School of Computer Engineering
KIIT deemed to be University
Data Base Management System Laboratory Lesson Plan
Spring 2024 - 2025 (4th Semester)
CSE

Discipline: B.Tech (All branches of School of Computer Engineering)

Course name and Code: Data Base Management System Laboratory (CS29006)

L-T-P-Cr **:** 0-0-2-1

Faculty Name: Dr. Priyanka Panigrahi, Email: priyanka.panigrahifcs@kiit.ac.in

Faculty Chamber: F-221, Block – A, Campus-15

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

CO1: Understand various queries execution such as relational constraints, joins, set operations, aggregate functions, trigger, views and embedded SQL.
CO2: Apply DDL, DML, DQL, DCL, and TCL commands to practice the designing, developing and querying a database.
CO3: Analyse integrity constraints on a database design
CO4: Evaluate query using SQL commands as solutions to a broad range of query and data update problems.
CO5: Create stored functions, stored procedures, cursor, trigger, and exception handling using PL/SQL block.
CO6: Create and design database systems.

Technical Assistants Names:

- TA 1
- TA 2

Course Contents:

The course focuses on basic and essential topics in data structures and algorithms, including:

Lab No.	Coverage Area
1	• An introduction to SQL • Introduction to database management systems • Overview of some basic terms and concepts
2	• Integrity Constraints • DDL • DML
3	**DQL – Part 1** • Recap on DML • SELECT, FROM, WHERE & ORDER BY Clause • Counting records in a Table • Column Aliases • Operators • Summarization
4	**DQL – Part 2** • Grouping • Character Function • Conversion Function • Mathematical Function • Date Function
5	**DQL – Part 3** • Table Joining **DDL Recap**

	• Auto Increment • Select Into clause • Creation of table from existing table
6	**DQL – Part 4** • Union, Union All, Intersect, and Minus Operator • Select Top Operator • Subquery
7	• Creating and Using Views • Creating and Using Synonyms • Using Indexes to improve performance **DCL** • Grant • Revoke **TCL** • Begin • Commit • Rollback
8	**PL/SQL Programming Language – Part 1** • Notable Facts • Features • Basic Syntax • Conditions • Loops
9	**PL/SQL Programming Language – Part 2** • Procedure • Function
10	**PL/SQL Programming Language – Part 3** • Exception • Cursor • Trigger

List of Experiments (Day wise):

Day	***Aim of the experiment***
Day 1	**1.1.** Create individual tables for Employee, Student, Product, Customer and Account. Identify relevant attributes for each table and link to respective tables. Ensure each table has a _ID column e.g. Employee should have EMPLOYEE_ID column, Student should have STUDENT_ID column etc. **Sample Input:** NA **Sample Output:** Tables are created.
	1.2. Describe each table. **Sample Input:** NA **Sample Output:** The output should display each table capturing column definitions.
	1.3. Insert at least 5 distinct rows to each table. **Sample Input:** NA **Sample Output:** 1 row(s) inserted for the desired table.
	1.4. Fetch all data from the respective tables. **Sample Input:** NA **Sample Output:** 5 rows returned from the respective table.

	1.5. Drop each table. **Sample Input:** NA **Sample Output:** Respective table has been deleted.
Day 2	**2.1.** Using appropriate integrity constraints, design & create suitable tables to store data for the following: • Restaurant Booking System involving Customer/Booking/Table/Order/Menu/Ingredients/Staff/ Billing • Hospitality Management System involving Patient/Doctor/Disease/Hospital/Billing • Car Service Center System involving Car/Customer/Booking/Mechanic/Billing • Banking System involving Branch/Customer/Account/Employee/Transaction/Product **Sample Input:** NA **Sample Output:** Tables are created.
	2.2. Add valid & suitable data (minimum 5 records) to all tables using INSERT, UPDATE and DELETE (if required) statement. **Sample Input:** NA **Sample Output:** Depends individual's own way of data insertion into respective table. By default: 1 row(s) inserted for the desired table.

Day 3	**3.1.** Create following tables and insert appropriate rows • EMPLOYEE table with the attributes: ID, LAST_NAME, FIRST_NAME, MIDDLE_NAME, FATHER_NAME, MOTHER_NAME, SEX, HIRE_DATE, ADDRESS, CITY, STATE, ZIP, PHONE, PAGER • SCHOOL table with the attributes: ID, NAME • EMPLOYEE _ALIGNMENT table with the attributes: EMPLOYEE_ID, SCHOOL_ID • JOB table with the attributes: ID, NAME, TITLE, SALARY, BONUS • EMPLOYEE _PAY table with the attributes: EMPLOYEE_ID, JOB_ID • PRODUCT table with the attributes: ID, name, desc, status, created_date, created_by, last_modified_date, last_modified_by, cost, injected_date **Sample Input:** NA **Sample Output:** Tables are created. 1 row(s) are inserted to respective table.
	3.2. Retrieve the row count in “Associate Professor” title. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	3.3. Retrieve the row count in “Associate Professor” title whose salary is between 80,000 and 100,000. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.

	3.4. Retrieve the details of jobs whose sum of salary and bonus is falling in the range 80,000 and 100,000. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	3.5. What is the average salary? **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	3.6. What is the maximum bonus? **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	3.7. Write a SELECT statement that returns product IDs and description (alpha order) for the product, created by Indiana D or Ohio D or Michigan R, or Illinois M. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.

Day 4	**4.1.** Refer the tables created in 3.1. **Sample Input:** NA **Sample Output:** Tables are created. 1 row(s) are inserted to respective table.
	4.2. Find the maximum cost, minimum cost by each Product Creator. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	4.3. Find the sum of cost, average cost by each Last Modifier. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	4.4. Find the number of products created by Indiana D or Ohio D. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.

	4.5. Find the number of products updated by Michigan R or Illinois M. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	4.6. Display the sum of all salary. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	4.7. Compute the average salary value across all non-null bonus values **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	4.8. Find the average, highest and lowest cost for the products. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.

	4.9. Display the total number of jobs for null bonus values. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	4.10. Display the last modified and created date year, month value for each product. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	4.11. Display the JOB name and title followed by a message based on salary. If the salary is > 1,00,000 display “High”. If the salary in between 50,000 and 100,000 then display “Medium’ else display ‘Low’. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	4.12. Display the concatenation of product name, description, created date and last updated date for active products. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.

Day 5	**5.1.** Refer the tables created in 3.1. **Sample Input:** NA **Sample Output:** Tables are created. 1 row(s) are inserted to respective table.
	5.2. Display all employees' full name and their school names. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	5.3. Display all employees' full name and their job title and salary. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	5.4. Display all employees' full name with their job name, title and total salary for non-null bonus. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.

5.5. Find the full name of the supervisors.

Sample Input:
NA

Sample Output:
Display the row details as per the specification.

5.6. Find the name, ID and number of supervisee for each supervisor.

Sample Input:
NA

Sample Output:
Display the row details as per the specification.

5.7. For each school, find the name, ID and number of supervisee for each supervisor.

Sample Input:
NA

Sample Output:
Display the row details as per the specification.

5.8. Find the employees who are working as “Associate Professor” in school of “Computer Engineering” or “Electronic Engineering”.

Sample Input:
NA

Sample Output:
Display the row details as per the specification.

	5.9. Find the employees who are working as “Professor” in school of “Computer Engineering” or “Mechanical Engineering” and whose bonus is NULL. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	5.10. For each school & title, find the average salary and number of employees. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	5.11. Who works in the same school in which John Smith works? **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
Day 6	**6.1.** Refer the tables created in 3.1. **Sample Input:** NA **Sample Output:** Tables are created. 1 row(s) are inserted to respective table.

	6.2. Display employee "Andrew Brown" school name. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	6.3. Display full name of the supervisor for employee ID "128433". **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	6.4. Who belongs to same school as "Rod Marsh"? **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	6.5. Who has the same title as "Stanely Garner"? **Sample Input:** NA **Sample Output:** Display the row details as per the specification.

	6.6. Which employees are working in the school longer than “Larry Houston”? **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	6.7. Find the number of employees working in school of computer engineering. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	6.8. Find the number of employees working in school of civil engineering and as professor. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	6.9. Find the employees with minimum salary in their own school in associate professor rank. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.

	6.10. Find the employees with maximum salary in their own school in professor rank. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
	6.11. Find the average salary by each school. **Sample Input:** NA **Sample Output:** Display the row details as per the specification.
Day 7	**7.1.** Refer the tables created in 3.1. **Sample Input:** NA **Sample Output:** Tables are created. 1 row(s) are inserted to respective table.
	7.2. Create a view to include all employee and school, but hide INJECTED_DATE. **Sample Input:** NA **Sample Output:** The view is created.

	7.3. Create a view to include all employee and JOB, but hide INJECTED_DATE. **Sample Input:** NA **Sample Output:** The view is created.
	7.4. Create a view to include all employee and PAY, but hide INJECTED_DATE. **Sample Input:** NA **Sample Output:** The view is created.
	7.5. Create a view to include school name, number of employees and average salary by each school. **Sample Input:** NA **Sample Output:** The view is created.
	7.6. Grant only select privilege to another user for EMPLOYEE table **Sample Input:** NA **Sample Output:** Grant is succeeded.

	7.7. Grant only update privilege to another user for SCHOOL table. **Sample Input:** NA **Sample Output:** Grant is succeeded.
	7.8. Grant only delete privilege to another user for JOB table. **Sample Input:** NA **Sample Output:** Grant is succeeded.
	7.9. Create a synonym to include all employee and school. **Sample Input:** NA **Sample Output:** Synonym is created.
	7.10. Create a synonym to include all employees, schools and jobs. **Sample Input:** NA **Sample Output:** Synonym is created.

	7.11. Create a synonym to include all employees, schools, jobs, and pay. **Sample Input:** NA **Sample Output:** Synonym is created.
	7.12. Create index on Sex in Employee. **Sample Input:** NA **Sample Output:** Index is created.
Day 8	**8.1.** Write a PL/SQL program to perform the addition, subtraction, division and multiplication of two user supplied integers and display the result. **Sample Input:** Enter integer 1:15 Enter integer 2:10 **Sample Output:** 15+10 = 25 15-10 = 5 15/10 = 1 15 * 10 = 150

	8.2. Write a PL/SQL program to find the square, cube, and double of an input number and print results. **Sample Input:** Enter the number:15 **Sample Output:** Square of 15 = 225 Cube of 15 = 3375 Double of 15 = 30
	8.3. Write a PL/SQL program to swap the values of two variables. Print the output before and after swapping. **Sample Input:** Enter number 1:15 Enter number 2:25 **Sample Output:** Before swapping: number 1 = 15 and number 2 = 25 After swapping: number 1 = 25 and number 2 = 15
	8.4. Write a PL/SQL program with two user supplied variables i.e. the first name and the last name. Print the full name with last name and first name separated by comma and a space. **Sample Input:** Enter first name: Rajat Enter first name: Behera **Sample Output:** Behera, Rajat

	8.5. Write a PL/SQL program to convert given seconds into its equivalent hours, minutes and seconds. Example. 7560 second = 2 hours, 27 minutes and 40 seconds **Sample Input:** Enter value in second: 7560 **Sample Output:** Hours: 2, Minutes: 27, and Seconds: 40
	8.6. Write a PL/SQL program to find the average mark of 5 subjects of a student. Assume full mark for each subject is 100. **Sample Input:** Enter mark of subject 1: 75 Enter mark of subject 2: 75 Enter mark of subject 3: 75 Enter mark of subject 4: 75 Enter mark of subject 5: 75 **Sample Output:** Average is 75
	8.7. A cashier has currency notes of denominations 10, 50 and 100. If the amount to be withdrawn is input through the keyboard in hundreds, find the total number of currency notes of each denomination the cashier will have to give to the withdrawer. **Sample Input:** Enter the currency: 2060 **Sample Output:** No of 100 denominations = 20 No of 50 denominations = 1 No of 10 denominations = 1

	8.8. Using a FOR loop, print the values 10 to 1 in reverse order. **Sample Input:** NA **Sample Output:** 10 9 8 7 6 5 4 3 2 1
	8.9. Input a user supplied number and then print its multiplication table (from 1 to 10) using a WHILE loop. **Sample Input:** Enter the number = 2 **Sample Output:** 2 X 1 = 2 2 X 2 = 4 2 X 3 = 6 2 X 4 = 8 2 X 5 = 10 2 X 6 = 12 2 X 7 = 14 2 X 8 = 16 2 X 9 = 18 2 X 10 = 20
	8.10. Ramesh’s basic salary is input through the keyboard. His dearness allowance is 40% of basic salary, and house rent allowance is 20% of basic salary. Write a PL/SQL program to calculate his gross salary. **Sample Input:** Enter basic salary = 1000 **Sample Output:** Gross Salary = 1600

Day 9	**9.1.** Create a PL/SQL procedure that outputs the message “I am a PL/SQL expert.” **Sample Input:** NA **Sample Output:** I am a PL/SQL expert.
	9.2. Create a PL/SQL function to find out if a year is a leap year. **Sample Input:** Enter the year: 2024 **Sample Output:** 2024 is a leap year.
	9.3. Refer the tables created in 3.1. Create a PL/SQL procedure that takes employee ID as the input and displays the employee full name with name of the school. **Sample Input:** Enter the employee ID: 1234 **Sample Output:** Display the desired result.
	9.4. Refer the tables created in 3.1. Create a PL/SQL function that takes school ID as the input. If the school does not contain school name, return a false, otherwise return a true value. Print the appropriate message in the calling program based on the result. **Sample Input:** Enter the school ID: 12 **Sample Output:** Display the desired result.

	9.5. Refer the tables created in 3.1. Create a PL/SQL function to revise the salary by 20%, who works in the same school in which "John Smith" works? Print the appropriate message in the calling program based on the result. **Sample Input:** NA **Sample Output:** Display the desired result.
Day 10	**10.1.** Write PL/SQL block that asks user to input first number, second number and an arithmetic operator (+, -, *, or /). If operator is invalid, throw and handle a user defined exception. If second number is 0 and the operator is /, handle ZERO_DIVIDE predefined server exception. **Sample Input:** Enter first number: 10 Enter second number: 0 **Sample Output:** 10 + 0 = 10 10 – 0 = 10 10 * 0 = 0 Divide by 0 error occurred.
	10.2. Refer the tables created in 3.1. Write a PL/SQL block that retrieves entire EMPLOYEE table into a cursor. Then, ask user to input an Employee Id to search. If employee exists then print its information, but if employee does not exist throw a user-defined exception and handle it with a message 'Employee does not exist'. **Sample Input:** Enter employee ID: 1234 **Sample Output:** Display the desired result.

	10.3. Refer the tables created in 3.1. Create a PL/SQL block to increase salary of employees in school of Computer Engineering. The salary increase is 15% for the employees making less than 100K and 10% for the employees making 100K or more. **Sample Input:** NA **Sample Output:** Display the desired result.
	10.4. Refer the tables created in 3.1. Create a PL/SQL block to declare a cursor to select last name, first name, salary and hire date of the employee. Retrieve each row from the cursor and print the employee’s information if the employee’s salary is greater than 50000 and the hire date is before 31-APR-2021. **Sample Input:** NA **Sample Output:** Display the desired result.
	10.5. Refer the tables created in 3.1. Create a PL/SQL block to declare a cursor and retrieve each row from the cursor to display employee information, drawing more than the average salary in “Professor” rank. **Sample Input:** NA **Sample Output:** Display the desired result.

	10.6. Refer the tables created in 3.1. Develop BEFORE INSERT trigger on EMPLOYEE. **Sample Input:** NA **Sample Output:** Trigger is created.

Grading Policies:

<table>
<tr><th colspan="5"><b><i>Continuous Evaluation components</i></b></th></tr>
<tr><th>Sr#</th><th>Area</th><th>Mark</th><th>#</th><th>Total</th></tr>
<tr><td><b>1</b></td><td colspan="4"><b>Internal Sending</b></td></tr>
<tr><td>1.1</td><td>Lab record evaluation</td><td>1</td><td>10</td><td>10</td></tr>
<tr><td>1.2</td><td>Quiz</td><td>5</td><td>2</td><td>10</td></tr>
<tr><td>1.3</td><td>Written Viva</td><td>1</td><td>10</td><td>10</td></tr>
<tr><td>1.4</td><td>Program Execution</td><td>2</td><td>10</td><td>20</td></tr>
<tr><td>1.5</td><td>Class Participation</td><td>1</td><td>10</td><td>10</td></tr>
<tr><td colspan="4"><b>Total</b></td><td>60</td></tr>
<tr><td colspan="5"><b><i>End semester evaluation</i></b></td></tr>
<tr><td>2.1</td><td>Program Execution</td><td>5</td><td>4</td><td>20</td></tr>
<tr><td>2.2</td><td>Program Approach</td><td>5</td><td>4</td><td>20</td></tr>
<tr><td colspan="4"><b>Total</b></td><td>40</td></tr>
</table>

Practice Problem Sets:-

Lab 1	None
Lab 2	None

Lab 3	Refer the tables created in 3.1. **Q1.** Write a SELECT statement that returns description, cost, and 5% sales tax for each product and total cost with sales tax. List the products in order from most to least expensive. **Q2.** Write a query to return all the products that start with the letter P. **Q3.** Write a query to return all products that do not start with the letter P. **Q4.** Write a SELECT statement that returns the ID, description, and cost. Limit the product cost to between 15.00 and 100.50 using between operator. **Q5.** Write a SELECT statement that returns the ID, description, and cost. Limit the product cost to between 1.00 and 12.50 using conjunctive and comparison operator
Lab 4	Refer the tables created in 3.1. **Q1.** Find the products created in the month of ‘May’. **Q2.** Find the products created in current year. **Q3.** Find 2 to the power 10. **Q4.** Find the ceil and floor value for each job salary. **Q5.** Find the average bonus for all jobs. **Q6.** Count the distinct job title. **Q7.** Display the product name, description, created date with the format ‘20 June 2023’. **Q8.** Display the product name, description in the ascending order of the length of the product description.
Lab 5	Refer the tables created in 3.1. **Q1.** Create an new table “EMP_24012017” from EMPLOYEE **Q2.** Insert data to “EMP_24012017” for the employees belongs to “Computer Engineering” school. **Q3.** Create a sequence called dummy_seq and associate with a table
Lab 6	Refer the tables created in 3.1. **Q1.** Find the employees who don't align to any school. **Q2.** Find the employees who are aligned to more than one school. **Q3.** Find the employees who are more than 5 years of veteran in school of Computer Science or school of Mechanical Engineering.

Lab 7	Refer the tables created in 3.1. **Q1.** Create index on First Name, Middle Name and Last Name in Employee. **Q2.** Rename any created index. **Q3.** Drop any created index. **Q4.** Commit the transaction performed till now. **Q5.** Drop any created view. **Q6.** Rollback the entire transaction.
Lab 8	Refer the tables created in 3.1. **Q1.** Write a PL/SQL program to display the full name, Job Title and Salary of the employee whose ID is “140345”. **Q2.** Write a PL/SQL program to display the full name, Job Title, School name and Salary of the employee whose ID is “150985”. **Q3.** Write a PL/SQL program to display the full name of the youngest employee in the school of computer engineering. **Q4.** Write a PL/SQL program to display the number of years, number of months and days of the oldest employee between hire date and today. **Q5.** Write a PL/SQL program to display the average age of the assistant professor in school of mechanical engineering. **Q6.** Write a PL/SQL program to display the job title, salary, bonus, name and employee ID of school of electrical engineering dean.
Lab 9	Refer the tables created in 3.1. Create a PL/SQL procedure that will present the analysis of the employee. **Q1.** Birthdays of number of employees per each month **Q2.** Ratio of male to female **Q3.** Number of hires per each month **Q4.** Number of employees working for each school **Q5.** Number of employees with different job titles working for each school.

Lab 10	Refer the tables created in 3.1. **Q1.** Develop AFTER INSERT trigger on SCHOOL. **Q2.** Develop BEFORE UPDATE trigger on EMPLOYEE. **Q3.** Develop AFTER UPDATE trigger on SCHOOL. **Q4.** Develop BEFORE DELETE and AFTER DELETE trigger on EMPLOYEE and SCHOOL.

Reference Materials:-

Reference Book:

- None

Reference Site:

- Oracle Tutorial - https://www.javatpoint.com/oracle-tutorial