Module 4 – Introduction to DBMS

❖ Introduction to SQL

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

Ans:-

→ SQL is a standard language for storing, manipulating and retrieving data in databases. SQL allows you to access and manipulate the databases.

→ To use SQL in: MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.

2) Explain the difference between DBMS and RDBMS.

Ans:-

→ DBMS :

- Data stored is in the file format.
- Individual access of data elements.
- No connection between data.
- No support for distributed database.
- Data stored is a small quantity.

- DBMS supports a single user.
- Example: XML, Microsoft Access.

→ RDBMS :

- Data stored is in table format.
- Multiple data elements are accessible together.
- Data in the form of a table are linked together.
- Support distributed database.
- Data is stored in a large amount.
- RDBMS supports multiple users.
- Example: Oracle, SQL Server.

3) Describe the role of SQL in managing relational databases?

Ans:-

→ Data Definition: SQL allows you to create, modify, and delete database structures such as tables, indexes, views, and relationships.

→ Data Manipulation: SQL enables you to insert, update, and delete data in tables.

→ Data Querying: SQL allows you to retrieve specific data from tables using SELECT statements.

→Data Security: SQL provides features to control access to databases, such as user authentication, authorization, and encryption.

→ Data Integrity: SQL ensures data consistency and accuracy by enforcing constraints, such as primary keys, foreign keys, and check constraints.

→ Data Optimization: SQL allows you to optimize database performance by creating indexes, optimizing queries, and managing database statistics.

4) What are the key features of SQL?

Ans:-

→ Declarative Language: SQL is a declarative language, meaning you specify what you want to do with your data, rather than how to do it.

→ Querying Data: SQL allows you to retrieve specific data from databases using SELECT statements.

→ Manipulating Data: SQL enables you to insert, update, and delete data in tables.

→ Data Definition: SQL allows you to create, modify, and delete database structures such as tables, indexes, views, and relationships.

→ Data Control: SQL provides features to control access to databases, such as user authentication, authorization, and encryption.

❖ LAB EXERCISES :

1) 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.

Ans:--

```
→CREATE TABLE student
→(
→   student_id int,
→   student_name text,
→  student_age int,
→  student_class text,
→  student_address text
→  );
```

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

Ans:--

```
→ INSERT INTO student
VALUES(101,'shraddha','21','a','ahemdabad'),(102
,'Tisha','19','b','Rajkot'),(103,'Aayushi','20','c','Juna
```

ghagh'),(104,'Gopi','18','d','gondal'),(105,'Bhoomi', '22','e','ahemdabad');

→

student_id	student_name	student_age	student_class	student_address
101	shraddha	21	a	ahemdabad
102	Tisha	19	b	Rajkot
103	Aayushi	20	c	Junaghagh
104	Gopi	18	d	gondal
105	Bhoomi	22	e	ahemdabad

2. SQL Syntax

1) What are the basic components of SQL syntax?

Ans:-

→ Commands: SQL commands are used to perform specific actions, such as creating or modifying database structures, inserting or updating data, and querying data. Common SQL commands include SELECT, INSERT, UPDATE, DELETE, CREATE, and DROP.

→ Clauses: SQL clauses are used to specify conditions or filters for a query. Common SQL clauses include WHERE, FROM, GROUP BY, HAVING, and ORDER BY.

→ Functions: SQL functions are used to perform calculations or transformations on data. Common SQL functions include SUM, AVG, MAX, MIN, and COUNT.

→ Operators: SQL operators are used to perform comparisons, arithmetic operations, and logical operations. Common SQL operators include =, <, >, <=, >=, !=, AND, OR, and NOT.

→ Identifiers: SQL identifiers are used to refer to database objects, such as tables, columns, and indexes. Identifiers can be either quoted or unquoted.

2) Write the general structure of an SQL SELECT statement.

Ans:-

→ SELECT
→ [DISTINCT]
→ column1,
→ column2,
→ Column

→FROM
→ table_name

→ [JOIN clause]
→ [WHERE clause]
→ [GROUP BY clause]
→ [HAVING clause]
→ [ORDER BY clause]
→ [LIMIT clause];

3) Explain the role of clauses in SQL statements.

Ans:-

→ Filtering: Clauses like WHERE, HAVING, and JOIN help to filter data based on conditions, relationships, or aggregations.
→ Sorting: Clauses like ORDER BY help to sort data in ascending or descending order.
→ Grouping: Clauses like GROUP BY help to group data by one or more columns.
→ Limiting: Clauses like LIMIT help to limit the number of rows returned.
→ Joining: Clauses like JOIN help to combine data from multiple tables.

LAB EXERCISES:

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

Ans:-

→ SELECT student_name,student_age FROM student;

→

student_name	student_age
shraddha	21
Tisha	19
Aayushi	20
Gopi	18
Bhoomi	22

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

Ans:-

→SELECT* FROM student WHERE student_age>10;

→

student_id	student_name	student_age	student_class	student_address
101	shraddha	21	a	ahemdabad

102	Tisha	19	b	Rajkot
103	Aayushi	20	c	Junaghagh
104	Gopi	18	d	gondal
105	Bhoomi	22	e	ahemdabad

3. SQL Constraints

1) What are constraints in SQL? List and explain the different types of constraints.

Ans:-

→ NOT NULL - Ensures that a column cannot have a NULL value

→ UNIQUE - Ensures that all values in a column are different

→ PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table.

→ FOREIGN KEY - Prevents actions that would destroy links between tables.

→ DEFAULT - Sets a default value for a column if no value is specified

→ CHECK - Ensures that the values in a column satisfies a specific condition

→ CREATE INDEX - Used to create and retrieve data from the database very quickly.

2) How do PRIMARY KEY and FOREIGN KEY constraints differ?

Ans:-

→ PRIMARY KEY (PK) Constraint :

- Uniquely identifies each record in a table.
- Ensures that no duplicate values are entered in the primary key column.
- Automatically creates a unique index on the column.
- Can be composed of one or more columns.

→ FOREIGN KEY (FK) Constraint :

- Links two tables together.
- Ensures that the value in the foreign key column matches an existing value in the primary key column of the other table.
- Maintains referential integrity between tables.
- Can be composed of one or more columns.

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

Ans:-

→ NOT NULL Constraint :

- Ensures that a column cannot contain null values.
- Requires a value to be entered for the column in every row.
- Prevents missing or unknown values from being stored in the column.

→ UNIQUE Constraint :

- Ensures that all values in a column are unique.
- Prevents duplicate values from being entered in the column.
- Can be applied to one or more columns.

LAB EXERCISES:

Lab 1: Create a table teachers with the following columns: teacher_id (Primary Key), teacher_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).

Ans:-

```
→CREATE TABLE teachers (
→  teacher_id INT PRIMARY KEY,
→   teacher_name VARCHAR(100) ,
→   subject VARCHAR(100) ,
→   email VARCHAR(100)
→   );
```

Lab 2: Implement a FOREIGN KEY constraint to relate the teacher_id from the teachers table with the students table.

Ans:-

```
→CREATE TABLE students
→(
→ student_id INT PRIMARY KEY,
→name VARCHAR(100),
→ teacher_id INT,
→FOREIGN KEY (teacher_id)  REFERENCES
   teachers(teacher_id)
→ );
```

→ INSERT INTO teachers VALUES (101,'Ankita','maths','ankita123@gmail.com'),(102,'Bhoomi','maths','bhoomi4123@gmail.com'),(103,'Jyoti','maths','jyoti1123@gmail.com'),(104,'priya','maths','priya23@gmail.com');

→

teacher_id _1	teacher_name	subject	email
101	Ankita	maths	ankita123@gmail.com
102	Bhoomi	maths	bhoomi4123@gmail.com
103	Jyoti	maths	jyoti1123@gmail.com
104	priya	maths	priya23@gmail.com

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

1) Define the SQL Data Definition Language (DDL).

Ans:-

→ SQL Data Definition Language (DDL) is a subset of SQL that is used to define and modify the structure of a database.

→DDL statements are used to create, modify, and delete database objects such as tables, indexes, views, and relationships.

2) Explain the CREATE command and its syntax.

Ans:-

→ The CREATE command is a Data Definition Language (DDL) statement that is used to create a new database object, such as a table, index, view, or database.

→ CREATE TABLE Syntax :

```
CREATE TABLE table_name (
column1 data_type,
column2 data_type,
column3 data_type,
);
```

3) What is the purpose of specifying data types and constraints during table creation?

Ans:-

→ Data Types :

- Ensures Data Integrity: By specifying a data type, you ensure that only valid data is stored in the column.
- Optimizes Storage: Choosing the correct data type helps optimize storage space, reducing the risk of data overflow or underflow.
- Improves Query Performance: Data types influence query performance, as the database can optimize queries based on the data type.

→ Constraints :

- Ensures Data Consistency: Constraints, such as PRIMARY KEY, FOREIGN KEY, and UNIQUE, ensure that data is consistent and follows specific rules.

- Prevents Data Errors: Constraints prevent errors, such as duplicate values, invalid data, or orphaned records.
- Maintains Data Relationships: Constraints, like FOREIGN KEY, maintain relationships between tables, ensuring data integrity.

LAB EXERCISES:

Lab 1: Create a table courses with columns: course_id, course_name, and course_credits. Set the course_id as the primary key.

Ans:-

→ Create database courses;

```
→CREATE TABLE courses (
    course_id INT PRIMARY KEY,
    course_name VARCHAR(100) ,
    course_credits INT
);
→INSERT INTO courses VALUES(101,'java','a'),(102,'paython','b'),(103,'data structure','c');
```

course_id	course_name	course_credits
101	java	0
102	python	0
103	data structure	0

Lab 2: Use the CREATE command to create a database university_db.

Ans:-

→`CREATE DATABASE university_db;`

5. ALTER Command

1) What is the use of the ALTER command in SQL?

Ans:-

→ Common uses of the ALTER command :

- Adding or removing columns: ALTER TABLE can be used to add new columns or remove existing columns from a table.

- Modifying column data types: ALTER TABLE can be used to change the data type of an existing column.
- Adding or removing constraints: ALTER TABLE can be used to add or remove constraints, such as PRIMARY KEY, FOREIGN KEY, or UNIQUE constraints.
- Renaming tables or columns: ALTER TABLE can be used to rename a table or column.
- Adding or removing indexes: ALTER TABLE can be used to add or remove indexes from a table.

2) How can you add, modify, and drop columns from a table using ALTER?

Ans:-

→ Adding a Column :

- Use the ALTER TABLE statement followed by the table name.
- Use the ADD COLUMN clause followed by the column name and data type.

→ Modifying a Column :

- Use the ALTER TABLE statement followed by the table name.

- Use the MODIFY COLUMN clause followed by the column name and new data type.

→ Dropping a Column :

- Use the ALTER TABLE statement followed by the table name.
- Use the DROP COLUMN clause followed by the column name.

LAB EXERCISES :

Lab 1: Modify the courses table by adding a column course_duration using the ALTER command.

Ans:-

→
```
ALTER TABLE courses
ADD course_duration INT;
```
→

course_id	course_name	course_credits	course_duration
101	java	0	*NULL*
102	paython	0	*NULL*
103	data structure	0	*NULL*

Lab 2: Drop the course_credits column from the courses table.

→ALTER TABLE courses
DROP COLUMN course_credits;
→

course_id	course_name	course_duration
101	java	*NULL*
102	paython	*NULL*
103	data structure	*NULL*

6. DROP Command

1) What is the function of the DROP command in SQL?

Ans:-

→ Tables: DROP TABLE statement deletes a table and all its data.

→Indexes: DROP INDEX statement deletes an index from a table.

→Views: DROP VIEW statement deletes a view.

→Stored Procedures: DROP PROCEDURE statement deletes a stored procedure.

→Functions: DROP FUNCTION statement deletes a function.

→Triggers: DROP TRIGGER statement deletes a trigger.

→Databases: DROP DATABASE statement deletes a database and all its objects.

2) What are the implications of dropping a table from a database?

Ans:-

→ Data : Dropping a table results in the permanent loss of all data stored in that table.

→ Table Structure : The table structure, including column definitions, data types, and constraints, is also lost.

→ Index and Constraint Removal: Any indexes and constraints associated with the table are automatically dropped.

LAB EXERCISES:

Lab 1: Drop the teachers table from the school_db database.

Ans:-

→ Drop Table teachers;

Lab 2: Drop the students table from the school_db database and verify that the table has been removed.
Ans:-

→ DROP TABLE students;

7. Data Manipulation Language (DML)

1) Define the INSERT, UPDATE, and DELETE commands in SQL.

Ans:-

→ INSERT Command :

- The INSERT command is used to add new records to a database table.

→ UPDATE Command :

- The UPDATE command is used to modify existing records in a database table.

→DELETE Command :

- The DELETE command is used to delete existing records from a database table.

2) What is the importance of the WHERE clause in UPDATE and DELETE operations?

Ans:-

→ The WHERE clause is crucial in UPDATE and DELETE operations because it specifies which records to modify or delete. Here are some reasons why the WHERE clause is important.

→ Without a WHERE clause, an UPDATE or DELETE operation will affect all records in the table, which can lead to unintended changes or data loss.

→ By specifying conditions in the WHERE clause, you can ensure that only the intended records are modified or deleted, maintaining data integrity.

→ A WHERE clause helps reduce errors by ensuring that only the correct records are modified or deleted.

LAB EXERCISES:

Lab 1: Insert three records into the courses table using the INSERT command.

Ans:-

→INSERT INTO courses VALUES (1, 'Introduction to Programming', 4), (2, 'Database Systems', 3),(3, 'Web Development', 3);

course_id	course_name	course_duration
1	Introduction to Programming	4
2	Database Systems	3
3	Web Development	3

Lab 2: Update the course duration of a specific course using the UPDATE command.

Ans:-

```
→UPDATE courses
  SET course_duration = 8
  WHERE course_id = 2;
→
```

course_id	course_name	course_duration
1	Introduction to Programming	4
2	Database Systems	8
3	Web Development	3

Lab 3: Delete a course with a specific course_id from the courses table using the DELETE command.

Ans:-

```
→ DELETE FROM courses
  WHERE course_id = 2;
→
```

course_id	course_name	course_duration
1	Introduction to Programming	4
3	Web Development	3

8. Data Query Language (DQL)

1) What is the SELECT statement, and how is it used to query data?

Ans:-

→ The SELECT statement is a fundamental SQL command used to retrieve data from a database.

→ It allows you to specify which columns, rows, and tables you want to retrieve, and it returns the data in a format that's easy to read and work with.

2) Explain the use of the ORDER BY and WHERE clauses in SQL queries.

Ans:-

→ WHERE Clause :

→ The WHERE clause is used to filter data based on specific conditions. It allows you to specify which rows to include in the result set.

→ ORDER BY Clause :

→ The ORDER BY clause is used to sort the result set in ascending or descending order.

LAB EXERCISES:

Lab 1: Retrieve all courses from the courses table using the SELECT statement.

Ans:-

```
→SELECT                                                    *
  FROM courses;
→
```

				course_id	course_name	course_duration
				1	Introduction to Programming	4
				3	Web Development	3
				101	java	*NULL*
				102	paython	*NULL*
				103	data structure	*NULL*
				104	java	*NULL*
				105	paython	*NULL*
				106	data structure	*NULL*

Lab 2: Sort the courses based on course_duration in descending order using ORDER BY.

Ans:-

```
→SELECT                                    *
  FROM                                  courses
  ORDER BY course_duration DESC;
→
```

course_id	course_name	course_duration _1
1	Introduction to Programming	4
3	Web Development	3
101	java	*NULL*
102	paython	*NULL*
103	data structure	*NULL*
104	java	*NULL*
105	paython	*NULL*
106	data structure	*NULL*

Lab 3: Limit the results of the SELECT query to show only the top two courses using LIMIT.

```
→ SELECT                                   *
  FROM                                  courses
  LIMIT 2;
→
```

course_id	course_name	course_duration
1	Introduction to Programming	4
3	Web Development	3

9. Data Control Language (DCL)

1) What is the purpose of GRANT and REVOKE in SQL?

Ans:-

→ GRANT :

→ The GRANT statement is used to assign permissions or privileges to a user or role, allowing them to perform specific actions on a database object.

→ REVOKE :

→ The REVOKE statement is used to remove permissions or privileges from a user or role, restricting their access to a database object.

2) How do you manage privileges using these commands?

Ans:-

→ Managing privileges using GRANT and REVOKE commands involves.

→ Determine the specific privilege you want to grant, such as SELECT, INSERT, UPDATE, or DELETE.

→ Specify the user or role to which you want to grant the privilege.

→ Only grant the necessary privileges to perform a task.

→ Use roles to manage privileges instead of individual users.

LAB EXERCISES:

Lab 1: Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table.

Ans:-

→
```
CREATE USER user1 IDENTIFIED BY 'password1';
CREATE USER user2 IDENTIFIED BY 'password2'
```
→
```
GRANT SELECT ON courses TO user1;
```
→

course_id	course_name	course_duration
1	Introduction to Programming	4
3	Web Development	3
101	java	*NULL*
102	paython	*NULL*

103	data structure	*NULL*
104	java	*NULL*
105	paython	*NULL*
106	data structure	*NULL*

Lab 2: Revoke the INSERT permission from user1 and give it to user2.

Ans:-

```
→REVOKE INSERT ON courses FROM user1;
→GRANT INSERT ON courses TO user2;
→
```

course_id	course_name	course_duration
1	Introduction to Programming	4
3	Web Development	3
101	java	*NULL*
102	paython	*NULL*
103	data structure	*NULL*
104	java	*NULL*
105	paython	*NULL*
106	data structure	*NULL*

10. Transaction Control Language (TCL)

1) What is the purpose of the COMMIT and ROLLBACK commands in SQL?

Ans:-

→ SQL, COMMIT and ROLLBACK are two essential commands used to manage transactions, which are sequences of operations performed on a database.

→ Permanently save all changes made during a transaction.

→ Mark the end of a transaction, releasing any locks on database resources.

→ Reverse all changes made during a transaction.

→Return the database to its original state before the transaction began.

2) Explain how transactions are managed in SQL databases.

Ans:-

→ Transaction management is a crucial aspect of SQL databases, ensuring data consistency and integrity.

→ A transaction is a sequence of one or more SQL operations (e.g., INSERT, UPDATE, DELETE) executed as a single, all-or-nothing unit.

→ Transactions ensure that either all changes are committed (saved) or none are, maintaining data consistency.

LAB EXERCISES:

Lab 1: Insert a few rows into the courses table and use COMMIT to save the changes.

Ans:-

→ INSERT INTO courses VALUES (4, 'Artificial Intelligence', 4),(5, 'Data Science', 4),(6, 'Cybersecurity', 3);

→

course_id	course_name	course_duration
1	Introduction to Programming	4
3	Web Development	3
4	Artificial Intelligence	4
5	Data Science	4
6	Cybersecurity	3

Lab 2: Insert additional rows, then use ROLLBACK to undo the last insert operation.

Ans:-

→INSERT INTO courses
→ VALUES
→ (7, 'Machine Learning', 4),
→ (8, 'Cloud Computing', 3);
→

course_id	course_name	course_duration
1	Introduction to Programming	4
3	Web Development	3
4	Artificial Intelligence	4
5	Data Science	4
6	Cybersecurity	3
7	Machine Learning	4
8	Cloud Computing	3

Lab 3: Create a SAVEPOINT before updating the courses table, and use it to roll back specific changes.

Ans:-

→BEGIN;
→
→ INSERT INTO courses
→ VALUES (111, 'Introduction to SQL', 'A');
→
→ INSERT INTO courses

→ VALUES (112, 'Python for Data Science', 'D');
→
→ INSERT INTO courses
→ VALUES (113, 'Web Development Basics', 'C');
→
→ COMMIT;
→

				course_id	course_name	course_duration
				1	Introduction to Programming	4
				3	Web Development	3
				4	Artificial Intelligence	4
				5	Data Science	4
				6	Cybersecurity	3
				7	Machine Learning	4
				8	Cloud Computing	3
				101	java	*NULL*
				102	paython	*NULL*
				103	data structure	*NULL*
				104	java	*NULL*
				105	paython	*NULL*
				106	data structure	*NULL*
				111	Introduction to SQL	0
				112	Python for Data Science	0
				113	Web Development Basics	0

→

Lab 2: Insert additional rows, then use ROLLBACK to undo the last insert operation.

Ans:-

```
→       BEGIN;
→
→ INSERT INTO courses
→ VALUES (201, 'SE', '6 weeks');
→
→ INSERT INTO courses
→ VALUES (202, 'Machine Learning Basics', '8 weeks');
→
→ INSERT INTO courses
→ VALUES (203, 'Cloud Computing', '5 weeks');
→
→ ROLLBACK;
→
```

course_id	course_name	course_duration
1	Introduction to Programming	4
3	Web Development	3
4	Artificial Intelligence	4
5	Data Science	4
6	Cybersecurity	3
7	Machine Learning	4
8	Cloud Computing	3

Lab 3: Create a SAVEPOINT before updating the courses table, and use it to roll back specific changes.

Ans:-

→

```
BEGIN
-- Update some records in the courses table
UPDATE courses
SET course_name = 'Mathematics 101'
WHERE course_id = 1;

-- Create a SAVEPOINT after the first update
SAVEPOINT savepoint1;

-- Update more records in the courses table
UPDATE courses
SET course_name = 'Physics 101'
WHERE course_id = 2;

-- Create another SAVEPOINT after the second update
SAVEPOINT savepoint2;

-- Update yet another record
UPDATE courses
SET course_name = 'Chemistry 101'
WHERE course_id = 3;

-- Rollback to the first SAVEPOINT (undoing all changes after SAVEPOINT1)
ROLLBACK TO savepoint1;

-- Commit the remaining changes
COMMIT;

DBMS_OUTPUT.PUT_LINE('Transaction complete. Changes after SAVEPOINT1 have been rolled back.');
```

```
END;
/
```

→

11. SQL Joins :

1) Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?

Ans:-

→ a JOIN is used to combine rows from two or more tables based on a related column between them.

→INNER JOIN: Returns only the rows that have a match in both tables.

→ LEFT JOIN (or LEFT OUTER JOIN): Returns all the rows from the left table and the matching rows from the right table. If there's no match, the result will contain NULL values.

→ RIGHT JOIN (or RIGHT OUTER JOIN): Similar to LEFT JOIN, but returns all the rows from the right table and the matching rows from the left table.

→ FULL OUTER JOIN: Returns all the rows from both tables, with NULL values in the columns where there are no matches.

2) How are joins used to combine data from multiple tables?

Ans:-

→ a fundamental concept in SQL that enable you to combine data from multiple tables based on a related column between them.

→ Combine data from multiple tables to provide a more complete picture.

→ Avoid data redundancy by storing related data in separate tables.

→ Improve data flexibility and scalability.

LAB EXERCISES:

Lab 1: Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.

Ans:-

→CREATE TABLE departments (
→ department_id INT PRIMARY KEY,
→ department_name text
→);
→ CREATE TABLE employees (employee_id INT PRIMARY KEY, employee_name text, department_id INT, FOREIGN KEY (department_id) REFERENCES departments(department_id));
→ INSERT INTO departments (department_id, department_name) VALUES (1, 'Sales'), (2, 'Human Resources'), (3, 'IT');
→ INSERT INTO employees (employee_id, employee_name, department_id) VALUES (101, 'Shraddha', 1), (102, 'Tisha', 2), (103, 'Charu', 3), (104, 'Dharmi', 1);
→ SELECT employees.employee_id, employees.employee_name, departments.department_name FROM employees INNER JOIN departments ON employees.department_id = departments.department_id;
→

employee_id	employee_name	department_id
101	Shraddha	1
102	Tisha	2
103	Charu	3
104	Dharmi	1

→

department_id	department_name
1	Sales

2	Human Resources
3	IT

Lab 2: Use a LEFT JOIN to show all departments, even those without employees.

Ans:-

→SELECT departments.department_id, departments.department_name, employees.employee_id, employees.employee_name
FROM departments
LEFT JOIN employees
ON departments.department_id = employees.department_id;

→

department_id	department_name	employee_id	employee_name
1	Sales	101	Shraddha
2	Human Resources	102	Tisha
3	IT	103	Charu
1	Sales	104	Dharmi

12. SQL Group By :

1) What is the GROUP BY clause in SQL? How is it used with aggregate functions?

Ans:-

→ The GROUP BY clause is a powerful tool in SQL that allows you to group rows of a result set based on one or more columns.

→ Divide a result set into groups based on one or more columns.

→ It's commonly used with aggregate functions to perform calculations on each group.

→ Apply aggregate functions to each group.

2) Explain the difference between GROUP BY and ORDER BY.

Ans:-

→ GROUP BY :

→Divide a result set into groups based on one or more columns.

→Grouping data to perform calculations or aggregations.

→Apply aggregate functions (e.g., SUM, COUNT, AVG) to each group.

→ ORDER BY :

→ Sort a result set in ascending or descending order.

→ Arrange rows in a specific order based on one or more columns.

→Sorting data to present it in a specific order.

LAB EXERCISES:

Lab 1: Group employees by department and count the number of employees in each department using GROUP BY.

Ans:-

```
→SELECT departments.department_name, COUNT(employees.employee_id)
  AS                                                  employee_count
  FROM                                                   departments
  LEFT                          JOIN                       employees
  ON    departments.department_id    =    employees.department_id
  GROUP BY departments.department_name;
→
```

department_name	employee_count
Human Resources	1
IT	1
Sales	2

Lab 2: Use the AVG aggregate function to find the average salary of employees in each department.

Ans:-

```
→SELECT department_id, AVG(salary) AS average_salary
  FROM employees
  GROUP BY department_id;
→
```

department_id	average_salary
1	*NULL*
2	*NULL*
3	*NULL*

13. SQL Stored Procedure :

1) What is a stored procedure in SQL, and how does it differ from a standard SQL query?

Ans:-

→ A stored procedure is a precompiled SQL program that performs a specific task or set of tasks.

→ It's a reusable code block that can be executed multiple times with different input parameters.

→ Stored procedures are compiled before execution, while standard SQL queries are compiled and executed on the fly.

→ Stored procedures are reusable, while standard SQL queries are typically executed once and then discarded.

→ Stored procedures can accept input parameters and return output parameters, while standard SQL queries do not support this.

2) Explain the advantages of using stored procedures.

Ans:-

→ Stored procedures are compiled and optimized before they're executed, making them faster than standard SQL queries.

→ Encapsulating sensitive data and logic within the procedure.

→ Breaking down complex logic into smaller, reusable modules.

→ Minimizing the amount of data that needs to be transmitted over the network.

→ Providing a single point of maintenance for complex logic.

→ Reducing the load on the database server.

→ Providing a reusable code block that can be executed multiple times.

→ Improving the ability to share code across different applications.

LAB EXERCISES:

Lab 1: Write a stored procedure to retrieve all employees from the employees table based on department.

Ans:-

```
→DELIMITER                                                    $$

CREATE PROCEDURE GetEmployeesByDepartment(IN department_id INT)
BEGIN
    SELECT                                                    *
    FROM                                                 employees
    WHERE         department_id         =         department_id;
END                                                           $$
```

```
DELIMITER ;
```

→

employee_id	employee_name	department_id	salary
101	Shraddha	1	*NULL*
102	Tisha	2	*NULL*
103	Charu	3	*NULL*
104	Dharmi	1	*NULL*

Lab 2: Write a stored procedure that accepts course_id as input and returns the course details.

Ans:-

```
→DELIMITER $$

CREATE PROCEDURE GetCourseDetails(IN course_id INT)
BEGIN
    SELECT *
    FROM courses
    WHERE course_id = course_id;
END $$

DELIMITER ;
```

→

course_id	course_name	course_duration
1	Introduction to Programming	4
3	Web Development	3
4	Artificial Intelligence	4

5	Data Science	4
6	Cybersecurity	3
7	Machine Learning	4
8	Cloud Computing	3

14. SQL View :

1) What is a view in SQL, and how is it different from a table?

Ans:-

→ a view is a virtual table that is based on the result of a query. It's a way to simplify complex queries and present data in a more meaningful way.

→ Tables are stored physically in the database, while views are not stored physically.

→ Tables can be modified directly, while views can only be modified by modifying the underlying tables.

→ Views can simplify complex queries by encapsulating the query logic within the view.

2) Explain the advantages of using views in SQL databases.

Ans:-

→ Simplified Complex Queries :

→ Encapsulating complex query logic within the view.

→ Providing a single, easy-to-use interface for querying the data.

→ Improved Data Security :

→ Providing a layer of abstraction between the physical tables and the application Limiting access to sensitive data.

→ Limiting access to sensitive data.

→ Data Abstraction :

→ Add or remove columns from the underlying tables without affecting the application.

→ Use different data sources or storage systems without affecting the application.

→ Reduced Data Redundancy :

→ Eliminating the need for duplicate data storage.

→ Providing a single source of truth for the data.

LAB EXERCISES:

Lab 1: Create a view to show all employees along with their department names.

Ans:-

```
→CREATE VIEW EmployeeDepartmentView AS
  SELECT e.employee_id, e.employee_name, e.salary, d.department_name
  FROM employees e
  JOIN departments d
  ON e.department_id = d.department_id;
→
```

department_id	department_name
1	Sales
2	Human Resources
3	IT

Lab 2: Modify the view to exclude employees whose salaries are below $50,000.

Ans:-

```
CREATE OR REPLACE VIEW EmployeeDepartmentView
AS
SELECT e.employee_id, e.employee_name,
e.salary, d.department_name
FROM employees e
JOIN departments d
ON e.department_id = d.department_id
WHERE e.salary >= 50000;
```

employee_id	employee_name	department_id	salary
101	Shraddha	1	*NULL*
102	Tisha	2	*NULL*
103	Charu	3	*NULL*
104	Dharmi	1	*NULL*

15. SQL Triggers

1) What is a trigger in SQL? Describe its types and when they are used.

Ans:-

→ A trigger is a stored procedure that is automatically executed when a specific event occurs, such as inserting, updating, or deleting data in a table.

→ After Trigger: Executed after the triggering event.

→ Before Trigger: Executed before the triggering event.

→ Instead of Trigger: Replaces the triggering event.

→ Row-Level Trigger: Executed for each row affected by the triggering event.

→ Statement-Level Trigger: Executed once for each triggering event, regardless of the number of rows affected.

2) Explain the difference between INSERT, UPDATE, and DELETE triggers?

Ans:-

→ Insert :----

→ Executed when a new row is inserted into a table.

→ Used to validate data before it is inserted, or to perform additional actions after the insertion.
→Can access the NEW table, which contains the values of the row being inserted.

→ Update:---
→ Executed when an existing row is updated in a table.
→Used to validate data before it is updated, or to perform additional actions after the update.
→Can access the OLD table, which contains the original values of the row being updated, and the NEW table, which contains the updated values.

→Delete :---
→Executed when a row is deleted from a table.
→Used to perform additional actions before or after the deletion, such as logging the deletion or updating related tables.
→Can access the OLD table, which contains the values of the row being deleted.

LAB EXERCISES:

Lab 1: Create a trigger to automatically log changes to the employees table when a new employee is added.

Ans:-

```
→DELIMITER                                                    $$

  CREATE                    TRIGGER                    LogNewEmployee
  AFTER               INSERT               ON               employees
  FOR                              EACH                              ROW
  BEGIN
      INSERT INTO employee_log (log_id, employee_id, employee_name,
  action,                                                 log_timestamp)
      VALUES  (NULL,  NEW.employee_id,  NEW.employee_name,  'INSERT',
  NOW());
  END                                                           $$

  DELIMITER ;
→
```

employee_id	employee_name	department_id	salary
101	Shraddha	1	*NULL*
102	Tisha	2	*NULL*
103	Charu	3	*NULL*
104	Dharmi	1	*NULL*

Lab 2: Create a trigger to update the last_modified timestamp whenever an employee record is updated.

Ans:-

```
→DELIMITER                                                    $$

  CREATE                    TRIGGER                  UpdateLastModified
  BEFORE              UPDATE               ON               employees
  FOR                              EACH                              ROW
  BEGIN
```

```
    SET NEW.last_modified = NOW();
END $$

DELIMITER ;
```

→

employee_id	employee_name	department_id	salary
101	Shraddha	1	*NULL*
102	Tisha	2	*NULL*
103	Charu	3	*NULL*
104	Dharmi	1	*NULL*

16. Introduction to PL/SQL

1) What is PL/SQL, and how does it extend SQL's capabilities?

Ans:-

→ PL/SQL (Procedural Language/Structured Query Language) is a procedural extension to the SQL language developed by Oracle Corporation.

→Procedural logic: PL/SQL allows you to write procedural code, including conditional statements, loops, and exception handling, which is not possible in standard SQL.

→Variables and data types: PL/SQL provides a range of data types, including numeric, character, and date types, which can be used to declare variables and store data.

→Control structures: PL/SQL includes control structures such as IF-THEN statements, CASE statements, and loops (FOR, WHILE, and LOOP), which enable you to control the flow of your program.

→Functions and procedures: PL/SQL allows you to create reusable functions and procedures, which can be called from other PL/SQL programs or from SQL statements.

→Triggers: PL/SQL enables you to create triggers, which are programs that are automatically executed in response to specific events, such as insert, update, or delete operations.

→Error handling: PL/SQL provides a robust error handling mechanism, which enables you to catch and handle exceptions, and to provide meaningful error messages.

→Integration with SQL: PL/SQL is tightly integrated with SQL, which means you can use SQL statements within PL/SQL programs, and vice versa.

2) List and explain the benefits of using PL/SQL?

Ans:-

→ By storing procedural logic in the database, you can reduce the amount of data that needs to be transferred between the database and the application.

→ PL/SQL enables you to encapsulate sensitive logic and data within the database, making it more difficult for unauthorized users to access or modify the data.

→PL/SQL provides a range of built-in features and tools, such as packages and procedures, which enable rapid development and deployment of database applications.

→PL/SQL is tightly integrated with the Oracle Database, which provides a range of benefits, including improved performance and security.

LAB EXERCISES:

Lab 1: Write a PL/SQL block to print the total number of employees from the employees table.

Ans:-

```
DECLARE
   total_employees NUMBER; -- Variable to store the total number of employees
BEGIN
   -- Query to count the total number of employees
```

```
   SELECT COUNT(*) INTO total employees FROM
employees;

   -- Display the total number of employees
   DBMS_OUTPUT.PUT_LINE('Total Number of
Employees: ' || total employees);
END;
/
```

Lab 2: Create a PL/SQL block that calculates the total sales from an orders table.

Ans:-

```
→DECLARE
      total sales NUMBER; -- Variable to store the total sales
   BEGIN
      --     Query     to     calculate     the     total     sales
      SELECT  SUM(sale  amount)  INTO  total  sales  FROM  orders;

      --          Display          the          total          sales
      DBMS_OUTPUT.PUT_LINE('Total  Sales:  '  ||  total  sales);
   END;
   /
```

17. PL/SQL Control Structures

1) What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.

Ans:-

→ IF-THEN Control Structure :

→IF-THEN control structure is used to execute a block of code if a certain condition is met.

→IF-THEN-ELSE Control Structure :

→IF-THEN-ELSE control structure is used to execute one block of code if a condition is met, and another block of code if the condition is not met.

→LOOP Control Structure :

→The LOOP control structure is used to repeat a block of code for a specified number of times.

→ 1) Basic LOOP :

→The basic LOOP control structure is used to repeat a block of code indefinitely.

→ 2) FOR LOOP :

→The FOR LOOP control structure is used to repeat a block of code for a specified number of times.

→3) WHILE LOOP :

→The WHILE LOOP control structure is used to repeat a block of code while a certain condition is met.

2) How do control structures in PL/SQL help in writing complex queries?

Ans:-

→ Conditional Statements :

→1. IF-THEN statements: Allow you to execute different blocks of code based on conditions, making it easier to handle complex logic.

→2. CASE statements: Enable you to perform different actions based on the value of a variable or expression, simplifying complex queries.

→ Loops :

→1. FOR loops: Allow you to iterate over a range of values, making it easier to perform repetitive tasks.

→2. WHILE loops: Enable you to repeat a block of code while a condition is met, helping you to handle complex logic.

→Exception Handling :

→1. TRY-CATCH blocks: Allow you to handle errors and exceptions in a controlled way, making it easier to write robust and reliable code.

LAB EXERCISES:

Lab 1: Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

Ans:-

```
→ DECLARE
      employee_id NUMBER := 101; -- Replace with the desired
   employee ID
      employee_department VARCHAR2(50); -- Variable to store the
   department name
   BEGIN
      -- Fetch the department of the specified employee
      SELECT department_name INTO employee_department
      FROM employees
      WHERE employee_id = employee_id;

      -- Use an IF-THEN condition to check the department
      IF employee_department = 'Sales' THEN
         DBMS_OUTPUT.PUT_LINE('The employee works in the Sales
   department.');
      ELSIF employee_department = 'HR' THEN
         DBMS_OUTPUT.PUT_LINE('The employee works in the HR
   department.');
      ELSE
         DBMS_OUTPUT.PUT_LINE('The employee works in another
   department: ' || employee_department);
      END IF;
   END;
```

Lab 2: Use a FOR LOOP to iterate through employee records and display their names.

Ans:-

```
→employee records = [
      {"name": "Amit", "id": 101, "department": "HR"},
      {"name": "Neha", "id": 102, "department": "Finance"},
      {"name": "Raj", "id": 103, "department": "IT"}
  ]

  # Using a FOR LOOP to display employee names
  print("Employee Names:")
  for employee in employee records:
      print(employee["name"])
```

18. SQL Cursors

1) What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.

Ans:-

→ Implicit Cursors :

→ Implicit cursors are automatically created by Oracle when you execute a SQL statement that returns only one row.

→The most common implicit cursor is the SQL cursor, which is used to retrieve the result of a SELECT statement.

→INSERT, UPDATE, and DELETE statements.

→SELECT statements that return only one row.

→Explicit Cursors :

→Explicit cursors are user-defined cursors that you create explicitly using the CURSOR keyword.

→They are used to retrieve multiple rows from a query result set.

→SELECT statements that return multiple rows.

→Complex queries that require processing multiple rows.

2) When would you use an explicit cursor over an implicit one?

Ans:-

→1) Multi-Row Operations :---

→ Use an explicit cursor when you need to process multiple rows returned by a query.

→2) Complex Query Processing :---

→Explicit cursors provide more control over complex query processing.

→Processing rows in a specific order.

→3) Row-by-Row Processing :--

→Use an explicit cursor when you need to process each row individually.

→Performing validation or checking constraints on each row.

→4) Large Result Sets :---

→ Explicit cursors can be more efficient when dealing with large result sets.

→Allow for fetching and processing rows in batches.

→5) Reusability and Modularity :---

→Explicit cursors can be defined as reusable modules, making it easier.

→Share and reuse cursor definitions across multiple procedures.

LAB EXERCISES:

Lab 1: Write a PL/SQL block using an explicit cursor to retrieve and display employee details

Ans:-

→
```
DECLARE
    -- Declare the cursor to fetch employee details
```

```
  CURSOR emp_cursor IS
    SELECT emp_id, emp_name, emp_department
    FROM employee;

  -- Variables to hold employee details
  v_emp_id employee.emp_id%TYPE;
  v_emp_name employee.emp_name%TYPE;
  v_emp_department employee.emp_department%TYPE;
BEGIN
  -- Open the cursor and fetch each record
  OPEN emp_cursor;

  LOOP
    FETCH emp_cursor INTO v_emp_id, v_emp_name, v_emp_department;

    -- Exit the loop when no more rows are found
    EXIT WHEN emp_cursor%NOTFOUND;

    -- Display employee details
    DBMS_OUTPUT.PUT_LINE('ID: ' || v_emp_id || ', Name: ' ||
v_emp_name || ', Department: ' || v_emp_department);
  END LOOP;

  -- Close the cursor
  CLOSE emp_cursor;
END;
/
```

Lab 2: Create a cursor to retrieve all courses and display them one by one.

Ans:-

```
→ DECLARE
    -- Declare the cursor to fetch course details
    CURSOR course_cursor IS
      SELECT course_id, course_name
```

```
    FROM                                                courses;

  --      Variables      to      hold      course      details
  v_course_id                             courses.course_id%TYPE;
  v_course_name                         courses.course_name%TYPE;
BEGIN
  --                Open                the                cursor
  OPEN                        course                        cursor;

  LOOP
    --     Fetch     each     record     into     the     variables
    FETCH   course   cursor   INTO   v_course_id,   v_course_name;

    --   Exit   the   loop   when   no   more   rows   are   found
    EXIT               WHEN               course_cursor%NOTFOUND;

    --              Display             course             details
    DBMS_OUTPUT.PUT_LINE('Course ID: ' || v_course_id || ', Course
Name:               '               ||               v_course_name);
  END                                                          LOOP;

  --                Close                the                cursor
  CLOSE                                              course_cursor;
END;
```

19. Rollback and Commit Save point

1) Explain the concept of SAVEPOINT in transaction management. How do ROLLBACK and COMMIT interact with save points?

Ans:-

→transaction management, a SAVEPOINT is a temporary marker within a transaction that allows you to roll back

to a specific point in the transaction, rather than rolling back the entire transaction.

→1) ROLLBACK :--

→If you execute a ROLLBACK statement without specifying a SAVEPOINT, the entire transaction is rolled back, and all changes are discarded.

→If you specify a SAVEPOINT, the transaction is rolled back to that point, and all changes made after that SAVEPOINT are discarded.

→2) COMMIT :--

→When you execute a COMMIT statement, the entire transaction is committed, including all changes made since the beginning of the transaction.

→SAVEPOINTS do not affect the COMMIT statement.

2) When is it useful to use save points in a database transaction?

Ans:-

→A transaction involves multiple operations, save points can help you roll back to a specific point in case of an error, rather than rolling back the entire transaction.

→Save points allow you to test and validate parts of a transaction without committing the entire transaction.

→If the test fails, you can roll back to the save point and try again.

→You can roll back to a save point within an inner transaction without affecting the outer transaction.

→long-running transactions, save points can help you recover from failures or errors that occur during the transaction.

→Save points can be used to create audit points within a transaction.

→By rolling back to a save point, you can undo changes and maintain a consistent audit trail.

LAB EXERCISES:

Lab 1: Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.

Ans:-

```
→BEGIN
    -- Insert a record into the table
    INSERT INTO employees (emp_id, emp_name, emp_department)
    VALUES (201, 'Ravi', 'HR');

    -- Create a savepoint
    SAVEPOINT before_second_insert;

    -- Insert another record into the table
    INSERT INTO employees (emp_id, emp_name, emp_department)
    VALUES (202, 'Anjali', 'Finance');

    -- Rollback to the savepoint
    ROLLBACK TO before_second_insert;

    -- Commit the transaction to save changes before the savepoint
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Transaction rolled back to the savepoint,
  and changes before the savepoint have been committed.');
  END;
  /
```

Lab 2: Commit part of a transaction after using a savepoint and then rollback the remaining changes.

Ans:-

```
→BEGIN
    -- Insert the first record
    INSERT INTO employees (emp_id, emp_name, emp_department)
    VALUES (301, 'Karan', 'Marketing');

    -- Create a savepoint
    SAVEPOINT before_second_insert;
```

```
  -- Insert the second record
  INSERT INTO employees (emp_id, emp_name, emp_department)
  VALUES (302, 'Priya', 'Sales');

  -- Commit changes up to the savepoint
  COMMIT;

  DBMS_OUTPUT.PUT_LINE('Changes up to the savepoint have been
committed.');

  -- Insert the third record
  INSERT INTO employees (emp_id, emp_name, emp_department)
  VALUES (303, 'Alok', 'Operations');

  -- Rollback the remaining changes
  ROLLBACK;

  DBMS_OUTPUT.PUT_LINE('Changes after the savepoint have been
rolled back.');
END;
/
```

EXTRA LAB PRACTISE FOR DATABASE CONCEPTS :--

1. Introduction to SQL

LAB EXERCISES:

Lab 3: Create a database called library_db and a table books with columns: book_id, title, author, publisher,

year_of_publication, and price. Insert five records into the table.

Ans:-

→CREATE TABLE books (book_id INT AUTO_INCREMENT PRIMARY KEY, title text, author text, publisher text, year_of_publication YEAR, price DECIMAL(10, 2));

→
```
INSERT INTO books (title, author, publisher,
year_of_publication, price)
VALUES
('To Kill a Mockingbird', 'Harper Lee', 'J.B. Lippincott & Co.',
1960, 399.99),
('1984', 'George Orwell', 'Secker & Warburg', 1949, 299.50),
('The Great Gatsby', 'F. Scott Fitzgerald', 'Charles Scribner\'s
Sons', 1925, 349.99),
('Pride and Prejudice', 'Jane Austen', 'T. Egerton', 1813,
199.99),
('The Catcher in the Rye', 'J.D. Salinger', 'Little, Brown and
Company', 1951, 399.00);
```

→

book_id	title	author	publisher	year_of_publication	price
1	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99
2	1984	George Orwell	Secker & Warburg	1949	299.50
3	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99
4	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99
5	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00

Lab 4: Create a table members in library_db with columns: member_id, member_name, date_of_membership, and email. Insert five records into this table.

Ans:-

→CREATE TABLE members (member_id INT AUTO_INCREMENT PRIMARY KEY, member_name text, date_of_membership DATE , email text);

→
```
INSERT INTO members (member_name, date_of_membership, email)
VALUES
('Alice Johnson', '2025-01-15', 'alice.johnson@example.com'),
('Bob Smith', '2025-02-10', 'bob.smith@example.com'),
('Charlie Brown', '2025-03-05', 'charlie.brown@example.com'),
('Diana White', '2025-03-20', 'diana.white@example.com'),
('Eve Black', '2025-04-01', 'eve.black@example.com');
```

→

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	alice.johnson@example.com
2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com
4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

2. SQL Syntax

LAB EXERCISES:

Lab 3: Retrieve all members who joined the library before 2022. Use appropriate SQL syntax with WHERE and ORDER BY.

Ans:-

→
```
SELECT member_id, member_name, date_of_membership, email
FROM members
WHERE date_of_membership < '2022-01-01'
ORDER BY date_of_membership ASC;
```
→

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	alice.johnson@example.com
2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com
4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

Lab 4: Write SQL queries to display the titles of books published by a specific author. Sort the results by year_of_publication in descending order.

Ans:-

→
```
SELECT title
FROM books
```

```
WHERE author = 'Specific Author'
ORDER BY year_of_publication DESC;
```

→

				book_id	title	author	publisher	year_of_publication	price
				1	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99
				2	1984	George Orwell	Secker & Warburg	1949	299.50
				3	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99
				4	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99
				5	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00

3. SQL Constraints

LAB EXERCISES:

Lab 3: Add a CHECK constraint to ensure that the price of books in the books table is greater than 0.

Ans:-

Lab 4: Modify the members table to add a UNIQUE constraint on the email column, ensuring that each member has a unique email address.

Ans:-

```
→     ALTER TABLE members
ADD CONSTRAINT unique_email
UNIQUE (email);

→     SELECT email, COUNT(*)
FROM members
GROUP BY email
HAVING COUNT(*) > 1;

→
```

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	alice.johnson@example.com
2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com
4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

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

LAB EXERCISES:

Lab 3: Create a table authors with the following columns: author_id, first_name, last_name, and country. Set author_id as the primary key.

Ans:-

→CREATE TABLE authors (author_id INT AUTO_INCREMENT PRIMARY KEY, first_name text, last_name text, country VARCHAR(100));

Lab 4: Create a table publishers with columns: publisher_id, publisher_name, contact_number, and address. Set publisher_id as the primary key and contact_number as unique.

Ans:-

→CREATE TABLE publishers (publisher_id INT AUTO_INCREMENT PRIMARY KEY, publisher_name text, contact_number text UNIQUE, address text);

→ INSERT INTO publishers VALUES(201,'Aayushi',34671298,'Ahemdabad'),(202,'Mahi',34623298,'Aanand'),(203,'Dipali',34671266,'Gandhinagar'),(204,'Reena',89671298,'Mumbai'),(205,'Bhavi',34658798,'Rajkot');

→

publisher_id	publisher_name	contact_number	address
201	Aayushi	34671298	Ahmedabad
202	Mahi	34623298	Aanand
203	Dipali	34671266	Gandhinagar

204	Reena	89671298	Mumbai
205	Bhavi	34658798	Rajkot

5. ALTER Command

LAB EXERCISES:

Lab 3: Add a new column genre to the books table. Update the genre for all existing records.

Ans:-

→
```
ALTER TABLE books
ADD genre VARCHAR(100);
```

→
```
UPDATE books
SET genre = CASE
    WHEN title = 'To Kill a Mockingbird' THEN 'Fiction'
    WHEN title = '1984' THEN 'Dystopian'
    WHEN title = 'The Great Gatsby' THEN 'Classic'
    WHEN title = 'Pride and Prejudice' THEN 'Romance'
    WHEN title = 'The Catcher in the Rye' THEN 'Young Adult'
    ELSE 'Unknown'
END;
```

→

book_id	title	author	publisher	year_of_publication	price	genre
1	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	Fiction
2	1984	George Orwell	Secker & Warburg	1949	299.50	Dystopian

3	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	Classic
4	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	Romance
5	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	Young Adult

Lab 4: Modify the members table to increase the length of the email column to 100 characters.

Ans:-

→
```
ALTER TABLE members
MODIFY email VARCHAR(100);
```
→

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	alice.johnson@example.com
2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com
4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

6. DROP Command

LAB EXERCISES:

Lab 3: Drop the publishers table from the database after verifying its structure.

Ans:-

→ DESCRIBE publishers;

→

Field	Type	Null	Key	Default	Extra
publisher_id	int(11)	NO	PRI	*NULL*	auto_increment
publisher_name	text	YES		*NULL*	
contact_number	text	YES	UNI	*NULL*	
address	text	YES		*NULL*	

→ DROP TABLE publishers;

Lab 4: Create a backup of the members table and then drop the original members table.

Ans:-

→ CREATE TABLE members_backup AS
SELECT * FROM members;

→ SELECT * FROM members_backup;

→

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	alice.johnson@example.com
2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com
4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

→ DROP TABLE members;

7. Data Manipulation Language (DML)

LAB EXERCISES:

Lab 4: Insert three new authors into the authors table, then update the last name of one of the authors.

Ans:-

```
→     INSERT INTO authors (first_name, last_name, country)
  VALUES
  ('John', 'Doe', 'USA'),
  ('Emily', 'Smith', 'UK'),
  ('Rahul', 'Sharma', 'India');

→     UPDATE authors
  SET last_name = 'Johnson'
  WHERE first_name = 'Emily' AND last_name = 'Smith';

→     SELECT * FROM authors;

→
```

author_id	first_name	last_name	country
1	John	Doe	USA
2	Emily	Johnson	UK
3	Rahul	Sharma	India

Lab 5: Delete a book from the books table where the price is higher than $100.

Ans:-

→
```
DELETE FROM books
WHERE price > 100;
```

8. UPDATE Command

LAB EXERCISES:

Lab 3: Update the year_of_publication of a book with a specific book_id.

Ans:-

→ UPDATE books SET year_of_publication = 2021 WHERE book_id = 1;

→ book_id, title, author, publisher, year_of_publication, price, genre.

→

book_id	title	author	publisher	year_of_publication	price	genre
1	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	Fiction
2	1984	George Orwell	Secker & Warburg	1949	299.50	Dystopian
3	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	Classic
4	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	Romance
5	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	Young Adult

Lab 4: Increase the price of all books published before 2015 by 10%.

Ans:-

```
→     UPDATE books
  SET price = price * 1.10
  WHERE year_of_publication < 2015;
→
```

book_id	title	author	publisher	year_of_publication	price	genre
1	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	Fiction
2	1984	George Orwell	Secker & Warburg	1949	299.50	Dystopian
3	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	Classic
4	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	Romance
5	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	Young Adult

9. DELETE Command

LAB EXERCISES:

Lab 3: Remove all members who joined before 2020 from the members table.

Ans:-

```
→     DELETE FROM members
  WHERE date_of_membership < '2020-01-01';
→
```

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	alice.johnson@example.com
2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com
4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

Lab 4: Delete all books that have a NULL value in the author column.

Ans:-

```
→     DELETE FROM books
  WHERE author IS NULL;
→     SELECT * FROM books
  WHERE author IS NULL;
→     SELECT * FROM books;
```

10. Data Query Language (DQL)

LAB EXERCISES:

Lab 4: Write a query to retrieve all books with price between $50 and $100.

Ans:-

→
```
SELECT *
FROM books
WHERE price BETWEEN 50 AND 100;
```
→

book_id	title	author	publisher	year_of_publication	price	genre
6	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	*NULL*
7	1984	George Orwell	Secker & Warburg	1949	299.50	*NULL*
8	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	*NULL*
9	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	*NULL*
10	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	*NULL*

Lab 5: Retrieve the list of books sorted by author in ascending order and limit the results to the top 3 entries.

Ans:-

→
```
SELECT *
FROM books
ORDER BY author ASC
LIMIT 3;
```
→

book_id	title	author_1	publisher	year_of_publication	price	genre
8	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	*NULL*
7	1984	George Orwell	Secker & Warburg	1949	299.50	*NULL*
6	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	*NU*

→
```
SELECT title, author
FROM books
ORDER BY author ASC
LIMIT 3;
```
→

title	author_1
The Great Gatsby	F. Scott Fitzgerald
1984	George Orwell
To Kill a Mockingbird	Harper Lee

11. Data Control Language (DCL)

LAB EXERCISES:

Lab 3: Grant SELECT permission to a user named librarian on the books table.

Ans:-

→
```
GRANT SELECT ON books TO librarian;
```

→

book_id	title	author	publisher	year_of_publication	price	genre
6	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	*NULL*
7	1984	George Orwell	Secker & Warburg	1949	299.50	*NULL*
8	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	*NULL*
9	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	*NULL*
10	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	*NULL*

Lab 4: Grant INSERT and UPDATE permissions to the user admin on the members table.

Ans:-

→ GRANT INSERT, UPDATE ON members_backup TO admin;

→

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	alice.johnson@example.com
2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com

4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

12. REVOKE Command

LAB EXERCISES:

Lab 3: Revoke the INSERT privilege from the user librarian on the books table.

Ans:-

→ `REVOKE INSERT ON books FROM librarian;`

→

book_id	title	author	publisher	year_of_publication	price	genre
6	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	*NULL*
7	1984	George Orwell	Secker & Warburg	1949	299.50	*NULL*
8	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	*NULL*
9	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	*NULL*
10	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	*NU*

Lab 4: Revoke all permissions from user admin on the members table.

Ans:-

→ REVOKE ALL PRIVILEGES ON members_backup FROM admin;

→

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	alice.johnson@example.com
2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com
4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

13. Transaction Control Language (TCL)

LAB EXERCISES:

Lab 3: Use COMMIT after inserting multiple records into the books table, then make another insertion and perform a ROLLBACK.

Ans:-

→ INSERT INTO books (title, author, price, year_of_publication) VALUES ('Book 1', 'Author A', 50, 2018);

→ INSERT INTO books (title, author, price, year_of_publication) VALUES ('Book 2', 'Author B', 70, 2016);

→ INSERT INTO books (title, author, price, year_of_publication) VALUES ('Book 3', 'Author C', 90, 2020);

→

book_id	title	author	publisher	year_of_publication	price	genre
6	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	*NULL*
7	1984	George Orwell	Secker & Warburg	1949	299.50	*NULL*
8	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	*NULL*
9	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	*NULL*
10	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	*NULL*
11	Book 1	Author A	*NULL*	2018	50.00	*NULL*
12	Book 2	Author B	*NULL*	2016	70.00	*NULL*
13	Book 3	Author C	*NULL*	2020	90.00	*NULL*

→

```
INSERT INTO books (title, author, price, year_of_publication)
VALUES ('Book 4', 'Author D', 120, 2021);
```

→

Book 1	Author A	*NULL*	2018	50.00	*NULL*	
12	Book 2	Author B	*NULL*	2016	70.00	*NULL*
13	Book 3	Author C	*NULL*	2020	90.00	*NULL*
14	Book 4	Author D	*NULL*	2021	120.00	*NULL*

Lab 4: Set a SAVEPOINT before making updates to the members table, perform some updates, and then roll back to the SAVEPOINT.

Ans:-

→ UPDATE members_backup

→ SET email = 'newemail1@example.com'

→ WHERE member_id = 1;

→

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	newemail1@example.com
2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com
4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

14. SQL Joins

LAB EXERCISES:

Lab 3: Perform an INNER JOIN between books and authors tables to display the title of books and their respective authors' names.

Ans:-

→

```
SELECT books.title AS book_title,
  authors.first_name || ' ' || authors.last_name AS
author_name
FROM books
INNER JOIN authors
ON books.author_id = authors.author_id;
```

→

book_id	title	author	publisher	year_of_publication	price	genre
6	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	*NULL*
7	1984	George Orwell	Secker & Warburg	1949	299.50	*NULL*
8	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	*NULL*
9	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	*NULL*
10	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	*NULL*
11	Book 1	Author A	*NULL*	2018	50.00	*NULL*
12	Book 2	Author B	*NULL*	2016	70.00	*NULL*
13	Book 3	Author C	*NULL*	2020	90.00	*NULL*
14	Book 4	Author D	*NULL*	2021	120.00	*NULL*

Lab 4: Use a FULL OUTER JOIN to retrieve all records from the books and authors tables, including those with no matching entries in the other table.

Ans:-

→
```
SELECT
    books.title AS book_title,
    authors.first_name || ' ' || authors.last_name AS
author_name,
    books.price,
    authors.country
FROM books
FULL OUTER JOIN authors
ON books.author_id = authors.author_id;
```

15. SQL Group By

LAB EXERCISES:

Lab 3: Group books by genre and display the total number of books in each genre.

Ans:-

→
```
SELECT genre, COUNT(*) AS total books
FROM books
GROUP BY genre;
```

→

genre	total books
NULL	9

Lab 4: Group members by the year they joined and find the number of members who joined each year.

Ans:-

→
```
SELECT EXTRACT(YEAR FROM date_of_membership) AS join year,
  COUNT(*) AS total members
FROM members
GROUP BY EXTRACT(YEAR FROM date_of_membership)
ORDER BY join year;
```

16. SQL Stored Procedure

LAB EXERCISES:

Lab 3: Write a stored procedure to retrieve all books by a particular author.

Ans:-

→
```
CREATE OR REPLACE PROCEDURE GetBooksByAuthor (
    p_author_name IN VARCHAR2
)
IS
    tittle books.title%TYPE;
    price books.price%TYPE;
    year books.year_of_publication%TYPE;
BEGIN
    -- Cursor to retrieve books by the given author
```

```
    FOR book_record IN (
        SELECT title, price, year_of_publication
        FROM books
        WHERE author = p_author_name
    ) LOOP
        -- Output book details
        DBMS_OUTPUT.PUT_LINE('Title: ' || book_record.title ||
                             ', Price: $' || book_record.price ||
                             ', Year: ' ||
book_record.year_of_publication);
    END LOOP;
END;
/
```

Lab 4: Write a stored procedure that takes book_id as an argument and returns the price of the book

Ans:-

```
→ DELIMITER //

  CREATE PROCEDURE GetBookPrice(
      IN book_id INT,
      OUT book price DECIMAL(10, 2)
  )
  BEGIN
      SELECT price INTO book price
      FROM books
      WHERE id = book_id;
  END //

  DELIMITER ;

→
```

book_id	title	author	publisher	year_of_publication	price	genre

6	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	*NULL*
7	1984	George Orwell	Secker & Warburg	1949	299.50	*NULL*
8	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	*NULL*
9	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	*NULL*
10	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	*NULL*
11	Book 1	Author A	*NULL*	2018	50.00	*NULL*
12	Book 2	Author B	*NULL*	2016	70.00	*NULL*
13	Book 3	Author C	*NULL*	2020	90.00	*NULL*
14	Book 4	Author D	*NULL*	2021	120.00	*NULL*

17. SQL View

LAB EXERCISES:

Lab 3: Create a view to show only the title, author, and price of books from the books table.

Ans:-

→
```
CREATE VIEW Book Details AS
SELECT title, author, price
FROM
```
→

book_id	title	author	publisher	year_of_publication	price	genre

6	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	*NULL*
7	1984	George Orwell	Secker & Warburg	1949	299.50	*NULL*
8	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	*NULL*
9	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	*NULL*
10	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	*NULL*
11	Book 1	Author A	*NULL*	2018	50.00	*NULL*
12	Book 2	Author B	*NULL*	2016	70.00	*NULL*
13	Book 3	Author C	*NULL*	2020	90.00	*NULL*
14	Book 4	Author D	*NULL*	2021	120.00	*NULL*

Lab 4: Create a view to display members who joined before 2020.

Ans:-

→
```
CREATE VIEW MembersBefore2020 AS
SELECT member_id, name, join date
FROM members
WHERE join date < '2020-01-01';
```
→

member_id	member_name	date_of_membership	email
1	Alice Johnson	2025-01-15	newemail1@example.com

2	Bob Smith	2025-02-10	bob.smith@example.com
3	Charlie Brown	2025-03-05	charlie.brown@example.com
4	Diana White	2025-03-20	diana.white@example.com
5	Eve Black	2025-04-01	eve.black@example.com

18. SQL Trigger

LAB EXERCISES:

Lab 3: Create a trigger to automatically update the last_modified timestamp of the books table whenever a record is updated.

Ans:-

→

```
DELIMITER //

CREATE TRIGGER UpdateLastModified
BEFORE UPDATE ON books
FOR EACH ROW
BEGIN
    SET NEW.last_modified = NOW();
END //

DELIMITER ;
```

Lab 4: Create a trigger that inserts a log entry into a log changes table whenever a DELETE operation is performed on the books table.

Ans:-

```
→ DELIMITER //

  CREATE TRIGGER LogDeleteOperation
  AFTER DELETE ON books
  FOR EACH ROW
  BEGIN
      INSERT INTO log changes (book_id, action, change time)
      VALUES (OLD.id, 'DELETE', NOW());
  END //

  DELIMITER ;
```

→

book_id	title	author	publisher	year_of_publication	price	genre
6	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	399.99	*NULL*
7	1984	George Orwell	Secker & Warburg	1949	299.50	*NULL*
8	The Great Gatsby	F. Scott Fitzgerald	Charles Scribner's Sons	1925	349.99	*NULL*
9	Pride and Prejudice	Jane Austen	T. Egerton	0000	199.99	*NULL*
10	The Catcher in the Rye	J.D. Salinger	Little, Brown and Company	1951	399.00	*NULL*
11	Book 1	Author A	*NULL*	2018	50.00	*NULL*
12	Book 2	Author B	*NULL*	2016	70.00	*NULL*

13	Book 3	Author C	*NULL*	2020	90.00	*NULL*
14	Book 4	Author D	*NULL*	2021	120.00	*NULL*

19. Introduction to PL/SQL

LAB EXERCISES:

Lab 3: Write a PL/SQL block to insert a new book into the books table and display a confirmation message.

Ans:-

→
```
DECLARE
    tittle VARCHAR2(100) := 'The Great Adventure';
    v_author VARCHAR2(100) := 'John Doe';
    v_price NUMBER := 499.99;
BEGIN
    -- Insert the new book into the books table
    INSERT INTO books (title, author, price)
    VALUES (tittle, v_author, price);

    -- Display a confirmation message
    DBMS_OUTPUT.PUT_LINE('New book "' || tittle || '" by ' ||
v_author || ' has been added with a price of ' || v_price ||
'.');
END;
/
```

Lab 4: Write a PL/SQL block to display the total number of books in the books table.

Ans:-

```
→     DECLARE
      total books NUMBER;
BEGIN

    SELECT COUNT(*) INTO total books
    FROM books;


    DBMS_OUTPUT.PUT_LINE('Total number of books in the books
table: ' || total books);
END;
/
```

20. PL/SQL Syntax

LAB EXERCISES:

Lab 3: Write a PL/SQL block to declare variables for book_id and price, assign values, and display the results.

Ans:-

```
→     DECLARE
      book_id NUMBER := 101;  -- Declaring and assigning a value to
book_id
      price NUMBER := 499.99; -- Declaring and assigning a value to
price
BEGIN

    DBMS_OUTPUT.PUT_LINE('Book ID: ' || book_id);
    DBMS_OUTPUT.PUT_LINE('Price: ' || price);
```

```
END;
/
```

Lab 4: Write a PL/SQL block using constants and perform arithmetic operations on book prices.

Ans:-

→

```
DECLARE

    c_discount_rate CONSTANT NUMBER := 0.1; -- 10% discount
    c_tax_rate CONSTANT NUMBER := 0.05; -- 5% tax


    original price NUMBER := 500; -- Original price of the book
    discounted_price NUMBER;
    final price NUMBER;
BEGIN

    discounted_price := original price - (original price *
c_discount_rate); -- Apply discount
    final price := discounted_price + (discounted_price *
c_tax_rate); -- Add tax


    DBMS_OUTPUT.PUT_LINE('Original Price: ' || original price);
    DBMS_OUTPUT.PUT_LINE('Discounted Price: ' ||
```

21. PL/SQL Control Structures

LAB EXERCISES:

Lab 3: Write a PL/SQL block using IF-THEN-ELSE to check if a book's price is above $100 and print a message accordingly.

Ans:-

```
DECLARE
    book price NUMBER := 120; -- Example
price for the book
BEGIN
    -- Check if the book's price is above
$100
    IF book price > 100 THEN
        DBMS_OUTPUT.PUT_LINE('The book is
expensive. Its price is $' || book price
|| '.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('The book is
affordable. Its price is $' || book price
|| '.');
    END IF;
END;
```

Lab 4: Use a FOR LOOP in PL/SQL to display the details of all books one by one.

Ans:-

→

```
DECLARE

    CURSOR book cursor IS
        SELECT title, author, price FROM books;
BEGIN

    FOR bookrack IN book cursor LOOP
        DBMS_OUTPUT.PUT_LINE('Title: ' || book_rec.title || ',
Author: ' || book_rec.author || ', Price: $' || book_rec.price);
    END LOOP;
END;
/
```

22. SQL Cursors

LAB EXERCISES:

Lab 3: Write a PL/SQL block using an explicit cursor to fetch and display all records from the members table.

Ans:-

→

```
DECLARE

   CURSOR members_cursor IS
      SELECT * FROM members;


   member_record members%ROWTYPE;
BEGIN

   OPEN members_cursor;
```

```
   LOOP
      FETCH members_cursor INTO member_record;


      EXIT WHEN members_cursor%NOTFOUND;


      DBMS_OUTPUT.PUT_LINE('Member ID: ' ||
member_record.member_id ||
                           ', Name: ' || member_record.name ||
                           ', Address: ' ||
member_record.address);
   END LOOP;


   CLOSE members cursor;
END;
/
```

Lab 4: Create a cursor to retrieve books by a particular author and display their titles.

Ans:-

```
→ DECLARE
       CURSOR books cursor (author_name VARCHAR2) IS

   book_title books.title%TYPE;
BEGIN

   OPEN books cursor('J.K. Rowling'); -- Replace 'J.K. Rowling'
with the desired author's name


   LOOP
      FETCH books cursor INTO book_title;
```

```
        EXIT WHEN books_cursor%NOTFOUND;


        DBMS_OUTPUT.PUT_LINE('Book Title: ' || book_title);
    END LOOP;

    CLOSE books cursor;
END;
/
```

23. Rollback and Commit Savepoint

LAB EXERCISES:

Lab 3: Perform a transaction that includes inserting a new member, setting a SAVEPOINT, and rolling back to the savepoint after making updates.

Ans:-

```
→BEGIN

    INSERT INTO members (member_id, member_name, member role)
    VALUES (101, 'Sonia', 'Manager');


    DBMS_OUTPUT.PUT_LINE('Inserted new member: Sonia');


    SAVEPOINT after insertion;

    UPDATE members
```

```
  SET member role = 'Director'
  WHERE member_id = 101;

  DBMS_OUTPUT.PUT_LINE('Updated member role to Director.');

  -- Rollback to the savepoint
  ROLLBACK TO after insertion;


  DBMS_OUTPUT.PUT_LINE('Rolled back to the savepoint. Member role
remains as initially inserted.');


  COMMIT;

  DBMS_OUTPUT.PUT_LINE('Transaction completed.');
END;
/
```

Lab 4: Use COMMIT after successfully inserting multiple books into the books table, then use ROLLBACK to undo a set of changes made after a savepoint.

Ans:-

→ BEGIN

```
  INSERT INTO books (book_id, book_title, author, genre)
  VALUES (1, 'The Great Adventure', 'John Doe', 'Fiction');


  INSERT INTO books (book_id, book_title, author, genre)
  VALUES (2, 'Learning SQL', 'Jane Smith', 'Education');
```

```
  COMMIT;

  DBMS_OUTPUT.PUT_LINE('Inserted multiple books and committed
successfully.');


  INSERT INTO books (book_id, book_title, author, genre)
  VALUES (3, 'Advanced PL/SQL', 'Alice Brown', 'Education');


  SAVEPOINT after_third_insert;


  INSERT INTO books (book_id, book_title, author, genre)
  VALUES (4, 'Exploring Databases', 'Mark Lee', 'Technology');


  INSERT INTO books (book_id, book_title, author, genre)
  VALUES (5, 'Database Design', 'Nina Roberts', 'Technology');


  ROLLBACK TO after_third_insert;

  DBMS_OUTPUT.PUT_LINE('Rolled back changes after the savepoint.
Only the third book remains.');

   COMMIT;

  DBMS_OUTPUT.PUT_LINE('Transaction completed.');
END;
```