        - Type of maintenance
        - Before and after screenshots
        - Change summary

Semester-V

C11: Business Intelligence Tools

Course Objectives:

1. Introduce foundational concepts of Business Intelligence (BI) and Decision Support Systems (DSS), including their scope, evolution, and organizational relevance.
2. Familiarize students with leading BI tools such as Power BI and Tableau, highlighting their ecosystems, interfaces, and comparative strengths.
3. Develop skills in data preparation and transformation, using Power Query and Tableau's data connection features to clean and model datasets.
4. Enable effective data visualization and storytelling, leveraging charts, dashboards, and advanced features to communicate insights.
5. Equip learners with data modeling techniques, including dimensional modeling, relationships, joins, and governance principles for robust BI solutions.

Course Outcomes:

At the end of the course, the students will be able to:

1. Differentiate between BI, Data Analytics, and Data Science, and explain the BI lifecycle and its applications across functional domains.
2. Use Power BI and Tableau to prepare, transform, and visualize data, applying basic DAX functions and calculated fields for analysis.
3. Design and implement dimensional data models, including star and snowflake schemas, and apply relationships and joins in BI tools.
4. Create interactive dashboards and visualizations, incorporating parameters, slicers, filters, and drilldowns to enhance decision-making.
5. Build and publish complete BI dashboards, and effectively communicate business insights through storytelling and visualization best practices.

Unit-I: Introduction to Business Intelligence and Decision Support Systems

Business Intelligence: Definition, Scope, and Evolution, Business Intelligence vs. Data Analytics vs. Data Science, BI Lifecycle, **Applications of BI in Functional Domains:** Finance,

HR, Marketing, Retail, Education, Healthcare, etc., BI Maturity Models & Organizational Readiness to BI adoption Decision Support Systems (DSS): Concepts, Components, and Architecture.

BI Tools Overview: Power BI, Tableau, Google Looker Studio (Data Studio), Strengths, limitations, and comparison of BI tools, Choosing the right BI tool for organizational needs

Case Study: Retail Chain's BI Strategy to Optimize Inventory

Unit-II: Data Preparation and Visualization with Power BI

Power BI Ecosystem: Power BI Desktop, Service, Mobile, Interface overview and workspace organization

Data Preparation using Power Query: Importing Data (Excel, CSV, Databases, Web APIs), Data Cleaning: Missing values, formatting, transformations, Data Shaping: Merge, Append, Pivot/Unpivot, grouping

Basic DAX (Simplified): SUM, COUNT, AVERAGE, CALCULATE, IF, and simple measures

Visualization in Power BI: Basic charts: Table, Matrix, Column, Line, Area, Cards, Publishing and sharing reports via Power BI Service

Case Study: Student Performance Dashboard, Finance Dataset Analysis

Unit-III: Data Preparation, Visualization and Storytelling with Tableau

Tableau Fundamentals: Introduction and characteristics, Tableau Public, Desktop, Online, Interface: Shelves, Marks Card, Workspace, Views

Data Preparation in Tableau: Connections, cleaning, filtering, pivoting, Basic Calculated Fields

Visualizations: Bar, Line, Scatter, Maps, Tree Maps, Dashboards and basic interactions, Storytelling Concepts: Story Points, Designing a Tableau Story

Looker Studio (Google Data Studio): Connecting to Google Sheets/CSV/DB sources, Creating simple dashboards, Filters, controls, and sharing

Case Study: HR Analytics Dashboard

Unit-IV: Data Modeling and Relationships in BI Tools

Dimensional Modeling: Dimension, Dimension table, fact, fact table, schema, Star and Snowflake Schemas

Power BI: Relationships, Cardinality, Cross-filtering; Tableau: Joins (Inner, Left, Full), Blending; Data Governance: Metadata, Hierarchies, Quality; Data Model Design Best Practices - Design and implement data model in Power BI and Tableau; HR or Retail data model design and insights

Case Study: Retail BI for Sales Optimization

Unit-V: Dashboard Design and Business Insights

Introduction to Dashboard, when to use dashboards, Dashboard components, Principles of Effective Visualization & Dashboarding, Advanced Visualizations: Parameters, Slicers, Filters, Drilldowns, Graphs and Maps, Dashboard Design: Layout, Alignment, Accessibility

Publishing Dashboards: Power BI Service, Tableau Public; Storytelling and Insight Communication, Build a complete BI dashboard using either tool.

Case Study: Business decision-making scenario (e.g., Sales Forecasting, Budgeting)

Text Books:

1. Decision Support and Business Intelligence Systems (9th ed.). Turban, E., Sharda, R., & Delen, D. (2014), Pearson Education.
2. Learning Tableau 2022: Create effective data visualizations, build interactive dashboards, and transform your data into insights (6th Edition), Milligan, J. N. (2022)., Packt Publishing.
3. Expert Data Modeling with Power BI: Enrich and optimize your data models for reporting and business needs (2nd Edition). Bakhshi, S. (2023), Packt Publishing.

Reference Books:

1. Visual Analytics with Tableau, Loth, A. (2019), Addison-Wesley Professional.
2. The Definitive Guide to DAX: Business intelligence for Microsoft Power BI, SQL Server Analysis Services, and Excel (2nd Edition), Russo, M., & Ferrari, A. (2020), Microsoft Press.

Online Resources:

1. Decision Support Systems Courses – Class Central
2. Advanced Business Decision Support Systems – NPTEL (IIT Kanpur)
3. Power BI Learning Paths – Microsoft Learn
4. Power BI Courses – Coursera
5. Tableau Learning Hub – Official Site
6. Free Tableau Course – Simplilearn
7. Dashboard Design Concepts – DataCamp
8. Business Intelligence with Power BI – Swayam

Activities:

1. Differentiate BI, Data Analytics, and Data Science + BI Lifecycle

Activity: Case Study Analysis: Provide students with a business scenario and ask them to identify how BI, analytics, and data science each contribute. Then, have them map the BI lifecycle stages to that scenario and explain its impact across departments (e.g., finance, marketing).

Evaluation Method: Evaluate on a 10-point scale based on Conceptual differentiation clarity (30%), BI lifecycle accuracy (30%), Application to domains (20%) and Written summary or presentation quality (20%)

2. Prepare, Transform & Visualize Data in Power BI/Tableau (with DAX & Calculated Fields)

Activity: Hands-on Data Challenge: Supply students with a raw dataset (e.g. sales or customer data). Task them with cleaning, transforming, and building visuals in Power BI/Tableau using basic DAX and calculated fields (e.g. profit margins, year-over-year growth).

Evaluation Method: Evaluate for 10-points based on Effective transformation workflow (25%), Correct DAX/formula usage (25%), Visualization relevance & clarity (25%) and Documentation of process (25%)

3. Dimensional Modelling with Star/Snowflake Schemas + Joins/Relationships

Activity: Model Building Lab: Students design a star or snowflake schema based on a retail or HR database. Then, implement the schema in Power BI or Tableau and create relationships between tables to ensure correct joins and data flow.

Evaluation Method: Evaluate on a 10-point scale based on Schema design accuracy (30%), Appropriate schema selection (star vs. snowflake) (20%), Implementation of joins/relationships (30%) and Functional model validation (20%)

4. Create Interactive Dashboards with Advanced Features

Activity: Dashboard Design Workshop: Students build an interactive dashboard using parameters, slicers, filters, and drilldowns to simulate real-time decision-making (e.g., tracking regional product sales or employee performance across departments).

Evaluation Method: Evaluate on a 10-point scale on the basis of Integration of interactive features (30%), Usability and navigation experience (30%), Data-driven insights extracted (20%) and Design polish and layout consistency (20%).

5. Build and Publish BI Dashboards + Business Storytelling

Activity: BI Capstone Project: Students design and publish a complete dashboard solving a real or simulated business problem (e.g. customer churn, supply chain bottlenecks). Include visual storytelling techniques-titles, annotations, color themes, and narratives.

Evaluation Method: Evaluate on a 10-point scale on the basis of Clarity of business insights communicated (30%), Storytelling elements (25%), Dashboard completeness and polish (25%) and Peer or instructor presentation (20%)

C11P: Business Intelligence Tools Lab

List of Experiments:

1. Exploring BI Tools

- Introduction and comparison: **Power BI, Tableau, Looker Studio**
- Create a **simple retail dashboard** using any two tools

2. Connecting to Different Data Sources

- Load datasets from Excel, CSV, SQL, and Web
- Demonstrate schema view and applied transformations

3. Data Cleaning with Power Query

- Remove duplicates, handle missing values, filter rows
- Transform, merge, append, pivot/unpivot
- Submit the **Power Query applied steps** report

4. Case Study: Student Performance Analytics

- Clean and transform a **student performance dataset**
- Visualize KPIs such as GPA trend, pass percentage, subject-level insights
- Submit cleaned dataset + visualization outputs

5. Implementing Basic DAX Functions

- Create calculated columns & measures using:
  SUM, AVERAGE, COUNT, CALCULATE, IF
- Attach DAX expressions with visual outputs

6. Dimensional Modeling in Power BI

- Create **Star/Snowflake schema** using retail or HR data
- Define relationships, hierarchies, metadata

- Submit relationship diagram + explanation

7. Tableau Basics & Connecting to Data

- Load a dataset in Tableau Public
- Explore data structure and fields

8. Data Cleaning & Preparation in Tableau

- Perform pivoting, filtering, sorting, grouping
- Prepare clean worksheets

9. Creating Visualizations in Tableau

- Build charts using Shelves and Marks (Bar, Line, Scatter, Map)
- Combine visuals into a **dashboard with at least 3 charts**

10. Tableau Storytelling

- Build a **Tableau Story** (HR Analytics or Student Performance)
- Combine multiple sheets with narrative captions

11. Looker Studio (Google Data Studio)

- Load data from Google Sheets/CSV
- Create a simple dashboard with interactive filters

12. Interactive Dashboard in Power BI

- Add slicers, filters, drill-downs, drill-through
- Design a multi-level BI dashboard
- Submit the .pbix file + Power BI Service link

Track A (Application Development)

C12: Web Design & Development Fundamentals

Course Objectives

1. Understand the principles of web design and distinguish between web and desktop application architectures.
2. Develop static web pages using HTML elements, attributes, and multimedia integration techniques.
3. Style web pages effectively using CSS, including layout control, responsive design, and UI enhancements.
4. Implement dynamic behaviors and form validations using JavaScript and the Document Object Model (DOM).
5. Explore JSON and jQuery for handling structured data and simplifying client-side scripting in web development.

Course Outcomes

At the end of the course, students will be able to:

1. Design and structure HTML-based webpages incorporating text, images, tables, forms, and multimedia content.
2. Apply CSS styling rules to manage layout aesthetics, interactivity, and responsiveness across devices.
3. Use JavaScript for string manipulation, event handling, arrays, object operations, and basic validation.
4. Employ client-side scripting to enhance form functionality, create dialog interactions, and add animations via events.
5. Parse JSON data and use jQuery to simplify DOM manipulation, AJAX calls, and build dynamic, data-driven web applications.

Unit 1.HTML:

Introduction to web designing, difference between web applications and desktop applications, introduction to HTML, HTML structure, elements, attributes, headings, paragraphs, images, tables, lists, blocks, symbols, embedding multi-media components in HTML, HTML forms

Unit 2.CSS:

CSS home, introduction, syntax, CSS combinators, colors, background, borders, margins, padding, height/width, text, fonts, tables, lists, position, overflow, float, pseudo class, pseudo elements, opacity, tool tips, image gallery, CSS forms, CSS counters.

Unit 3.Java Script:

What is DHTML, JavaScript, basics, variables, operators, statements, string manipulations, mathematical functions, arrays, functions. objects, regular expressions, exception handling.

Unit 4. Client-Side Scripting:

Accessing HTML form elements using Java Script object model, basic data validations, data format validations, generating responsive messages, opening windows using java script, different kinds of dialog boxes, accessing status bar using java script, embedding basic animative features using different keyboard and mouse events.

Unit 5. JSON and jQuery

Introduction to JSON: Need for data exchange formats, JSON syntax, JSON vs XML, parsing JSON, creating JSON objects and arrays, accessing nested JSON data, reading/writing JSON in JavaScript.

Working with jQuery: Introduction to jQuery, selectors, filters, DOM manipulation, event handling, animations, effects, and chaining.

Text Book(s)

1. Web Programming: Building Internet Applications, Chris Bates, Wiley, Second Edition.
2. An Introduction to Web Design plus Programming, Paul S. Wang, Sanda S. Katila, Thomson.

3. Learning jQuery – Jonathan Chaffer, Karl Swedberg, Packt Publishing.
4. JSON at Work – Tom Marrs, O'Reilly Media.

Reference Books

1. Head First HTML and CSS, Elisabeth Robson, Eric Freeman, O'Reilly Media Inc.
2. An Introduction to HTML and JavaScript: for Scientists and Engineers, David R. Brooks, Springer.
3. Schaum's Easy Outline: HTML, David Mercer, McGraw Hill Professional.
4. jQuery in Action, Bear Bibeault, Yehuda Katz, Manning Publications.
5. Beginning JSON, Ben Smith, Apress.

Activities:

Outcome: Design and structure HTML webpages with text, images, tables, forms, and multimedia.

Activity: Create a personal profile webpage with all these elements.

Evaluation Method: Checklist (10 pts) - correct tags, structure, working forms/media.

Outcome: Apply CSS rules for layout, aesthetics, interactivity, and responsiveness.

Activity: Style the profile page using colors, fonts, Flexbox/Grid, hover effects, and media queries.

Evaluation Method: Rubric (10 pts) - visual appeal, responsiveness, selector/layout usage, clean code.

Outcome: Use JavaScript for string manipulation, events, arrays/objects, and validation.

Activity: Add form validation, greeting message, array/object display, and button click handling.

Evaluation Method: Demo & testing (10 pts) - correct syntax, validation, event handling, output behavior.

Outcome: Employ client-side scripting for dialogs, animations, and event-based interactions.

Activity: Add show/hide sections, confirmation dialog on submit, and animations via mouse/keyboard events.

Evaluation Method: Live demo (10 pts) - smooth interaction, correct event listeners, proper animations, good UX.

Outcome: Parse JSON and use jQuery for DOM manipulation, AJAX, and dynamic data display.

Activity: Fetch JSON via jQuery AJAX, render as table/list, add filter/sort, and compute summaries.

Evaluation Method: Demo & checklist (10 pts) - JSON parsing, jQuery usage, filter/sort functionality, error handling.

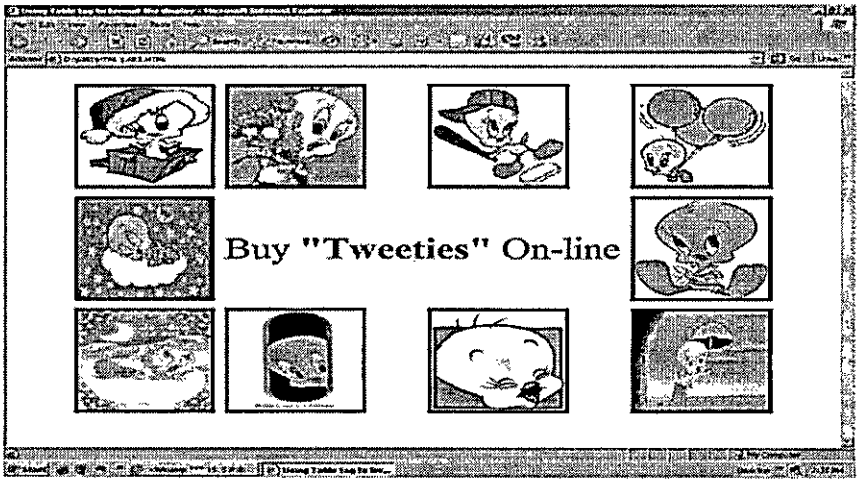
C12P: Web Design & Development Fundamentals Lab

List of Experiments:

1. Create an HTML document with the following formatting options:

   (a) Bold, (b) Italics, (c) Underline, (d) Headings (Using H1 to H6 heading styles), (e) Font (Type, Size and Color), (f) Background (Colored background/Image in background), (g) Paragraph, (h) Line Break, (i) Horizontal Rule; (j) Pre tag
2. Create an HTML document which consists of:

   (a) Ordered List (b) Unordered List (c) Nested List (d) Image
3. Collect any ten images of your choice. Using table tag, align the images as follows:



4. Create a form using HTML which has the following types of controls:

   (a) Text Box (b) Option/radio buttons (c) Check boxes (d) Reset and Submit buttons
5. Embed a calendar object in your web page.

6. Create a form that accepts the information from the subscriber of a mailing system.
7. Apply CSS to design a student registration form (use different selectors, colors, borders, spacing).
8. Create a responsive webpage using CSS Flexbox/Grid.
9. Add hover effects and transitions on images and buttons using CSS.
10. Write a JavaScript program to perform string operations (reverse, substring, count vowels).
11. Create a JavaScript form validation program (check email format, password length, required fields).
12. Develop a webpage that displays greetings based on the current time (morning, afternoon, evening).
13. Use JavaScript to manipulate arrays and objects (add, delete, sort, search).
14. Fetch and display student information stored in a JSON object on a webpage.
15. Fetch real-time weather data from an open API (e.g., OpenWeatherMap) in JSON format and display temperature, humidity, and conditions dynamically on a webpage.
16. Use jQuery to simplify DOM manipulation (hide, show, fade, slide, toggle).

C13: Web Application Development Using PHP & MySQL

Course Objectives:

1. Understand the foundational elements of PHP, including variables, data types, operators, and flow control functions.
2. Develop proficiency in managing arrays, objects, strings, dates, and time functionalities using PHP.
3. Design and process HTML forms integrated with PHP, including advanced operations like file uploads, redirection, and exception handling.
4. Implement session management and cookie handling to preserve user state and provide secure, personalized experiences.
5. Connect PHP with MySQL databases, enabling learners to perform CRUD operations and build dynamic web applications.

Course Outcomes

At the End of the Course, The Students will be able to:

1. Demonstrate effective use of PHP building blocks such as variables, expressions, constants, control structures, and functions with appropriate scope and argument handling.
2. Manipulate complex data structures including arrays and objects, and utilize PHP string and date/time functions for dynamic content generation.
3. Create functional web forms using PHP, retrieve form inputs, manage file uploads, and execute form-based operations like redirection and email dispatch.
4. Apply secure techniques for maintaining user sessions and cookies, including session lifecycle control, session variable manipulation, and user authentication workflows.
5. Integrate PHP with MySQL to build database-driven web components, perform record management, and architect structured menu-based data operations.

Unit 1. The building blocks of PHP:

Variables, Data Types, Operators and Expressions, Constants.

Flow Control Functions in PHP: Switching Flow, Loops, Code Blocks and Browser Output.

Working with Functions: Creating functions, Calling functions, Returning the values from User-Defined Functions, Variable Scope, Saving state between Function calls with the static statement, arguments of functions

Unit 2.Working with Arrays:

Creating Arrays, Some Array-Related Functions.

Working with Objects: Creating Objects, Accessing Object Instances,

Working with Strings, Dates and Time: Formatting strings with PHP, Manipulating Strings with PHP, Using Date and Time Functions in PHP.

Unit 3.Working with Forms:

Creating Forms, Accessing Form Input with User defined Arrays, Combining HTML and PHP code on a single Page, Using Hidden Fields to save state, Page redirection, Sending Mail on Form Submission, Working with File Uploads, Managing files on server, Exception handling.

Unit 4. Working with Cookies and User Sessions:

Introducing Cookies, setting a Cookie with PHP, Session Function Overview, starting a Session, working with session variables, passing session IDs in the Query String, Destroying Sessions and Unsetting Variables, Using Sessions in an Environment with Registered Users.

Unit 5.Interacting with MySQL using PHP:

MySQL Versus MySQLi Functions, connecting to MySQL with PHP, Working with MySQL Data. Planning and Creating Database Tables, Creating Menu, Creating Record Addition Mechanism, Viewing Records, Creating the Record Deletion Mechanism.

Text Book(s)

1. SAMS Teach yourself PHP MySQL and Apache,, Julie C. Meloni, Pearson Education
2. PHP: The Complete Reference, Steven Holzner , McGraw-Hill

Reference Books

1. Learning PHP, MySQL, JavaScript, CSS & HTML5, Robin Nixon, Third Edition, O'reilly, 2014
2. The web warrior guide to Web Programming, Xue Bai Michael Ekedahl,, Thomson, 2006.

Activities:

Outcome: Demonstrate effective use of PHP building blocks such as variables, expressions, constants, control structures, and functions with appropriate scope and argument handling.

Activity: Write a PHP script that:

- Declares variables and constants
- Uses expressions and control structures (if, switch, loops)
- Defines and calls functions with arguments and return values
- Demonstrates variable scope (global, local)

Evaluation Method: Evaluate on a 10-point scale based on code review checklist to verify the

- Correct syntax and use of each building block
- Logical flow using control structures
- Proper function definition and scope handling

- Output accuracy and readability

Outcome: Manipulate complex data structures including arrays and objects, and utilize PHP string and date/time functions for dynamic content generation.

Activity: Create a PHP script that:

- Stores student data in arrays and objects
- Formats and manipulates strings (e.g., name formatting)
- Displays current date/time and calculates age from DOB

Evaluation Method: Rubric-based assessment on a 10-point scale to check the:

- Correct use of arrays and objects
- Effective string and date/time functions
- Dynamic output generation
- Code clarity and structure

Outcome: Create functional web forms using PHP, retrieve form inputs, manage file uploads, and execute form-based operations like redirection and email dispatch.

Activity: Build a contact form that:

- Accepts name, email, message
- Uploads a file (e.g., resume)
- Sends an email confirmation
- Redirects to a thank-you page

Evaluation Method: Evaluate on a 10-point scale based on functional testing to perform:

- Form input retrieval and validation
- File upload success
- Email dispatch and redirection
- Error handling and user feedback

Outcome: Apply secure techniques for maintaining user sessions and cookies, including session lifecycle control, session variable manipulation, and user authentication workflows.

Activity: Develop a login system that:

- Starts a session on login
- Stores user data in session variables

- Sets a cookie for Remember Me
- Logs out and destroys session securely

Evaluation Method: Security checklist:

- Session lifecycle control
- Cookie setup with secure flags
- Authentication logic
- Session/cookie cleanup on logout

Outcome: Integrate PHP with MySQL to build database-driven web components, perform record management, and architect structured menu-based data operations.

Activity: Create a student management system:

- Connect to MySQL database
- Add, view, update, delete student records
- Display menu-based navigation (e.g., by class or grade)

Evaluation Method: Database interaction test on a 10-point scale:

- Successful CRUD operations
- Structured menu navigation
- SQL query correctness

C13P: Web Application Development Using PHP & MySQL Lab

List of Experiments:

1. Write a PHP program to Display Hello
2. Write a PHP Program to display today's date.
3. Write a PHP program to display Fibonacci series.
4. Write a PHP Program to read the employee details.
5. Write a PHP program to prepare the student marks list.
6. Create student registration form using text box, check box, radio button, select, submit button. And display user inserted values in the new PHP page.

7. Create Website Registration Form using text box, check box, radio button, select, submit button. And display user inserted values in the new PHP page.

8. Write a PHP script to demonstrate passing variables with cookies.

9. Write a PHP script to connect to the MySQL server from your website.

10. Write a program to keep track of how many times a visitor has loaded the page.

11. Write a PHP application to perform CRUD (Create, Read, Update and Delete) operations on a database table.

12. Create a web site using any open-source framework built on PHP and MySQL – It is a team activity wherein students are divided into multiple groups and each group comes up with their own website with basic features.

Semester-VI

C14: Mobile Application Development

Course Objectives

1. Understand core concepts of mobile app development and differentiate between native and cross-platform approaches.
2. Set up the Flutter and Dart development environment and apply foundational Dart programming constructs.
3. Design interactive and responsive UIs using Flutter widgets and implement custom design elements.
4. Develop multi-screen applications with effective state management and navigation techniques.
5. Integrate external data through APIs, manage local storage, and incorporate Firebase for cloud functionality.
6. Utilize advanced Flutter features, optimize performance, and deploy apps to the Google Play Store.

Course Outcomes

At the End of the Course, The Students will be able to:

1. Configure the Flutter SDK and development tools, and write Dart programs using object-oriented principles and error-handling mechanisms.
2. Create visually consistent and interactive user interfaces using built-in Flutter widgets and custom styling.
3. Implement navigation between screens and manage application state using Provider and other built-in mechanisms.
4. Consume RESTful APIs, parse JSON data, and display structured content using dynamic UI components like ListView and GridView.
5. Incorporate device-level features (camera, location), apply animations, and package Flutter apps for deployment on Android platforms. \

Unit 1. Introduction to Flutter and Dart:

Overview of mobile app development trends, Native vs. cross-platform development, Introduction to Flutter SDK and its architecture, Setting up Flutter & Dart environment (IDE, emulator, device) (Install Flutter SDK and set up IDE (VS Code / Android Studio), Dart syntax: variables, data types, control structures, Functions, classes, and object-oriented principles in Dart, Error handling and assertions.

Unit 2. Flutter Widgets and UI Design:

Stateless vs Stateful widgets, Layout widgets: Container, Row, Column, Stack, Input & selection widgets: TextField, Checkbox, Radio, Switch, Styling widgets: Padding, Margin, Fonts, Colors, Custom widgets and theming

Unit 3. Navigation and State Management:

Navigation: Navigator, routes, passing data between screens,
State management: setState, InheritedWidget, Provider, Dialogs, alerts, and snackbars, Forms and validation

Unit 4. Working with Data and APIs:

HTTP package for API calls, JSON parsing and model classes, Displaying data with ListView and GridView, Local storage: SharedPreferences, File handling, Firebase integration (basic setup & Firestore)

Unit 5. Advanced Features and Deployment:

Animations and custom transitions, Accessing device features: Camera, Location, Introduction to packages and plugins, Debugging and performance optimization, Building and releasing apps to Google Play Store

Textbooks:

1. Beginning Flutter: A Hands-On Guide to App Development, Marco L. Napoli, Wiley
2. Flutter for Beginners (2nd Edition), Alessandro Biessek, Packt Publishing

Reference Books:

1. Flutter for Mobile Apps: Miguel Farmer, Rafael Sanders, Lincoln Publishers
2. Flutter Development Masterclass, E.M. Redwood, 2025

Activities:

Outcome: Configure the Flutter SDK and development tools and write Dart programs using object-oriented principles and error-handling mechanisms.

Activity: Set up Flutter SDK and create a Dart console app that:

- Uses object-oriented principles (classes, inheritance)
- Includes error-handling (try-catch-finally)
- Demonstrates basic input/output

Evaluation Method: Evaluate on a 10-point scale based on a checklist to verify:

- SDK and IDE properly configured
- Correct use of classes and inheritance
- Functional error-handling logic
- Output correctness and code readability

Outcome: Create visually consistent and interactive user interfaces using built-in Flutter widgets and custom styling.

Activity: Build a login screen using:

- Built-in widgets (TextField, Button, Column, etc.)
- Custom styling (colors, fonts, padding)
- Responsive layout

Evaluation Method: Rubric-based assessment on a 10-point scale:

- UI consistency and alignment
- Use of appropriate widgets
- Styling customization
- Responsiveness across screen sizes

Outcome: Implement navigation between screens and manage application state using Provider and other built-in mechanisms.

Activity: Create a multi-screen app with:

- Home, Profile, and Settings screens
- Navigation using Navigator
- State management using Provider (e.g., toggle dark mode)

Evaluation Method: Functional testing:

- Smooth navigation between screens
- Correct state updates via Provider
- Code structure and separation of concerns
- UI reflects state changes

Outcome: Consume RESTful APIs, parse JSON data, and display structured content using dynamic UI components like ListView and GridView.

Activity: Build a news app that:

- Fetches articles from a public REST API
- Parses JSON response
- Displays data in ListView and GridView

Evaluation Method: Live demo and code inspection:

- API integration and error handling
- JSON parsing accuracy
- Dynamic UI rendering
- Performance and responsiveness

Outcome: Incorporate device-level features (camera, location), apply animations, and package Flutter apps for deployment on Android platforms.

Activity: Create a photo journal app that:

- Captures images using device camera
- Retrieves current location
- Applies basic animations (e.g., fade-in)
- Packages and runs on Android device

Evaluation Method: Device-based testing to check:

- Camera and location permissions handled
- Feature functionality verified
- Animation smoothness
- Successful APK build and installation

C14P: Mobile Application Development Lab

List of Experiments:

1. Write Basic Dart Programs Using Variables, Functions, and Classes
2. Design a UI Layout Using Container, Row, Column, and Stack Widgets
3. Create an Interactive Form Using TextField, Checkbox, Radio, and Switch
4. Implement Custom Widgets and Apply Theming (Colors, Fonts, Styles)
5. Navigate Between Screens and Pass Data Using Navigator and Routes
6. Manage State Using setState and Provider Package
7. Create and Validate a Registration Form Using TextFormField and Validators
8. Fetch and Display JSON Data from a Public API Using HTTP Package
9. Parse JSON into Model Classes and Display Using ListView
10. Use SharedPreferences to Store and Retrieve Local Data

11. Integrate Firebase Firestore: Add and Retrieve Data
12. Implement Basic Animations Using AnimatedContainer and Hero Widgets
13. Access Device Camera or Location Using Flutter Plugins
14. Debug and Optimize Flutter Apps Using DevTools

C15: MERN STACK

Course Objectives:

1. Understand full-stack architecture and the individual roles of the MERN components-MongoDB, Express.js, React.js, and Node.js.
2. Develop interactive front-end applications using React.js and master state management and routing techniques.
3. Model and manipulate NoSQL databases using MongoDB and Mongoose, including CRUD operations and schema validations.
4. Build RESTful APIs using Express.js and integrate server-side logic with frontend and database components.
5. Implement full-stack application features such as authentication, session management, and deployment using modern platforms.

Course Outcomes:

1. Explain the architecture of the MERN stack and configure a Node.js development environment with essential modules and server setup.
2. Develop dynamic user interfaces using React functional components, hooks, forms, and routing for seamless user experiences.
3. Perform database operations using MongoDB and Mongoose, including schema design, data validation, and relational mapping.
4. Construct RESTful backend services using Express.js with routing, middleware, and error handling mechanisms.
5. Integrate frontend and backend components with secure data flow, apply JWT-based authentication, and deploy applications on platforms like Render, Netlify, or Vercel.

Unit 1. Introduction to MERN Stack & Node.js:

Introduction to Full Stack Web Development, Frontend vs Backend, What is the MERN stack? Architecture of MERN Applications

Introduction to Node.js, Installing Node.js & npm, Node.js fundamentals (Modules, Events, Streams), Asynchronous Programming & Event Loop, Node.js File System module, npm modules & custom modules, Setting up a basic server with Node.js

Unit 2. React.js (Frontend Framework):

Introduction to React, Functional Components & JSX, State & Props, Handling Events, useState, useEffect Hooks, Conditional Rendering & Lists, React Router (Routing), Forms in React (Controlled Components), Axios for HTTP requests

Unit 3. MongoDB with Mongoose:

Introduction to NoSQL Databases, MongoDB vs SQL Databases, Installing & using MongoDB (local & Atlas), CRUD Operations with MongoDB Shell & Compass, Mongoose ODM, Models, Schemas, Validation, Relationships (One-to-Many, Many-to-Many), Aggregation

Unit 4. Express.js (Backend Framework):

Introduction to Express.js, Creating RESTful APIs using Express, Routing (GET, POST, PUT, DELETE), Middleware in Express, Error Handling, Connecting to MongoDB using Mongoose, Environment variables and `.env` files

Unit 5. Full Stack Integration & Deployment:

Connecting Frontend (React) with Backend (Express), CORS and Proxy setup, Authentication with JWT, Protected Routes in Frontend, Role-based access control, Deployment: Deploying backend to Render/Heroku, Deploying frontend to Netlify/Vercel, Connecting MongoDB Atlas

Textbooks:

1. Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node, Subramanian, Vasan, 2nd Edition, Apress,

2. Learning React: Functional Web Development with React and Redux, Alex Banks & Eve Porcello, O'Reilly

3. MongoDB: The Definitive Guide, 3rd Edition, Shannon Bradshaw, Eoin Brazil, Kristina Chodorow, O'Reilly

Reference Books:

1. Ultimate Full-Stack Web Development with MERN, Nabendu Biswas, Orange Education Pvt Ltd.

2. Full Stack Development with MERN, Thompson Carter, Lincoln Publishers

Activities:

Outcome: Explain the architecture of the MERN stack and configure a Node.js development environment with essential modules and server setup.

Activity: Set up a basic Node.js project with Express and required modules (express, nodemon, dotenv). Create a simple server that responds with Hello MERN Stack on a browser.

Evaluation Method: Checklist-based review:

- Correct installation of Node.js and modules
- Functional server response
- Use of environment variables
- Folder structure and code clarity

Outcome: Develop dynamic user interfaces using React functional components, hooks, forms, and routing for seamless user experiences.

Activity: Build a multi-page React app (e.g., user profile and contact form) using:

- Functional components
- useState and useEffect hooks
- Controlled form inputs
- React Router for navigation

Evaluation Method: Live demo and rubric:

- Component structure and reusability
- Hook usage and state management
- Form validation and routing functionality

- UI responsiveness and layout

Outcome: Perform database operations using MongoDB and Mongoose, including schema design, data validation, and relational mapping.

Activity: Create a student database using MongoDB and Mongoose. Implement:

- Schema with validation rules
- CRUD operations (Create, Read, Update, Delete)
- Reference between collections (e.g., student and course)

Evaluation Method:

Code walkthrough and test cases:

- Schema correctness and validation
- CRUD functionality
- Relational mapping using ref
- Console output and error handling

Outcome: Construct RESTful backend services using Express.js with routing, middleware, and error handling mechanisms.

Activity: Develop a REST API for a task manager app using Express.js. Include:

- Routes for task operations
- Middleware for logging and JSON parsing
- Error handling for invalid routes

Evaluation Method: API testing with Postman:

- Route functionality and status codes
- Middleware implementation
- Error response structure
- Code readability and modularity

Outcome: Integrate frontend and backend components with secure data flow, apply JWT-based authentication, and deploy applications on platforms like Render, Netlify, or Vercel.

Activity: Build a login system with:

- JWT-based authentication

- Protected routes
- Frontend-backend integration
- Deploy frontend on Netlify/Vercel and backend on Render

Evaluation Method: Deployment demo and checklist:

- Token generation and validation
- Secure data flow between client and server
- Working login/logout flow
- Successful deployment and accessibility

C15P: MERN STACK

List of Experiments:

1. Install Node.js and npm; Create and Use Built-in & Custom Node.js Modules
2. Demonstrate Asynchronous Programming Using Callbacks and Promises
3. Implement File Read/Write Operations Using Node.js File System Module
4. Create a React App and Build Functional Components Using JSX
5. Handle Events and Use useState & useEffect Hooks in React
6. Implement Navigation Between Pages Using React Router
7. Create and Validate a Form Using Controlled Components in React
8. Install and Connect to MongoDB (Local and MongoDB Atlas)
9. Perform CRUD Operations Using MongoDB Shell and MongoDB Compass
10. Create Mongoose Schemas and Models, and Connect Them to a Node.js App
11. Create a RESTful API Using Express.js (GET, POST, PUT, DELETE)
12. Use Middleware and Error Handling in an Express App
13. Connect Express App to MongoDB Using Mongoose
14. Implement User Authentication Using JWT in Express and React
15. Create Protected Routes and Role-Based Access Control in React
16. Deploy Backend to Render/Heroku and Frontend to Netlify/Vercel

Track B (AIML and Data Engineering)

Semester-V

C12: Artificial Intelligence

Course Objectives

1. Understand the fundamental concepts, history, types, and applications of Artificial Intelligence.
2. Develop problem-solving skills using state-space representations and search strategies for AI applications.
3. Apply informed and advanced search techniques including heuristics, local search, genetic algorithms, and constraint satisfaction problems.
4. Learn knowledge representation methods and reasoning techniques using propositional and first-order logic for intelligent agents.
5. Explore expert systems, probabilistic reasoning, fuzzy logic, and emerging AI technologies including NLP, robotics, and ethical considerations.

Course Outcomes

At the end of the course, students will be able to:

1. Explain the concepts, scope, types of AI, and structure of intelligent agents and their environments.
2. Formulate problems using state-space representation and solve them using uninformed search strategies like BFS, DFS, and Uniform Cost Search.
3. Apply informed search, local search, genetic algorithms, and constraint satisfaction techniques to solve complex AI problems.
4. Represent knowledge using propositional and first-order logic, perform reasoning, and design knowledge-based agents.
5. Design simple expert systems, implement probabilistic reasoning and fuzzy logic, and demonstrate awareness of emerging AI technologies and ethical issues.

Unit 1. Introduction to Artificial Intelligence & Intelligent Agents:

Definition and scope of AI, history and evolution of AI, Turing Test, Applications of AI in real world.

Types of AI: Weak AI vs Strong AI, Narrow AI vs General AI. Intelligent Agents: Structure of agents, Rationality, Agent types. Environments: Deterministic vs Stochastic, Static vs Dynamic, Discrete vs Continuous. PEAS representation (Performance measure, Environment, Actuators, Sensors).

Unit 2. Problem Solving: State Space & Uninformed Search:

State space representation: Components (State, Actions, Goal test, Path cost). Problem formulation and examples (8-puzzle, water jug, vacuum cleaner world).

Uninformed search strategies: Breadth First Search (BFS), Depth First Search (DFS), Uniform Cost Search- Properties: Completeness, Optimality, Time & Space complexity.

Unit 3. Informed & Advanced Search Strategies:

Informed search strategies: Heuristics (concept, admissibility, consistency), Greedy Best First Search, A* Algorithm

Local Search: Hill Climbing, Simulated Annealing. Genetic Algorithms

Constraint Satisfaction Problems (CSP): Definition, Backtracking search.

Unit 4. Knowledge Representation & Reasoning:

Knowledge Representation: Issues, Approaches.

Propositional Logic: Syntax, Semantics, Truth tables, Inference rules.

First Order Logic (FOL): Syntax, Semantics, Quantifiers, Substitution, Unification.

Inference in Logic: Forward Chaining, Backward Chaining, Resolution.

Knowledge-based agents.

Unit 5. Expert Systems, Probabilistic & Emerging AI:

Expert Systems: Architecture, Knowledge base, Inference engine, Explanation facility.

Probabilistic Reasoning: Bayes Theorem, Bayesian Belief Networks (concepts & examples)

Fuzzy Logic and Uncertainty handling.

Emerging topics: NLP basics, Robotics, AI Ethics & societal impact.

Textbooks:

1. Artificial Intelligence: A Modern Approach, Stuart Russell & Peter Norvig, 4th Edition, Pearson
2. Artificial Intelligence, Elaine Rich & Kevin Knight, 3rd Edition, McGraw-Hill

Reference Books:

1. The Art of Prolog, Leon Sterling & Ehud Shapiro, MIT Press
2. Learn Prolog Now, Patrick Blackburn, Johan Bos, Kristina Striegnitz (Free online book)

Activities:

Outcome: Explain the concepts, scope, types of AI, and structure of intelligent agents and their environments.

Activity: Divide the class into small groups and ask each group to create a poster or digital infographic comparing Weak AI vs Strong AI, and Narrow AI vs General AI, including real-world examples of intelligent agents and their environments.

Evaluation Method: Peer and instructor review rubric (10 points) assessing correctness of concepts, clarity of illustrations, examples provided, and creativity in presentation.

Outcome: Formulate problems using state-space representation and solve them using uninformed search strategies like BFS, DFS, and Uniform Cost Search.

Activity: Provide students with a simple problem like the 8-puzzle or water jug problem. Students model the state space, draw the state tree, and manually perform BFS and DFS traversal to reach the goal state.

Evaluation Method: Submission of state-space diagrams and solution steps, graded on accuracy, completeness, and correct application of search strategies.

Outcome 3: Apply informed search, local search, genetic algorithms, and constraint satisfaction techniques to solve complex AI problems.

Activity: Conduct a mini-coding session in Python where students implement A* search for a maze-solving problem or hill climbing for a simple optimization task. They can also experiment with a genetic algorithm for a simple fitness function problem.
Evaluation Method: Code demonstration and correctness check, including explanation of heuristics, search path, and results.

Outcome: Represent knowledge using propositional and first-order logic, perform reasoning, and design knowledge-based agents.
Activity: Give students a logic puzzle (e.g., Sudoku rules or "who owns which pet" problem). Students write propositional and/or FOL statements, draw inference chains, and perform forward/backward chaining to solve the puzzle.
Evaluation Method: Solution submission and in-class demonstration of inference process, graded on logical correctness, clarity of reasoning, and completeness.

Outcome: Design simple expert systems, implement probabilistic reasoning and fuzzy logic, and demonstrate awareness of emerging AI technologies and ethical issues.
Activity: Ask students to design a rule-based expert system for a small domain (e.g., medical symptom checker), represent a simple Bayesian network for probabilistic reasoning, and prepare a short presentation on AI ethics or NLP applications.
Evaluation Method: Project-based evaluation including expert system rules, Bayesian reasoning correctness, and quality of presentation on emerging AI topics.

C12P: Artificial Intelligence Lab

SWI-Prolog environment for practice without installation.

1. . Write Prolog facts for a family tree.
    - Define rules for ancestor/2, sibling/2, cousin/2.
    - Query for ancestors, descendants, and siblings.
2. Implement member/2, append/3, reverse/2, length/2.
3. Write a predicate to find the maximum element of a list.
4. Flatten a nested list into a single-level list.
5. Write Prolog rules to calculate factorial and Fibonacci numbers.

6. Implement GCD of two numbers using recursion.
7. Demonstrate the use of cut (!) and fail predicates.
8. Represent a simple graph using edge/2 predicates.
9. Write a recursive DFS to find a path between two nodes.
10. Implement BFS to find a path between two nodes in a graph.
11. Compare DFS and BFS by finding path lengths.
12. Solve the 8-puzzle or grid problem using Greedy Best-First search and A*.
13. Represent a map with regions and adjacency constraints.
    - Assign colors to regions using backtracking.
    - Ensure no adjacent regions share the same color.
14. Place N queens on an N×N chessboard such that no two queens threaten each other.
    - Generate all possible solutions using backtracking.
15. Encode facts and rules using propositional and first-order logic.
16. Implement forward and backward chaining for simple queries.
17. Build a rule-based expert system (e.g., medical diagnosis or plant disease).
    - Include knowledge base, inference engine, and explanation facility.
    - Test the system with sample queries.
18. Write a DCG grammar for simple English sentences.
    - Parse sentences to generate a syntax tree.
19. Implement a simple deterministic Naïve Bayes calculation for categorical data.

C13: Big Data Technologies:

Course Objectives:

1. Introduce students to the concepts, characteristics, and challenges of Big Data.
2. Familiarize students with the Hadoop ecosystem and its core components (HDFS, YARN, MapReduce).
3. Develop practical knowledge of distributed storage and parallel processing in Hadoop.
4. Provide hands-on exposure to data ingestion tools (Sqoop, Flume) and serialization techniques.
5. Enable students to explore NoSQL databases (HBase), coordination services (ZooKeeper), and Hadoop–Spark integration for large-scale data analysis.

Course Outcomes:

At the end of this course, students will be able to:

1. Explain Big Data concepts and challenges along with the role of the Hadoop ecosystem.
2. Demonstrate understanding of HDFS and YARN architectures and their functions in distributed data management.
3. Apply MapReduce and high-level tools (Hive, Pig, Spark) to process and analyze large datasets.
4. Design and implement data ingestion workflows using Sqoop, Flume, and serialization formats like Avro and Parquet.
5. Utilize NoSQL databases and ecosystem enhancements such as HBase, ZooKeeper, and Hadoop–Spark integration for scalable big data solutions.

Unit 1. Foundations of Big Data & Hadoop Ecosystem

Introduction to Big Data: characteristics (volume, variety, velocity, veracity, value), Hadoop Ecosystem Overview: HDFS, MapReduce, YARN, Hadoop Common, Hadoop architecture and use cases

Unit 2. Hadoop Distributed File System (HDFS) & YARN:

Deep dive into HDFS architecture: blocks, NameNode, DataNodes, HDFS file operations, fault tolerance, replication

YARN architecture: ResourceManager, NodeManager, application scheduling

Unit 3. MapReduce & High-Level Tools

MapReduce programming model: map, shuffle, reduce phases, Writing MapReduce applications in Hadoop

High-level abstractions: Hive, Pig, Crunch, and introduction to Spark integration

Unit 4. Data Ingestion & Serialization

Data ingestion pipelines: Sqoop (for RDBMS), Flume (streaming), Data formats & serialization: Avro, Parquet, SequenceFile, Practical ingestion workflows-batch and streaming

Unit 5. NoSQL & Ecosystem Enhancements

Overview of NoSQL within Hadoop ecosystem: HBase, Configuration and usage of ZooKeeper for coordination, Hadoop integration with Spark for data processing

Textbooks

1. Hadoop: The Definitive Guide, Tom White, 4th Edition, O'Reilly
   Free Resource available at piazza-resources.s3.amazonaws.comO'Reilly Media
2. Learning Spark, 2nd Edition, Jules S. Damji, Brooke Wenig, Tathagata Das, Denny Lee, O'Reilly

Reference Books

3. BIG DATA, Black Book TM, DreamTech Press, 2016 Edition.
4. BIG DATA and ANALYTICS, Seema Acharya, SubhasniChellappan , Wiley publications, 2016

Activities:

Outcome: Explain Big Data concepts and Hadoop ecosystem

Activity: Short seminar / presentation on Big Data applications

Evaluation Method: Oral presentation + concept quiz

Outcome: Demonstrate HDFS and YARN architectures

Activity: Hands-on lab to configure HDFS & analyze NameNode/DataNode logs

Evaluation Method: Lab performance + viva

Outcome: Apply MapReduce and high-level tools

Activity: Mini-project implementing MapReduce job and Hive queries

Evaluation Method: Project report + execution demo

Outcome: Design and implement data ingestion workflows

Activity: Lab task to ingest data using Sqoop/Flume and serialize with Avro/Parquet

Evaluation Method: Lab record + output validation

Outcome: Utilize NoSQL and ecosystem enhancements

Activity: Case study on HBase–Spark integration with example dataset

Evaluation Method: Case study report + written test

C13P: Big Data Technologies Lab:

1. Installation & setup of Hadoop single-node cluster
2. Explore Hadoop directory structure and basic commands (hadoop fs operations)
3. Demonstration of Hadoop architecture components (HDFS, YARN, MapReduce) using sample logs
4. Store and retrieve large files from HDFS (block distribution, replication factor demo)
5. Simulate NameNode/DataNode failure and observe fault tolerance & recovery
6. Configure YARN and run sample applications, observe ResourceManager and NodeManager roles
7. Write a simple MapReduce program for word count
8. Develop a MapReduce job for inverted index creation
9. Perform data analysis using Pig Latin scripts
10. Execute Hive queries for structured data analysis (tables, partitions)
11. Import data from RDBMS into Hadoop using Sqoop
12. Capture and store log/streaming data using Flume
13. Serialize and store datasets in Avro and Parquet formats
14. Build an end-to-end ingestion workflow combining batch (Sqoop) and streaming (Flume)
15. Create and manage tables in HBase (CRUD operations)
16. Demonstrate coordination with ZooKeeper
17. Process HBase datasets using Spark integration with Hadoop

Semester-VI

C14: Machine Learning

Course Objectives:

1. Understand fundamental concepts, types, and applications of machine learning.
2. Develop, evaluate, and optimize machine learning models through preprocessing, training, and feature engineering techniques.
3. Apply supervised and unsupervised learning algorithms to real-world problems using appropriate tools and methods.

Course Outcomes:

Upon successful completion of this course, students will be able to:

1. Describe various machine learning paradigms, data types, and the overall structure of a machine learning pipeline.
2. Perform data preprocessing, feature engineering, and evaluate models using appropriate metrics.
3. Implement and analyze supervised learning algorithms for regression and classification tasks.
4. Apply unsupervised learning techniques for clustering and identify suitable machine learning approaches for specific application domains

Unit 1. Introduction to Machine Learning:

Introduction to Machine Learning: Types of human learning, What is machine learning?, Types of machine learning: supervised, unsupervised, semi-supervised and reinforcement learning, machine learning activities, applications of machine learning. Types of data in machine learning, Structure of data

Unit 2. Model Preparation, Evaluation and feature engineering:

Data pre-processing, Model selection and training (for supervised learning), Model representation and interpretability, Evaluating machine learning algorithms and performance

enhancement of models. What is feature engineering?, Feature transformation, Feature subset selection. Principal component analysis.

Unit 3. Supervised Learning-Regression:

Regression: Introduction of regression, Regression algorithms: Simple linear regression, Multiple linear regression, Polynomial regression model, Logistic regression, Maximum likelihood estimation.

Unit 4. Supervised Learning- Classification:

Introduction of supervised learning, Classification model and learning steps, Classification algorithms: Naïve Bayes classifier, k-Nearest Neighbour (kNN), Decision tree, Support vector machines, Random Forest.

Unit 5. Unsupervised Learning:

Introduction of unsupervised learning, Unsupervised vs supervised learning, Application of unsupervised learning, Clustering and its types, Partitioning method: k-Means and KMedoids, Hierarchical clustering, Density-based methods - DBSCAN.

Case-study of ML applications: Image recognition, speech recognition, Email spam filtering, Online fraud detection and other.

Textbooks:

1. Introduction to Machine Learning, Ethem Alpaydin, MIT Press, Fourth Edition, 2020.
2. Machine Learning: Theory and Practice, M N Murthy, V.S Ananthanarayana, Universities press
3. Machine Learning, S. Sridhar, M. Vijayalakshmi, Second Edition, Oxford University Press

Reference Books:

1. Machine Learning: An Algorithmic Perspective, Second Edition, Stephen Marsland, CRC Press, 2014
2. Machine Learning, Tom Mitchell, McGraw Hill, 3rd Edition.
3. Python Machine Learning, Sebastain Raschka, Vahid Mirjalili , Packt publishing 3rd Edition, 2019.

Activities:

Outcome: Describe various machine learning paradigms, data types, and the overall structure of a machine learning pipeline.

Activity: Prepare a detailed comparative report/chart explaining supervised, unsupervised, and reinforcement learning paradigms, data types, and step-by-step machine learning pipeline stages.

Evaluation Method: Rubric-based assessment evaluating completeness, clarity, correctness, and presentation quality - scored on a 10-point scale.

Outcome: Perform data preprocessing, feature engineering, and evaluate models using appropriate metrics.

Activity: Conduct a hands-on lab exercise using a real dataset to perform data cleaning, normalization, feature extraction/selection, and evaluate model performance using metrics like accuracy, precision, recall, and F1-score.

Evaluation Method: Practical assessment including code correctness, applied techniques, and interpretation of evaluation metrics; assessed with a rubric out of 10.

Outcome: Implement and analyze supervised learning algorithms for regression and classification tasks.

Activity: Implement at least two supervised learning algorithms (e.g., Linear Regression and Decision Trees) to solve prediction tasks, followed by comparative analysis of their performance on test datasets.

Evaluation Method: Code and report evaluation focusing on implementation accuracy, performance comparison, and analysis depth; scored on a 10-point rubric.

Outcome: Apply unsupervised learning techniques for clustering and identify suitable machine learning approaches for specific application domains.

Activity: Perform clustering (e.g., K-Means, Hierarchical) on a given dataset and prepare a case study selecting and justifying machine learning methods suited for different application scenarios.

Evaluation Method: Lab practical combined with a written case study; assessed for correct

algorithm application, cluster interpretation, and justification of approach - evaluated on a 10-point rubric.

C14P: Machine Learning Lab

Lab Experiments:

1. Write a python program to import and export data using Pandas library functions.
2. Demonstrate various data pre-processing techniques for a given dataset
3. Implement Dimensionality reduction using the Principal Component Analysis (PCA) method.
4. Write a Python program to demonstrate various Data Visualization Techniques.
5. Implement MLE on a Dataset
6. Implement Simple and Multiple Linear Regression Models.
7. Develop Logistic Regression Model for a given dataset.
8. Develop Decision Tree Classification model for a given dataset and use it to classify a new sample.
9. Implement Naïve Bayes Classification in Python.
10. Develop K-Means for a Given Dataset
11. Build KNN Classification model for a given dataset.
12. Develop DBSCAN on a given Dataset

C15: Cloud Computing for Data Science:

Course Objectives

1. Introduce the fundamentals of cloud computing and its role in data science.
2. Provide understanding of virtualization, service, and deployment models.
3. Familiarize students with cloud storage, data management, and databases.
4. Expose students to cloud-based big data and machine learning platforms.
5. Train students in building, deploying, and monitoring ML pipelines on the cloud.

Course Outcomes

At the end of this course, students will be able to:

1. Explain cloud computing concepts including service models, deployment models, and virtualization.
2. Demonstrate cloud storage and database services for managing large-scale data.
3. Apply cloud-based platforms to run data science and machine learning workflows.
4. Build and deploy ML models on cloud services using AutoML and managed ML platforms.
5. Evaluate and monitor cloud-deployed solutions with respect to scalability, performance, and cost.

Unit 1. Introduction to Cloud Computing

Definition & Evolution of Cloud Computing, Service-Oriented Architecture (SOA) & Web Services, Utility & Grid Computing concepts, Characteristics of Cloud Computing

Cloud Computing Architecture: Front-end, Back-end, Networking, Delivery Models

Cloud Service Models: SaaS, PaaS, IaaS, Continuous Delivery using PaaS

Unit 2. Virtualization & Deployment Models

Concept & importance of Virtualization, Types of Virtualizations: Application, Network, Desktop, Storage, Server, Data Virtualization

Cloud Deployment Models: Public, Private, Community, Hybrid

Role of Cloud Computing in Data Science, Advantages of Cloud in Machine Learning

Unit 3. Cloud Storage & Data Management

Cloud Storage: Introduction, Benefits, Use Cases (Backup, Archiving, DR, Content Delivery)

Cloud Storage Systems: Block-based, File-based, Object-based storages

Key-Value Databases: Features & limitations

Batch vs. Streaming data for ML pipelines

Cloud Data Warehouses: AWS Redshift, Google BigQuery

Unit 4. Cloud Platforms for Data Science & ML

Machine Learning in the Cloud: Benefits & Limitations, Cloud-based ML Services: AIaaS, GPUaaS

Managed ML Platforms: Overview & advantages, Cloud ML Platforms: AWS SageMaker, Azure ML Studio, Google Cloud AutoML

Unit 5. Training & Deployment of ML on Cloud

Factors for selecting Cloud ML Platforms: ETL/ELT pipeline support, Scale-up/out training, ML frameworks, Pre-tuned services

Steps for Training ML Models in Cloud: Data source identification, Feature engineering, Training, Validation, Deployment, Monitoring, Monitoring & improving cloud-deployed ML models

Case studies & industry applications

Text / Reference Books

1. Handbook of Cloud Computing, Dr. Anand Nayyar, BPB Publications (2019)
2. Cloud Computing: A Practical Approach, Toby Velte, Anthony Velte, Robert C., McGraw Hill
3. Cloud Computing for Data Analysis, Noah Gift, Alfredo Deza, Pragmatic AI Labs
4. Data Science on the Google Cloud Platform, Valliappa Lakshmanan, O'Reilly
5. Machine Learning in the AWS Cloud: Amazon SageMaker, Abhishek Mishra, Wiley

Activities:

Outcome: Explain cloud computing concepts including service models, deployment models, and virtualization.

Activity: Prepare a comparative chart/report on cloud service and deployment models with real-world examples (AWS, Azure, GCP).

Evaluation Method: Report submission & viva (assess clarity, accuracy, and examples used).

Outcome: Demonstrate cloud storage and database services for managing large-scale data.

Activity: Perform lab experiments on block/file/object storage and execute queries in BigQuery / RDS.

Evaluation Method: Lab performance & practical exam (students demonstrate CRUD operations and explain storage use cases).

Outcome: Apply cloud-based platforms to run data science and machine learning workflows.

Activity: Implement a cloud-based ETL pipeline (e.g., using AWS Glue / Dataflow) for preparing a dataset.

Evaluation Method: Lab report + demo evaluation (workflow completeness, correctness of execution).

Outcome: Build and deploy ML models on cloud services using AutoML and managed ML platforms.

Activity: Train and deploy a classification/regression model using AWS SageMaker / Azure ML / Google AutoML.

Evaluation Method: Practical demo + oral viva (assess deployment success, prediction results, and understanding of pipeline).

Outcome: Evaluate and monitor cloud-deployed solutions with respect to scalability, performance, and cost.

Activity: Configure monitoring (e.g., CloudWatch, Stackdriver) for a deployed ML service and analyze resource usage.

Evaluation Method: Mini-project report & presentation (grading based on monitoring setup, analysis quality, cost optimization suggestions).

C15P: Cloud Computing for Data Science Lab:

1. Create Virtual Machine using VMware workstation for Windows/Linux
2. Install & configure WAMP/XAMPP/Apache on the VM and host a sample page
3. Install and configure a cloud account (AWS/Azure/GCP free tier).

4. Create and manage storage buckets; upload and access datasets.
5. Launch an instance and configure Block-based storage (EBS)
6. Create & configure File-based storage on cloud VM (EFS/Network FS).
7. Set up Jupyter Notebook/Colab on cloud VM.
8. Connect to cloud-hosted database services (AWS RDS, BigQuery, Cosmos DB).
9. Implement a batch ETL pipeline in the cloud.
10. Launch a SageMaker notebook, attach IAM role and S3 bucket, run sample notebook
11. Build a classification/regression model using AWS SageMaker / Azure ML Studio / GCP AI Platform.
12. Implement a simple ETL job: extract (RDS / CSV), transform, load into cloud DW (e.g., Redshift / BigQuery).
13. Use CloudWatch / Stackdriver to monitor endpoints, set alarms and auto-scale rules.
14. Use cloud AutoML services for dataset prediction tasks.
15. Deploy a trained ML model as a REST API endpoint in the cloud.