SEMESTER-III

COURSE 5: OOPS THROUGH JAVA

Theory **Credits: 3** **3 hrs/week**

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, 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.

SEMESTER-III

COURSE 5: OOPS THROUGH JAVA

Practical **Credits: 1** **2 hrs/week**

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 which having the following fields and button SAVE
    Form Fields are: Name, RNO, Mailid, Gender, Branch, Address.

SEMESTER-III

COURSE 6: DATABASE MANAGEMENT SYSTEMS

Theory **Credits: 3** **3 hrs/week**

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, relational algebra and calculus, 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, relational algebra, relational calculus, 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, relational algebra, relational calculus, and normalization techniques.

Activity: Normalize a given unstructured dataset up to 3NF. Then, write relational algebra expressions for sample queries.

Evaluation Method: Written assignment graded on:

- Correctness of normalization steps
- Accuracy of relational algebra expressions
- Short-answer questions on CODD rules and relational calculus

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

SEMESTER-III

COURSE 6: DATABASE MANAGEMENT SYSTEMS

Practical **Credits: 1** **2 hrs/week**

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	manager_id
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@cor	789-012-	2023-	IT	45000	2	104

			p.com	3456	04-15	Support			
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 EMP table on Saturday & Sunday.

SEMESTER-III

COURSE 7: COMPUTER ORGANIZATION

Theory **Credits: 3** **3 hrs/week**

Course Objectives

1. To introduce foundational concepts of register transfer language and micro-operations, enabling understanding of basic computer organization.
2. To examine CPU architecture and the control unit's design through hardwired and microprogrammed approaches.
3. To explore various memory organization strategies, including hierarchy, cache mapping, and associative techniques.
4. To understand input-output systems and data transfer mechanisms using I/O interfaces and DMA methods.
5. To analyze arithmetic algorithms and the principles of parallel and pipelined processing.

Course Outcomes

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

1. Interpret register-level operations and perform arithmetic, logic, and shift micro-operations within a basic computer framework.
2. Illustrate CPU functionality, addressing modes, and control unit design with both hardware-based and microprogramming techniques.
3. Categorize types of memory and explain memory hierarchy, access methods, and mapping techniques.
4. Describe I/O organization, including asynchronous transfer modes, DMA, and interrupt-driven processing.
5. Apply arithmetic algorithms and demonstrate understanding of parallel and pipelined processing models.

Unit 1. Register Transfer Language and Micro Operations:
Introduction- Functional units, computer registers, register transfer language, register transfer, bus and memory transfers, arithmetic, logic and shift micro-operations, arithmetic logic shift unit. Basic Computer Organization and Design: Instruction codes, instruction cycle. Register reference instructions, Memory – reference instructions, input – output and interrupt.

Unit 2. CPU and Micro Programmed Control:
Central Processing unit: Introduction, instruction formats, addressing modes.Control memory, address sequencing, design of control unit - hard wired control, micro programmed control.

Unit 3. Memory Organization:
Memory hierarchy, main memory, auxiliary memory, associative memory, cache Memory and mappings.

Unit 4. Input-Output Organization:
Peripheral Devices, input-output interface, asynchronous datatransfer, modes of transfer-programmed I/O, priority interrupt, direct memory access, Input – Output Processor (IOP).

Unit 5. Computer Arithmetic:
Data representation- fixed point, floating point, addition and subtraction, multiplication and division algorithms.

Text Books:

1. Computer Systems Architecture, M. Moris Mano, 3rd edition, Pearson/ PHI
2. Computer Organization and Architecture, William Stallings, 8th edition, Pearson/PHI

Reference Books:

1. Computer Organisation and Architecture, V. Rajaraman, T. Radha Krishnan, PHI
2. Computer Organization and Architecture Hamacher, Vranesic, Zaky, 5th edition, McGraw Hill

Activities:

Outcome: Interpret register-level operations and perform arithmetic, logic, and shift micro-operations within a basic computer framework.

Activity: Use a simulation tool (e.g., Logisim or Digital Works) to design a simple 4-bit ALU that performs:

- Addition, subtraction
- AND, OR
- Logical and arithmetic shifts

Evaluation Method: Demonstration-based assessment where students explain each operation using test inputs. Include a short worksheet with expected vs. actual outputs.

Outcome: Illustrate CPU functionality, addressing modes, and control unit design with both hardware-based and microprogramming techniques.

Activity: Create a flowchart or block diagram showing:

- CPU components (ALU, registers, control unit)
- Addressing modes (immediate, direct, indirect, etc.)
- Control unit types (hardwired vs. microprogrammed)

Evaluation Method:

Oral presentation or peer-reviewed poster session. Use a rubric to assess clarity, completeness, and correct identification of modes and control logic on 1 10-point scale.

Outcome: Categorize types of memory and explain memory hierarchy, access methods, and mapping techniques.

Activity: Build a memory hierarchy chart using coloured cards or digital tools. Include:

- Registers, cache, RAM, secondary storage
- Access methods (direct, associative, etc.)
- Mapping techniques (direct, associative, set-associative)

Evaluation Method: Quiz with matching and short-answer questions. Include a scenario-based question where students choose the best memory type for a given task.

Outcome: Describe I/O organization, including asynchronous transfer modes, DMA, and interrupt-driven processing.

Activity: Role-play simulation: Assign students' roles (CPU, DMA controller, I/O device) and simulate data transfer using cards or tokens to represent data and control signals.

Evaluation Method: Reflective worksheet where students describe the flow of control and data in each mode. Include a diagram labelling key signals and steps.

Outcome: Apply arithmetic algorithms and demonstrate floating point arithmetic operations.

Activity: Use a spreadsheet or visual tool to simulate floating point arithmetic operations.

Evaluation Method: Submission of simulation results with explanation. Evaluate the report on a 10-point scale.

SEMESTER-III

COURSE 7: COMPUTER ORGANIZATION

Practical **Credits: 1** **2 hrs/week**

List of Experiments:

1. Simulate Register Transfer Using Logisim
2. Build a microprogrammed control unit using Logisim or VHDL (GHDL).
3. Simple Register Transfers and Arithmetic Operations using EMU8086 Simulator
4. Simulate a Simple Instruction Cycle in Emu8086
5. Demonstrate Addressing Modes (Immediate, Register, Direct, Indexed) in Emu8086
6. Implement Subroutines Using CALL and RET in Emu8086
7. Simulate DMA Transfer in Ripes or Logisim
8. Simulate Booth's Algorithm for multiplication and restoring division in Python/C or Logisim
9. Write assembly language code for A+B*(C-D) using various instruction formats in any open-source assembler.
10. Write assembly language code for A+B*C using various addressing modes in any open-source assembler.