UNIVERSITY INSTITUTE OF ENGINEERING (UIE)

Department of Career Planning and Development
B.E. – Computer Science and Engineering

Course File on
Subject Name – Advanced Database Management System

Subject Code – NEW CODE
Semester VI
Even Semester – Jan-May 2025

Syllabus

Program Code - CS201	Course Title -Advanced Database Management System	L	T	P	C	CH	Course Type
CourseCode- NEW CODE		0	0	2	0	2	-

a. Course Description

The **Advanced Database Management Systems (DBMS)** course is designed to equip students with in-depth knowledge and practical skills in database design, optimization, and management. The course covers essential concepts such as database architecture, schema design, ER modeling, and relational algebra. It explores advanced topics including functional dependencies, normalization, multi-valued dependencies, and database security. Students will gain proficiency in SQL, focusing on complex queries, joins, subqueries, window functions, CTEs, and performance optimization techniques. The course also covers transaction processing, concurrency control, and recovery methods to ensure data consistency and reliability. Through hands-on practice, students will develop expertise in handling real-world database challenges, preparing them for successful placements in the IT industry.

b. Course Objectives

1. **Understand Database Concepts:** To provide students with a comprehensive understanding of database fundamentals, including database architecture, schema design, and relational algebra.
2. **Master SQL Queries:** To equip students with the skills to write complex SQL queries using SELECT, JOINs, subqueries, window functions, CTEs, and aggregate functions to efficiently retrieve and manipulate data.
3. **Learn Normalization and Design Principles:** To develop an in-depth understanding of functional dependencies, normalization techniques (1NF to BCNF), multi-valued dependencies (4NF), and join dependencies (5NF) to design optimal and efficient database schemas.
4. **Implement Advanced Features:** To enable students to implement advanced DBMS features such as triggers, stored procedures and transaction management in SQL databases.
5. **Understand Transaction Management:** To provide students with knowledge of transaction processing, ACID properties, concurrency control mechanisms, and database recovery techniques to ensure database consistency and reliability.
6. **Prepare for Industry Challenges:** To equip students with the practical skills and theoretical knowledge required to tackle complex database management challenges in real-world scenarios and prepare them for industry placements.

c. Course Outcomes

Course Outcome (CO)	Description
CO1	Understand and explain the fundamentals of database concepts, architecture,

	schema design, and ER modeling.
CO2	Develop advanced SQL skills, including the use of complex queries, joins, subqueries, window functions, and CTEs.
CO3	Analyze and apply normalization techniques (1NF, 2NF, 3NF, BCNF), functional dependencies, and decomposition strategies.
CO4	Implement database features like triggers, stored procedures and transaction management to optimize performance.
CO5	Apply transaction processing, concurrency control mechanisms, and database recovery techniques to ensure data consistency and reliability in a multi-user environment.

d. Syllabus **30 Sessions**

Unit	Topics	Subtopics
Unit I: SQL Revision: From Basics to Queries	Introduction to SQL	- Characteristics of SQL - SQL Data Types and Literals - Types of SQL Commands: DDL, DML, DCL, TCL - Basic Commands: CREATE, DROP, TRUNCATE, INSERT, UPDATE, DELETE, SELECT
	Advanced SQL Operations	- Filtering and Sorting: WHERE, LIMIT, ORDER BY - Aggregating Data: GROUP BY, HAVING - SQL Operators: IN, ANY, EXISTS - Aggregate Functions: COUNT, SUM, MIN, MAX, AVG
	Joins, Sub-queries, and Views	- Joins: Inner Join, Outer Join, Self Join - Set Operations: UNION, INTERSECTION - Sub-queries: Nested Queries (SELECT, FROM, WHERE) - Views: Creation, Manipulation, and Benefits
Unit II: Advanced SQL Querying	Window Functions	- Introduction to Window Functions - Using PARTITION BY for Windowing - Common Window Functions: ROW_NUMBER(), RANK(), DENSE_RANK(), NTILE(), LEAD(), LAG()
	Common Table Expressions (CTEs)	- Introduction to CTEs - Syntax and Benefits of CTEs - Recursive CTEs - Practical Use Cases of CTEs in Query Optimization

Unit III: Advanced DBMS Concepts and Transaction Management	Procedures, Functions, and Triggers	- Procedures: Syntax, Parameter Modes (IN, OUT, IN-OUT), Advantages - Functions: Differences from Procedures - Triggers: Need, Syntax, Types
	Transaction Management	- ACID Properties of Transactions - Transaction Isolation Levels - Schedules: Serializability, Conflict/View Serializable Schedules - Recoverability and Deadlock Handling
	Concurrency Control and Recovery	- Concurrency Control: Problems (Dirty Read, Lost Update) - Lock-Based Protocols (2PL), Timestamp-Based Protocols - Recovery: Transaction Failures, Log-Based Recovery, Checkpoints
	Hands-On Practical Exercises	- SQL Commands: Aggregations, WHERE, LIMIT, ORDER BY, GROUP BY, HAVING - Joins, Sub-queries (SELECT, FROM, WHERE) - Views Creation - Cascading Operations for Foreign Keys

Lecture Plan: **15W*2S=30S**

Sr. No.	Topic	Objective of the Session	Focused Technical Skill	Practical Activity / Hands-on Exercise	Duration (50 min each)	Parameters of Evaluation	Session Outcome	Follow-up Assignment / Practice Task
1	Introduction to SQL	Understand SQL basics and types of SQL commands	SQL Data Types, SQL Commands (DDL, DML, DCL, TCL)	Hands-on coding with queries in SQL (MySQL/SSMS)	1	Accuracy in basic queries and understanding of SQL syntax	Knowledge of SQL basics, data types, and commands	Create simple queries using DDL and DML commands
2	Basic SQL Commands	Learn the use of fundamental SQL commands (CREATE, DROP, INSERT, UPDATE, DELETE)	CRUD Operations (Create, Read, Update, Delete)	Practice with sample tables in SQL IDE (MySQL/SSMS)	1	Successful execution of basic SQL commands, correct syntax	Ability to create and manipulate tables and data using SQL commands	Write queries to insert, update, and delete data from tables

3	Filtering and Sorting	Learn to filter and sort data using SQL commands	WHERE, LIMIT, ORDER BY	Hands-on exercises to filter and sort data	1	Correct usage of WHERE, LIMIT, ORDER BY clauses	Ability to filter and sort data effectively	Write queries using WHERE and ORDER BY for sorting and filtering
4	Advanced Filtering and Sorting	Master advanced filtering techniques using AND, OR, NOT operators	Advanced Filtering (AND, OR, NOT), Sorting	Practice advanced filters and sorting with real-life data scenarios	1	Complexity of queries and accuracy of results	Ability to write complex queries for data filtering and sorting	Use AND, OR, NOT operators to filter data in complex queries
5	Aggregating Data	Learn to aggregate data using GROUP BY and HAVING, and functions like COUNT, SUM	Aggregating Functions (COUNT, SUM, MIN, MAX, AVG)	Hands-on exercises with aggregation functions and GROUP BY/HAVING clauses	1	Correct use of aggregation functions and GROUP BY clause	Ability to perform data aggregation and grouping	Write aggregate queries to calculate SUM, AVG, COUNT, etc.
6	SQL Operators	Master SQL operators for filtering and comparing values (IN, ANY, EXISTS, LIKE)	Operators (IN, ANY, EXISTS, LIKE)	Practice writing queries using various operators for filtering	1	Correct usage of operators for filtering and comparison	Ability to filter and compare data using SQL operators	Use operators like IN, LIKE, EXISTS in query examples
7	Joins	Understand and implement SQL joins to combine data from multiple tables	Joins (Inner Join, Left Join, Right Join, Full Outer Join)	Writing and debugging SQL queries using joins	1	Ability to combine data using different types of joins	Ability to use joins for combining data from multiple tables	Write queries using various joins (INNER, LEFT, RIGHT, FULL)
8	Self Join & Set Operations	Learn self joins and set operations (UNION, INTERSECT)	Self Join, Set Operations (UNION, INTERSEC T)	Practice using self joins and combining results with set operations	1	Correct usage of self join and set operations in queries	Ability to perform self joins and set operations (UNION, INTERSEC T)	Write queries using self join and UNION for data combination
9	Sub-queries	Learn to write and optimize sub-queries for complex data retrieval	Sub-queries (Nested Queries)	Hands-on practice with writing and optimizing	1	Correct nesting and optimization of queries	Ability to write and optimize nested queries for	Write nested queries for complex data

				sub-queries			data retrieval	retrieval
10	Views	Learn to create and manage views for query optimization	View Creation and Management	Practice creating and managing views to simplify queries and improve performance	1	Correct creation and manipulation of views	Ability to use views for query optimization	Create views to optimize query execution
11	Window Functions	Understand window functions and how to use PARTITION BY for advanced analysis	Window Functions (ROW_NUMBER, RANK, etc.)	Hands-on practice using window functions for ranking and analysis	1	Successful implementation of window functions with PARTITION BY	Ability to use window functions for advanced data analysis	Implement window functions like ROW_NUMBER, RANK, etc.
12	Common Window Functions	Learn common window functions like ROW_NUMBER(), RANK(), LEAD(), LAG()	Ranking Functions (ROW_NUMBER, RANK, LEAD, LAG)	Practice using common window functions for ranking and analysis	1	Ability to use ranking functions to analyze data	Mastery in ranking and windowing operations using SQL	Write queries using RANK, ROW_NUMBER, LEAD, LAG functions
13	Common Table Expressions (CTEs)	Learn the syntax and benefits of Common Table Expressions (CTEs)	CTE Syntax and Recursive Queries	Write and practice queries using CTEs for data organization and modularity	1	Ability to implement and optimize queries using CTEs	Ability to write complex queries with CTEs	Create recursive CTEs and optimize queries
14	CTEs in Query Optimization	Understand the use of CTEs for query optimization	CTE Optimization	Hands-on exercises in optimizing queries using CTEs	1	Correct application of CTEs for query optimization	Ability to optimize queries using CTEs	Optimize queries using CTEs in SQL
15	PLSQL Introduction	Understand the basics of PLSQL, including variables, data types, and constants	PLSQL Variables, Data Types, Constants	Hands-on exercises in Oracle SQL Developer or similar tools to practice PLSQL basics	1	Correct declaration and usage of variables and constants in PLSQL	Understanding of basic PLSQL syntax and variable declaration	Practice declaring variables and constants in PLSQL

16	PLSQL Control Structures	Learn control flow constructs in PLSQL (IF-ELSE, CASE, FOR, WHILE)	Control Flow (IF-ELSE, CASE, Loops)	Writing and practicing PLSQL blocks with control structures	1	Correct implementation of control flow structures in PLSQL	Ability to write complex control structures in PLSQL	Write PLSQL blocks with IF-ELSE, CASE, and loops
17	PLSQL Cursors	Understand and implement implicit and explicit cursors in PLSQL	Implicit and Explicit Cursors	Practice using cursors to fetch multiple rows and process them efficiently	1	Correct implementation of cursors in PLSQL	Ability to handle multiple rows using cursors	Implement cursors in real-world scenarios
18	PLSQL Procedures	Learn to write stored procedures with parameters in PLSQL	Writing Stored Procedures (IN, OUT, IN-OUT)	Hands-on practice writing procedures to modularize SQL logic	1	Correct creation and implementation of procedures with parameters	Ability to create and use stored procedures in PLSQL	Write stored procedures with different parameter modes
19	PLSQL Functions	Learn the difference between functions and procedures and how to create functions	PLSQL Functions	Practice writing and testing PLSQL functions for reusable logic	1	Successful creation and testing of PLSQL functions	Mastery in creating PLSQL functions for reusable logic	Create functions in PLSQL for various tasks
20	PLSQL Triggers	Learn to write triggers for automatic actions in databases	Writing Triggers	Hands-on exercises in writing and implementing triggers in PLSQL	1	Correct syntax and functionality of triggers	Ability to write triggers to automate database actions	Write triggers for data changes or events
21	Transaction Management	Understand ACID properties and transaction management in SQL/PLSQL	Transaction Management (ACID properties)	Practical exercises in managing transactions and ensuring data integrity	1	Correct use of transaction commands and understanding of ACID properties	Understanding of transaction management and ACID principles	Practice handling transactions in SQL/PLSQL
22	Transaction Isolation Levels	Master transaction isolation levels and their effects	Read Uncommitted, Read Committed, Repeatable Read, Serializable	Hands-on with isolation levels in PLSQL	1	Correct understanding of isolation levels	Understanding transaction isolation and consistency in SQL	Practice managing transactions with different isolation levels

23	Transaction Schedules & Deadlocks	Learn to handle transaction scheduling and deadlocks	Serializability, Conflict/View Serializable Schedules, Deadlocks	Case studies on transaction scheduling and deadlock resolution	1	Correct scheduling and deadlock management	Ability to manage transaction scheduling and handle deadlocks	Practice resolving deadlocks in transaction scenarios
24	Transaction Recovery	Learn log-based recovery methods and checkpointing techniques	Log-Based Recovery, Checkpoints	Hands-on with recovery operations	1	Correct transaction recovery handling	Understanding recovery techniques for transaction failures	Practice implementing recovery strategies
25	PLSQL Exception Handling	Learn to handle exceptions and errors in PLSQL	Exception Handling, User-Defined Exceptions	Writing error-handling procedures in PLSQL	1	Correct handling of exceptions in PLSQL	Ability to implement exception handling in PLSQL	Practice writing exception handling blocks in PLSQL
26	Advanced PLSQL: Nested Blocks	Learn to write nested procedures, functions, and triggers	Nested Procedures, Functions, Triggers	Practice with nested blocks in PLSQL	1	Correct creation and use of nested blocks	Ability to write modular, reusable code with nested blocks	Practice writing nested procedures and triggers
27	Advanced Query Optimization	Learn indexing strategies and query execution plans	Indexing, Query Execution Plans	Query optimization exercises	1	Correct use of indexing and optimization methods	Understanding advanced query optimization techniques	Practice optimizing queries with indexes
28	Complex SQL and PLSQL Operations	Master complex SQL and PLSQL queries for advanced data retrieval	Complex Joins, Subqueries, Window Functions	Case studies and exercises with complex queries	1	Correct implementation of advanced operations	Mastery of complex SQL and PLSQL operations	Practice writing advanced SQL and PLSQL queries
29	Hands-On PLSQL Projects	Build real-world projects using SQL and PLSQL	SQL, PLSQL Procedures, Functions, Triggers, Transactions	Implementing SQL and PLSQL in a project	2	Quality of project implementation	Experience with real-world SQL and PLSQL applications	Complete project and present solutions
30	Final Exam & Assessment	Assess overall knowledge of SQL and PLSQL	Troubleshooting SQL, PLSQL Queries	Hands-on project presentations, online assessment	2	Performance in practical test, project debugging	Demonstrated mastery of SQL and PLSQL concepts	Review errors from the exam and practice key areas of weakness

Assessment Plan:

Sr. No.	**Assessment Component**	**Marks**	**Frequency**	**Total Weightage**
1	Assignments	20 marks each	2 assignments	25 marks
2	Quizzes	20 marks each	2 quizzes	25 marks
3	Final Assessment Test	50 marks	1 test	50 marks

Problem Statements :

Sample Database Tables

- Employees(EmployeeID, FirstName, LastName, DepartmentID, Salary, HireDate, ManagerID, Status)
- Departments(DepartmentID, DepartmentName, Budget)
- Projects(ProjectID, ProjectName, DepartmentID, StartDate, EndDate, Budget)
- EmployeeProjects(EmployeeID, ProjectID, Role)

- Salaries(EmployeeID, Salary, StartDate, EndDate)
- Attendance(EmployeeID, Date, Status)

Sample Table Schema- https://codeshare.io/kargil_dbms

1. **Employees Table**

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    DepartmentID INT,
    Salary DECIMAL(10, 2),
    HireDate DATE,
    ManagerID INT,
    Status VARCHAR(20)
);
```

2. **Departments Table**

```
CREATE TABLE Departments (
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50),
    Budget DECIMAL(10, 2)
);
```

3. **Projects Table**

```
CREATE TABLE Projects (
    ProjectID INT PRIMARY KEY,
    ProjectName VARCHAR(100),
    DepartmentID INT,
    StartDate DATE,
    EndDate DATE,
    Budget DECIMAL(10, 2)
);
```

4. **EmployeeProjects Table (Many-to-Many Relationship)**

```
CREATE TABLE EmployeeProjects (
    EmployeeID INT,
    ProjectID INT,
    Role VARCHAR(50),
    FOREIGN KEY (EmployeeID) REFERENCES
Employees(EmployeeID),
    FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID)
);
```

5. **Salaries Table**

```
CREATE TABLE Salaries (
    EmployeeID INT,
    Salary DECIMAL(10, 2),
    StartDate DATE,
    EndDate DATE,
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

6. **Attendance Table**

```
CREATE TABLE Attendance (
    EmployeeID INT,
    Date DATE,
    Status VARCHAR(20),  -- 'Present', 'Absent', 'Leave'
    FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

100 Problem Statements (Scenario-Based)

Basic SQL Queries

1. Retrieve the names of employees who are assigned to more than two projects.
2. List the names of employees who are not assigned to any project, along with their department names.
3. Find the employees whose hire date falls on a weekend (Saturday or Sunday).
4. Display the names of employees who have been working for more than 5 years and are still active.
5. Retrieve the details of projects that are associated with departments having a budget below 100,000.
6. Retrieve all employee details along with their department names and project roles, if any.
7. Find the first and last names of employees who work in the 'IT' department and have a salary above the department's average salary.
8. List the names and hire dates of employees hired in 2023, including their department names and managers' names.
9. Get the department names, total budget, and the number of employees in each department.
10. List the employees whose salaries are in the top 10% of all salaries in the organization.

Filtering Data

1. Find employees whose salaries are in the top 25% of their department.
2. Retrieve the names of employees who have the word "Developer" in their role in any project.
3. Get the details of projects where no employees have been assigned yet.

4. List the employees who have a salary less than the average salary of all employees in their department.
5. Retrieve the names of employees who joined in the last 6 months and are not assigned to any projects.
6. Find employees whose salary is between 60,000 and 120,000, and who are assigned to more than one project.
7. Retrieve active employees hired before 2022 who are working in departments with a budget above 80,000.
8. Get employees working in 'HR' or 'Finance' departments who are assigned to projects that started in 2022 or later.
9. List employees who have not been assigned any projects and whose salaries are below the organization's average salary.
10. Display employees whose last name starts with 'J' and who were hired in or after 2020.

Sorting Data

1. Sort the projects by their budget in descending order and display the top 3 highest-budget projects.
2. Sort employees by their last name alphabetically but display those with a status of 'Active' first.
3. Sort departments by the number of employees in descending order.
4. Sort employees by their salary in descending order, and for employees with the same salary, sort them by their hire date in ascending order.
5. Sort departments by their budget in ascending order and include the number of employees in each department.
6. Sort employees by hire date, but group them first by their department name.
7. Sort projects by their end date, listing ongoing projects first, and for ongoing projects, sort by their start date.
8. Sort employees by their first name in alphabetical order and include their project roles, if any.

Aggregating Data

1. Find the total number of projects each employee is working on, and list employees with more than 2 projects.
2. Calculate the average salary of employees in each department and list departments with an average salary greater than 6,000.
3. Find the maximum, minimum, and average project budget for each department.
4. Retrieve the total salary expenditure for all employees grouped by their department.
5. Count the number of employees hired in each year.
6. Find the average salary of employees in each department and compare it to the overall average salary in the organization.
7. Get the total salary expenditure for each department, including employees with multiple salary records.

8. Count the number of employees in each department who are assigned to at least one project.
9. Find the maximum and minimum salary in each department, along with the names of the employees earning those salaries.
10. Calculate the total budget allocated to projects in the 'IT' department and compare it with the department's overall budget.

Joins

1. Retrieve the names of employees along with their project names, even if they are not assigned to any project.
2. List the projects along with the total number of employees assigned to each project.
3. Find the names of employees working on projects managed by employees from the same department.
4. Retrieve the details of employees and their managers, including employees who don’t have a manager.
5. Display the projects that have employees from more than one department assigned to them
6. List all employees along with their department names, project roles, and the names of the projects they are assigned to.
7. Get the names of employees working on projects with a name containing 'Alpha' or 'Beta'.
8. Find all employees managed by the employee with ID 101, including their department and project details.
9. Display the projects assigned to employees along with their roles, and include employees not assigned to any projects.
10. Show the details of employees working on projects with a budget greater than the department's average budget.

Subqueries

1. Find the employees whose salary is higher than the average salary of all employees in the organization.
2. List the projects that have a budget greater than the total budget of the 'Marketing' department.
3. Retrieve the names of employees who are not assigned to any projects, using a subquery.
4. Get the departments where the number of employees is less than the number of projects.
5. List employees whose salaries are higher than the average salary of their managers.
6. List employees whose salaries are above the average salary of their respective departments, and include their department names.
7. Find employees working in departments with a budget greater than the organization's average department budget.
8. Get employees who are not assigned to any projects using a subquery, and include their department names.
9. Find employees earning more than the average salary in the 'Finance' department and include their project roles, if any.

10. List employees who were hired after the hire date of the highest-earning employee in their department.

Set Operations

1. Get employees who are either managers or are working on projects (use UNION).
2. List departments that have no projects assigned (use EXCEPT).
3. Find the employees who are working on both 'Project A' and 'Project B' (use INTERSECT).
4. Retrieve employees who are working in either the 'HR' or 'Finance' departments (use UNION).
5. Display all project names that do not involve 'Employee 101' (use EXCEPT).

Window Functions

6. Rank the employees in each department by their salary, with the highest salary ranked first.
7. Find the cumulative budget of all projects sorted by their start date.
8. Retrieve the running total of salaries for all employees ordered by hire date.
9. List employees with their rank based on hire date (earliest hired gets rank 1).
10. Calculate the difference between each employee's salary and the average salary of their department.
11. Assign a rank to each employee based on their salary within each department (use RANK()).
12. Find the top 5 highest paid employees in the company (use ROW_NUMBER()).
13. Get the salary difference between each employee and the one with the highest salary (use LEAD()).
14. List employees and their salaries compared to the average salary in their department (use LAG()).
15. Rank employees within each department based on their salary and display the rank (use DENSE_RANK()).

Real-World Scenarios

1. Find employees who are eligible for a bonus (hired more than 2 years ago and have worked on at least one project).
2. Retrieve the details of projects that have not been completed and have exceeded their budget by more than 10%.
3. Identify employees who have been consistently marked 'Absent' for more than 3 consecutive days in the attendance table.
4. List the departments with employees who have been assigned to projects in other departments.
5. Display employees who have the highest salary in their respective departments.

Common Table Expressions (CTEs)

16. Use a CTE to list all employees and their departments.
17. Create a recursive CTE to display the hierarchy of employees and their managers.
18. Use a CTE to list the total salary expenditure for each department.

19. Write a CTE to find employees working on multiple projects in the same department.
20. Use a CTE to list employees whose salaries have changed over time.

Transactions

21. Write a transaction that updates the salary of an employee and commits the changes.
22. Create a transaction that transfers a project from one department to another.
23. Write a transaction that adds an employee to a project and rolls back if there is an error.
24. Create a transaction that deletes a department and its employees (use CASCADE DELETE).
25. Implement a transaction that updates multiple tables (e.g., Salary and Attendance).

Foreign Keys and Constraints

51. Add a foreign key constraint between 'Employees' and 'Departments'.
52. Add a unique constraint on the 'EmployeeID' column in the 'Salaries' table.
53. Create a foreign key between 'EmployeeProjects' and 'Projects' to ensure referential integrity.
54. Write a query to display the records that violate foreign key constraints.
55. Alter the 'Employees' table to add a foreign key constraint to the 'ManagerID' column.

Indexing

56. Create an index on the 'DepartmentID' column in the 'Employees' table.
57. Create a composite index on 'ProjectID' and 'EmployeeID' in the 'EmployeeProjects' table.
58. Optimize the query performance for the 'Employees' table using indexing.
59. Create an index on the 'Salary' column in the 'Employees' table for faster salary-related queries.
60. Identify slow queries and suggest indexing strategies for optimization.

Placement-Oriented SQL Questions

61. Write a query to get the second highest salary from the 'Employees' table.
62. Find the employees who have a salary greater than the average salary of their department.
63. Write a query to find all employees who do not have any subordinates (no employees under their management).
64. Write a query to find the nth highest salary (use ROW_NUMBER or a subquery).
65. Write a query to find the most recent hire in each department.
66. Write a query to calculate the running total of salaries for employees.
67. Find the employee with the highest number of projects assigned (using GROUP BY).
68. Write a query to display the employees with the highest salaries in each department (using RANK()).
69. Create a view to show employees with their total salary and total number of projects.
70. Write a query to find employees whose salary is among the top 10% highest salaries.
71. Write a query to find employees who have been working for more than 5 years in the company.
72. Write a query to list employees who have worked on more than one project.
73. Write a query to find employees with the second highest salary in the company.
74. Write a query to list the top 3 departments based on the total salary of employees.
75. Write a query to find employees whose hire date is before the company’s founding date.
76. Write a query to find the total number of employees in each department.
77. Write a query to find employees who are working in a project with a budget more than $500,000.
78. Write a query to find the employees whose salary is greater than the average salary of their department.

79. Write a query to find employees who have worked in more than one department.
80. Write a query to find employees who have the highest salary in their respective departments.
81. Write a query to find the number of projects handled by each department.
82. Write a query to find the top 5 highest-paid employees.
83. Write a query to list employees who are working in the same project for more than 2 years.
84. Write a query to find the employee who has worked in the company for the longest period.
85. Write a query to calculate the average salary of employees who have worked in both 'Project X' and 'Project Y'.
86. Write a query to list employees who received a salary increment in the last year.
87. Write a query to find the department that pays the most and the least in total salary.
88. Write a query to find employees who do not have a manager (NULL value in the manager_id column).
89. Write a query to find employees who have not worked on any project.
90. Write a query to find the total salary for each department, including the number of employees in that department.
91. Write a query to retrieve employees whose salary is more than the highest salary of their department.
92. Write a query to find the number of employees who have worked for the company for more than 3 years and have a salary less than the average company salary.
93. Write a query to find the employees who were hired in the same year as the employee with employee_id 102.
94. Write a query to find the department with the lowest total salary expenditure.
95. Write a query to find employees who have worked on multiple projects and calculate the number of projects they have worked on.
96. Write a query to display the names of employees who work in the 'Engineering' department but are not part of the 'Project Z' team.
97. Write a query to find the department with the most number of employees earning more than $80,000.
98. Write a query to list employees who earn more than the average salary of all employees in the company.
99. Write a query to retrieve the top 10 employees with the highest salary in each department.
100. Write a query to find the employees who received a salary increment greater than 10% in the past year.

Resources:

- **Textbooks**:
  - *Database System Concepts* by Abraham Silberschatz, Henry F. Korth, S. Sudarshan
  - *Fundamentals of Database Systems* by Ramez Elmasri, Shamkant B. Navathe
  - *SQL Performance Explained* by Markus Winand
  - *Database Management Systems* by Raghu Ramakrishnan, Johannes Gehrke
- **Online Resources**:
  - W3Schools SQL Tutorial (https://www.w3schools.com/sql/)
  - SQLZoo (Interactive SQL Tutorials) (https://sqlzoo.net/)
  - TutorialsPoint DBMS Guide (https://www.tutorialspoint.com/dbms/index.htm)

    - GeeksforGeeks SQL Articles (https://www.geeksforgeeks.org/sql/)
- **Software Tools**:
    - SSMS
    - MySQL Workbench
    - Oracle SQL Developer

Learning Outcomes:

Proficiency in SQL and Database Querying:

- Demonstrate expertise in writing and optimizing SQL queries, including advanced operations like Joins, Sub-queries, and Views, to handle complex data scenarios commonly encountered in job roles.

Hands-On Database Management Skills:

- Gain practical experience in managing relational databases, performing data manipulation, and ensuring data integrity, preparing for roles such as Database Developer or Data Analyst.

Transaction and Concurrency Management:

- Apply ACID principles and concurrency control techniques to ensure reliable and scalable database operations, a key skill for enterprise-level database management roles.

Advanced Data Design and Optimization:

- Design normalized, efficient database schemas and optimize query performance using indexing and partitioning, aligning with the expectations of back-end development and system design positions.