Module 4 –

Introduction to DBMS Introduction to SQL Theory Questions:

1. What is SQL, and why is it essential in database management?

.Sql is a special language used to manage and work with data in databases. It helps you add, find, change, or delete data easily. SQL is important because it makes handling large amounts of data simple and organized, which is essential for websites, apps, and businesses that rely on data.

1. Explain the difference between DBMS and RDBMS.

**DBMS (Database Management System)** is software used to store and manage data. It can handle any type of data but doesn’t always organize it into tables.

**RDBMS (Relational Database Management System)** is a type of DBMS that stores data in **tables** with rows and columns. It follows a strict structure and supports relationships between different tables using keys.

1. Describe the role of SQL in managing relational databases.

SQL plays a key role in managing relational databases by allowing users to interact with the data stored in tables. It helps in creating and modifying database structures, adding or updating data, and retrieving specific information through queries. SQL also manages user access and ensures data security. Overall, it provides a simple and effective way to organize, control, and analyze data in relational databases.

1. What are the key features of SQL?

 Data Querying – SQL allows users to retrieve specific data from a database using the SELECT command.

 Data Manipulation – It helps in inserting, updating, and deleting data using commands like INSERT, UPDATE, and DELETE.

 Data Definition – SQL is used to create and modify database structures like tables using CREATE, ALTER, and DROP commands.

 Data Control – It manages user permissions with commands like GRANT and REVOKE.

 Transaction Control – SQL ensures data accuracy by grouping operations using COMMIT and ROLLBACK.

 Built-in Functions – It provides functions for calculations such as COUNT, AVG, MAX, and MIN.

 Multiple Table Handling – SQL can connect data from different tables using JOIN operations.

LAB EXERCISES:

• Lab 1: Create a new database named school\_db and a table called students with the following columns: student\_id, student\_name, age, class, and address.

-- Create a new database

CREATE DATABASE school\_db;

-- Use the new database

USE school\_db;

-- Create the students table

CREATE TABLE students (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(100),

age INT,

class VARCHAR(50),

address VARCHAR(255)

);

• Lab 2: Insert five records into the students table and retrieve all records using the SELECT statement.

-- Insert records into the students table

INSERT INTO students (student\_id, student\_name, age, class, address) VALUES

(1, 'Alice Smith', 14, '8A', '123 Oak Street'),

(2, 'Bob Johnson', 15, '9B', '456 Maple Avenue'),

(3, 'Charlie Lee', 13, '7C', '789 Pine Lane'),

(4, 'Diana Patel', 14, '8A', '321 Birch Road'),

(5, 'Ethan Kim', 15, '9B', '654 Cedar Street');

-- Retrieve all records from the students table

SELECT \* FROM students;

SQL SYNTAX:

**Theory Questions**

1. What are the basic components of SQL syntax?  
   SQL syntax is made up of keywords, identifiers (like table and column names), operators, clauses, and expressions. The basic structure includes commands like SELECT, INSERT, UPDATE, DELETE, and uses punctuation like commas and semicolons.
2. Write the general structure of an SQL SELECT statement.

SELECT column1, column2

FROM table\_name

WHERE condition

ORDER BY column;

1. Explain the role of clauses in SQL statements.  
   Clauses are parts of SQL statements that define how the operation should be performed. For example, SELECT defines what to retrieve, FROM shows where the data is coming from, WHERE filters records, and ORDER BY sorts the results.

**Lab Exercises**

Lab 1: Write SQL queries to retrieve specific columns (student\_name and age) from the students table.

SELECT student\_name, age

FROM students;

Lab 2: Write SQL queries to retrieve all students whose age is greater than 10.

SELECT \*

FROM students

WHERE age > 10;

4o

1. SQL Constraints

Theory Questions:

**1. What are constraints in SQL? List and explain the different types of constraints.**  
Constraints are rules applied to table columns to ensure the accuracy and integrity of the data in a database.  
Common types of constraints include:

* primary key – ensures each row has a unique and non-null identifier
* foreign key – links a column to the primary key of another table to maintain referential integrity
* not null – ensures a column cannot have a null value
* unique – ensures all values in a column are different
* check – ensures that values in a column meet a specific condition
* default – assigns a default value to a column if no value is provided

**2. How do PRIMARY KEY and FOREIGN KEY constraints differ?**

* primary key uniquely identifies each record in a table and does not allow null or duplicate values
* foreign key is used to link two tables and ensures that the value in one table matches a value in another table's primary key

**3. What is the role of NOT NULL and UNIQUE constraints?**

* not null ensures that a column must always have a value; it cannot be left empty
* unique ensures that all values in a column are different, preventing duplicate entries in that column

LAB EXERCISES:

Lab 1: Create a table teachers with constraints

CREATE TABLE teachers (

teacher\_id INT PRIMARY KEY,

teacher\_name VARCHAR(100) NOT NULL,

subject VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE

);

Lab 2: Add a foreign key in the students table to relate to teacher\_id  
First, add the teacher\_id column if it doesn't exist:

ALTER TABLE students

ADD teacher\_id INT;

Then, add the foreign key constraint:

ALTER TABLE students

ADD CONSTRAINT fk\_teacher

FOREIGN KEY (teacher\_id) REFERENCES teachers(teacher\_id);

1. Main SQL Commands and Sub-commands (DDL)

Theory Questions:

1. Define the SQL Data Definition Language (DDL)  
   DDL stands for Data Definition Language. It includes SQL commands used to define and manage the structure of database objects like tables, views, and indexes. Common DDL commands are CREATE, ALTER, DROP, and TRUNCATE.
2. Explain the CREATE command and its syntax  
   The CREATE command is used to create new database objects such as databases and tables.  
   Syntax for creating a table:

CREATE TABLE table\_name (

column1 datatype constraint,

column2 datatype constraint,

...

);

1. What is the purpose of specifying data types and constraints during table creation?  
   Specifying data types ensures that only the correct kind of data is stored in each column, such as numbers, text, or dates. Constraints are used to enforce rules on the data, like making sure values are not empty, unique, or linked to another table. This helps maintain data accuracy, consistency, and integrity.

4 LAB EXERCISES:

Lab 1: Create a table courses with course\_id as the primary key

CREATE TABLE courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(100),

course\_credits INT

);

Lab 2: Use the CREATE command to create a database named university\_db

. CREATE DATABASE university\_db;

* 1. ALTER Command

Theory Questions:Top of Form

1. What is the use of the ALTER command in SQL?  
   The ALTER command is used to change the structure of an existing table. You can use it to add new columns, modify existing columns, or delete columns from a table. It is also used to add or remove constraints.
2. How can you add, modify, and drop columns from a table using ALTER?

* To add a column:

ALTER TABLE table\_name

ADD column\_name datatype;

* To modify a column:

ALTER TABLE table\_name

MODIFY column\_name new\_datatype;

* To drop (remove) a column:

ALTER TABLE table\_name

DROP COLUMN column\_name;

SQL ASSIGNMENT: THEORY AND LAB EXERCISES

**6. DROP Command**

Theory Questions:

1. The DROP command is used to permanently delete database objects like tables, views, or entire databases from the system.
2. Dropping a table permanently removes the table structure and all the data inside it. This action cannot be undone.

Lab Exercises:  
Lab 1:

DROP TABLE teachers;

Lab 2:

DROP TABLE students;

-- To verify, try selecting from the table (this will show an error if the table is dropped)

SELECT \* FROM students;

**7. Data Manipulation Language (DML)**

Theory Questions:

1. INSERT is used to add new records into a table. UPDATE is used to change existing records. DELETE is used to remove records from a table.
2. The WHERE clause is important in UPDATE and DELETE to specify which rows to change or delete. Without it, all rows could be affected.

Lab Exercises:  
Lab 1:

INSERT INTO courses (course\_id, course\_name, course\_credits) VALUES

(101, 'Math', 3),

(102, 'Science', 4),

(103, 'History', 2);

Lab 2:

UPDATE courses SET course\_credits = 5 WHERE course\_id = 102;

Lab 3:

DELETE FROM courses WHERE course\_id = 103;

**8. Data Query Language (DQL)**

Theory Questions:

1. The SELECT statement is used to query and retrieve data from a database.
2. ORDER BY is used to sort results. WHERE is used to filter rows based on a condition.

Lab Exercises:  
Lab 1:

SELECT \* FROM courses;

Lab 2:

SELECT \* FROM courses ORDER BY course\_credits DESC;

Lab 3:

SELECT \* FROM courses LIMIT 2;

**9. Data Control Language (DCL)**

Theory Questions:

1. GRANT gives users access or permissions. REVOKE removes those permissions.
2. You can control user actions (like SELECT or INSERT) using these commands.

Lab Exercises:  
Lab 1:

CREATE USER user1 IDENTIFIED BY 'password1';

CREATE USER user2 IDENTIFIED BY 'password2';

GRANT SELECT ON courses TO user1;

Lab 2:

REVOKE INSERT ON courses FROM user1;

GRANT INSERT ON courses TO user2;

**10. Transaction Control Language (TCL)**

Theory Questions:

1. COMMIT saves changes permanently. ROLLBACK undoes changes since the last COMMIT.
2. Transactions allow multiple operations to be grouped. You can commit or roll them back together.

Lab Exercises:  
Lab 1:

INSERT INTO courses VALUES (104, 'English', 3);

COMMIT;

Lab 2:

INSERT INTO courses VALUES (105, 'Geography', 2);

ROLLBACK;

Lab 3:

SAVEPOINT before\_update;

UPDATE courses SET course\_credits = 6 WHERE course\_id = 101;

ROLLBACK TO before\_update;

**11. SQL Joins**

Theory Questions:

1. JOIN combines rows from two or more tables. INNER JOIN shows matched records. LEFT JOIN shows all from the left table. RIGHT JOIN shows all from the right. FULL OUTER JOIN shows all records with matches where possible.
2. Joins are used to view related data from different tables together.

Lab Exercises:  
Lab 1:

CREATE TABLE departments (dept\_id INT, dept\_name VARCHAR(100));

CREATE TABLE employees (emp\_id INT, emp\_name VARCHAR(100), dept\_id INT);

SELECT emp\_name, dept\_name FROM employees

INNER JOIN departments ON employees.dept\_id = departments.dept\_id;

Lab 2:

SELECT dept\_name, emp\_name FROM departments

LEFT JOIN employees ON departments.dept\_id = employees.dept\_id;

**12. SQL Group By**

Theory Questions:

1. GROUP BY groups rows with the same values and is often used with aggregate functions like COUNT or AVG.
2. GROUP BY organizes results into groups. ORDER BY sorts the final result.

Lab Exercises:  
Lab 1:

SELECT dept\_id, COUNT(\*) FROM employees

GROUP BY dept\_id;

Lab 2:

SELECT dept\_id, AVG(salary) FROM employees

GROUP BY dept\_id;

**13. SQL Stored Procedure**

Theory Questions:

1. A stored procedure is a set of SQL statements saved under a name that can be reused. It is more efficient than writing SQL every time.
2. Stored procedures help with reusability, security, and improved performance.

Lab Exercises:  
Lab 1:

CREATE PROCEDURE GetEmployeesByDept (IN dep\_id INT)

BEGIN

SELECT \* FROM employees WHERE dept\_id = dep\_id;

END;

Lab 2:

CREATE PROCEDURE GetCourseDetails (IN id INT)

BEGIN

SELECT \* FROM courses WHERE course\_id = id;

END;

**14. SQL View**

Theory Questions:

1. A view is a virtual table created from a SELECT query. It doesn’t store data permanently.
2. Views simplify complex queries and enhance security by restricting access to specific data.

Lab Exercises:  
Lab 1:

CREATE VIEW emp\_dept\_view AS

SELECT emp\_name, dept\_name FROM employees

JOIN departments ON employees.dept\_id = departments.dept\_id;

Lab 2:

CREATE OR REPLACE VIEW emp\_dept\_view AS

SELECT emp\_name, dept\_name FROM employees

JOIN departments ON employees.dept\_id = departments.dept\_id

WHERE salary >= 50000;

**15. SQL Triggers**

Theory Questions:

1. A trigger is a special procedure that runs automatically when an INSERT, UPDATE, or DELETE happens.
2. INSERT triggers run when new data is added, UPDATE triggers when data is changed, and DELETE triggers when data is removed.

Lab Exercises:  
Lab 1:

CREATE TRIGGER log\_new\_employee

AFTER INSERT ON employees

FOR EACH ROW

BEGIN

INSERT INTO log\_table (action\_type, emp\_id) VALUES ('INSERT', NEW.emp\_id);

END;

Lab 2:

CREATE TRIGGER update\_timestamp

BEFORE UPDATE ON employees

FOR EACH ROW

BEGIN

SET NEW.last\_modified = NOW();

END;

**16. Introduction to PL/SQL**

Theory Questions:

1. PL/SQL is Oracle's extension of SQL that includes programming features like loops and conditions.
2. PL/SQL supports reusability, better performance, and error handling.

Lab Exercises:  
Lab 1:

DECLARE

total\_emps INT;

BEGIN

SELECT COUNT(\*) INTO total\_emps FROM employees;

DBMS\_OUTPUT.PUT\_LINE('Total Employees: ' || total\_emps);

END;

Lab 2:

DECLARE

total\_sales NUMBER;

BEGIN

SELECT SUM(amount) INTO total\_sales FROM orders;

DBMS\_OUTPUT.PUT\_LINE('Total Sales: ' || total\_sales);

END;

**17. PL/SQL Control Structures**

Theory Questions:

1. Control structures control the flow of code. IF-THEN checks conditions. LOOP repeats code multiple times.
2. They help perform actions conditionally and repeatedly, making logic more flexible.

Lab Exercises:  
Lab 1:

DECLARE

dept\_name VARCHAR(50);

BEGIN

SELECT dept\_name INTO dept\_name FROM departments WHERE dept\_id = 1;

IF dept\_name = 'HR' THEN

DBMS\_OUTPUT.PUT\_LINE('Employee is in HR department');

END IF;

END;

Lab 2:

DECLARE

emp\_name VARCHAR(100);

emp\_cursor CURSOR IS SELECT emp\_name FROM employees;

BEGIN

FOR emp\_record IN emp\_cursor LOOP

DBMS\_OUTPUT.PUT\_LINE(emp\_record.emp\_name);

END LOOP;

END;

**18. SQL Cursors**

Theory Questions:

1. A cursor lets you process query results row by row. Implicit cursors are automatic. Explicit cursors are defined manually.
2. Use explicit cursors when you need detailed control over row-by-row processing.

Lab Exercises:  
Lab 1:

DECLARE

emp\_record employees%ROWTYPE;

CURSOR emp\_cursor IS SELECT \* FROM employees;

BEGIN

OPEN emp\_cursor;

LOOP

FETCH emp\_cursor INTO emp\_record;

EXIT WHEN emp\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(emp\_record.emp\_name);

END LOOP;

CLOSE emp\_cursor;

END;

Lab 2:

DECLARE

course\_record courses%ROWTYPE;

CURSOR course\_cursor IS SELECT \* FROM courses;

BEGIN

OPEN course\_cursor;

LOOP

FETCH course\_cursor INTO course\_record;

EXIT WHEN course\_cursor%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(course\_record.course\_name);

END LOOP;

CLOSE course\_cursor;

END;

**19. Rollback and Commit Savepoint**

Theory Questions:

1. A SAVEPOINT marks a point in a transaction you can roll back to without undoing the entire transaction. COMMIT makes changes permanent, ROLLBACK undoes them.
2. Savepoints are useful when you want to undo only part of a transaction.

Lab Exercises:  
Lab 1:

SAVEPOINT sp1;

INSERT INTO courses VALUES (106, 'Biology', 4);

ROLLBACK TO sp1;

Lab 2:

SAVEPOINT sp2;

INSERT INTO courses VALUES (107, 'Art', 2);

COMMIT;

INSERT INTO courses VALUES (108, 'Music', 3);

ROLLBACK TO sp2;

Bottom of Form