Module 4 – Introduction to DBMS

Introduction to SQL

Theory Questions:

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

Ans: SQL (Structured Query Language) is a standard programming language used to communicate with and manage databases. It allows users to store, retrieve, update, and delete data in a relational database management system (RDBMS) such as MySQL, PostgreSQL, SQL Server, or Oracle.

Why SQL is Essential in Database Management

1. Data Retrieval and Manipulation
    - SQL allows users to easily query and extract specific data using commands like SELECT, INSERT, UPDATE, and DELETE.
2. Data Organization
    - It helps create and manage database structures such as tables, views, and indexes to organize data efficiently.
3. Data Integrity and Security
    - SQL supports constraints, permissions, and transactions to ensure that data remains accurate, consistent, and secure.
4. Scalability and Flexibility
    - SQL databases can handle large volumes of data and support complex queries, making them ideal for enterprise-level applications.
5. Standardization
    - SQL is a universal standard language, meaning it works across most database systems with only minor variations.
6. Integration
    - SQL integrates easily with programming languages (like Python, Java, or C++) and analytical tools, making it essential for full-stack development and data analysis.

2. Explain the difference between DBMS and RDBMS.

Ans-

DBMS	RDBMS
DBMS stores data as file.	RDBMS stores data in tabular form.
Data elements need to access individually.	Multiple data elements can be accessed at the same time.
No relationship between data.	Data is stored in the form of tables which are related to each other.
Normalization is not present.	Normalization is present.
DBMS does not support distributed database.	RDBMS supports distributed database.
It stores data in either a navigational or hierarchical form.	It uses a tabular structure where the headers are the column names, and the rows contain corresponding values.

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

Ans- 1. Data Definition (DDL – Data Definition Language)

SQL allows you to create and modify the structure of database objects such as tables, views, and indexes.
Examples:

CREATE TABLE Students (ID INT, Name VARCHAR(50), Age INT);

ALTER TABLE Students ADD Email VARCHAR(50);

DROP TABLE Students;

2. Data Manipulation (DML – Data Manipulation Language)

SQL is used to insert, update, delete, and retrieve data from tables.
Examples:

INSERT INTO Students VALUES (1, 'Parth', 20);

UPDATE Students SET Age = 21 WHERE ID = 1;

DELETE FROM Students WHERE ID = 1;

3. Data Querying (SELECT Statements)

The most common use of SQL is querying data — retrieving meaningful information from large datasets.
Example:

SELECT Name, Age FROM Students WHERE Age > 18;

4. Data Control (DCL – Data Control Language)

SQL manages user permissions and security within the database.
Examples:

GRANT SELECT ON Students TO user1;

REVOKE UPDATE ON Students FROM user2;

5. Transaction Control (TCL – Transaction Control Language)

SQL manages transactions — groups of operations that must be executed together.
Examples:

BEGIN TRANSACTION;

UPDATE Account SET Balance = Balance - 500 WHERE ID = 1;

UPDATE Account SET Balance = Balance + 500 WHERE ID = 2;

COMMIT; -- or ROLLBACK;

4. What are the key features of SQL?

Ans- 1. Data Definition

2. Data Manipulation

3. Data Querying

4. Data Security

5. Transaction Control

6. Data Integrity

7. Portability

8. High Performance with Large Data

9. Integration with Other Languages

10. Flexibility with Views and Joins

1-Introduction to SQL

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-

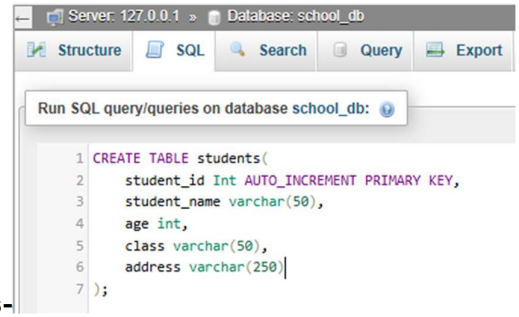


Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	student_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	student_name	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	age	int(11)			Yes	NULL			Change Drop More
4	class	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
5	address	varchar(250)	utf8mb4_general_ci		Yes	NULL			Change Drop More

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

Ans-

	student_id	student_name	age	class	address
Edit Copy Delete	1	Rahul Sharma	15	10A	Ahmedabad
Edit Copy Delete	2	Priya Patel	14	9B	Surat
Edit Copy Delete	3	Amit Mehta	16	11C	Rajkot
Edit Copy Delete	4	Neha Singh	15	10B	Vadodara
Edit Copy Delete	5	Karan Joshi	13	8A	Bhavnagar

2. SQL Syntax Theory Questions:

1. What are the basic components of SQL syntax?

Ans-Basic components include:

Component	Description	Example
Keywords	Reserved words used to perform operations.	SELECT, FROM, WHERE, INSERT, UPDATE, DELETE
Identifiers	Names of database objects such as tables, columns, or databases.	students, name, marks
Operators	Used to perform comparisons or logical operations.	=, >, <, AND, OR, NOT
Clauses	Parts of SQL statements that perform specific tasks.	WHERE, GROUP BY, ORDER BY
Expressions	Combination of values, operators, and functions that produce a value.	salary + bonus
Literals (Values)	Constant data values used in SQL.	'Parth', 100, '2025-10-17'
Comments	Used to add notes in SQL code.	-- This is a comment

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

Ans-General Syntax:

SELECT column1, column2, ...

FROM table_name

[WHERE condition]

[GROUP BY column]

[HAVING condition]

[ORDER BY column ASC|DESC];

Example:

SELECT name, marks

FROM students

WHERE marks > 60

ORDER BY marks DESC;

3. Explain the role of clauses in SQL statements.

Ans-

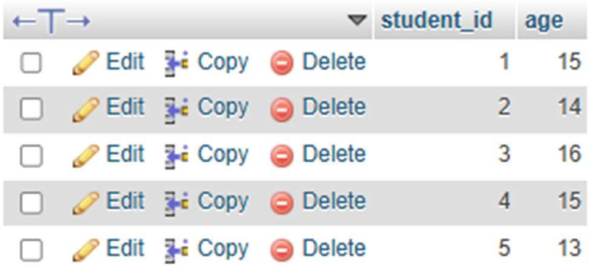
Clause	Role / Purpose	Example
SELECT	Specifies which columns to display.	SELECT name, age
FROM	Specifies the table(s) from which to retrieve data.	FROM students
WHERE	Filters rows based on given conditions.	WHERE age > 18
GROUP BY	Groups rows that have the same values in specified columns.	GROUP BY department
HAVING	Applies conditions to groups (used with GROUP BY).	HAVING AVG(salary) > 50000
ORDER BY	Sorts the result set in ascending or descending order.	

2. SQL Syntax

LAB EXERCISES:

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

Ans-



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

Ans-

3. SQL Constraints

Theory Questions:

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

Ans-Constraints in SQL are rules applied to table columns to ensure the accuracy, validity, and integrity of the data stored in the database.
They help maintain consistent and reliable data by preventing invalid entries.

Types of Constraints in SQL

Constraint	Description	Example
NOT NULL	Ensures that a column cannot have NULL (empty) values.	name VARCHAR(50) NOT NULL
UNIQUE	Ensures that all values in a column are unique (no duplicates).	email VARCHAR(100) UNIQUE
PRIMARY KEY	Uniquely identifies each record in a table. It is a combination of NOT NULL + UNIQUE.	customer_id INT PRIMARY KEY
FOREIGN KEY	Establishes a link between two tables using a key. Ensures referential integrity.	salesman_id INT REFERENCES Salesman(salesman_id)
CHECK	Ensures that values in a column satisfy a specific condition.	CHECK (age >= 18)
DEFAULT	Sets a default value for a column if no value is provided.	status VARCHAR(10) DEFAULT 'Active'

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

Ans-

Feature	PRIMARY KEY	FOREIGN KEY
Purpose	Uniquely identifies each record in a table.	Creates a relationship between two tables.
Uniqueness	Must be unique and NOT NULL.	Can contain duplicate and NULL values.
Table	Defined in the parent (main) table.	Defined in the child (referencing) table.
Number Allowed	Only one per table.	Can be many in a table.
Example	PRIMARY KEY (customer_id)	FOREIGN KEY (salesman_id) REFERENCES Salesman(salesman_id)

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

Ans-

Constraint	Role / Function	Example
NOT NULL	Prevents a column from having a NULL (empty) value. It ensures that every record must have a value.	name VARCHAR(50) NOT NULL → Every customer must have a name.
UNIQUE	Ensures that all values in a column are different (no duplicates allowed).	email VARCHAR(100) UNIQUE → No two customers can have the same email address.

3. SQL Constraints

• 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-

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	teacher_id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	teacher_name	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
3	subject	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More
4	email	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More

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

Ans-

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	student_id	int(11)			No	None			Change Drop More
2	student_name	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
3	class	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	teacher_id	int(11)			Yes	NULL			Change Drop More

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

Theory Questions:

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

Ans-Define the SQL Data Definition Language (DDL)

Definition:
DDL (Data Definition Language) is a category of SQL commands used to define, modify, and manage database structures such as databases, tables, indexes, and views.

DDL commands affect the structure (schema) of the database rather than the data itself.

Common DDL Commands:

Command	Description
CREATE	Creates a new database object (table, view, index, etc.).
ALTER	Modifies the structure of an existing object.
DROP	Deletes an existing object permanently.
TRUNCATE	Removes all records from a table but keeps its structure.
RENAME	Renames a database object (like a table).

2. Explain the CREATE command and its syntax.

Ans-(a) Create Database

```
CREATE DATABASE database_name;
```

Example:

```
CREATE DATABASE SchoolDB;
```

(b) Create Table

```
CREATE TABLE table_name (
   column1 datatype constraint,
   column2 datatype constraint,
);
```

Example:

```
CREATE TABLE Students (
   student_id INT PRIMARY KEY,
   student_name VARCHAR(50) NOT NULL,
   age INT CHECK (age >= 5),
   city VARCHAR(50) DEFAULT 'Unknown'
);
```

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

Ans-Purpose of Specifying Data Types and Constraints During Table Creation

When creating a table, specifying data types and constraints ensures data accuracy, integrity, and efficiency.

Component	Purpose
Data Types	Define the kind of data a column can store (e.g., integer, text, date). Helps in proper memory allocation and prevents invalid data.
Constraints	Enforce rules on data to maintain integrity and consistency. Prevents invalid or duplicate data entry.

Example:

```
CREATE TABLE Employees (
   emp_id INT PRIMARY KEY,          -- Data type + unique identifier
   emp_name VARCHAR(50) NOT NULL,    -- Text field, cannot be NULL
   salary DECIMAL(10,2) CHECK (salary > 0), -- Only positive salary allowed
   dept VARCHAR(30) DEFAULT 'HR'     -- Default value
);
```

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

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-

```sql
use collage;

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 university_db.

Ans-

5. ALTER Command

Theory Questions:

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

Ans-The ALTER command in SQL is part of the Data Definition Language (DDL) and is used to modify the structure of an existing table in a database — without deleting or recreating it.

You can use the ALTER command to:

- Add new columns
- Modify existing columns
- Delete (drop) columns
- Rename columns or the table itself
- Add or remove constraints

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

Ans-(a) Add a Column

To add a new column to an existing table.

Syntax:

ALTER TABLE table_name

ADD column_name datatype constraint;

Example:

ALTER TABLE Customer

ADD phone VARCHAR(15);

This adds a new column named phone to the Customer table.

(b) Modify an Existing Column

To change the data type, size, or constraint of an existing column.

Syntax:

ALTER TABLE table_name

MODIFY column_name new_datatype constraint;

(In SQL Server, use ALTER COLUMN instead of MODIFY.)

Example (MySQL):

ALTER TABLE Customer

MODIFY city VARCHAR(100);

This changes the size of the city column to 100 characters.

Example (SQL Server / Oracle):

ALTER TABLE Customer

ALTER COLUMN city VARCHAR(100);

(c) Drop (Delete) a Column

To remove a column from an existing table permanently.

Syntax:

ALTER TABLE table_name

DROP COLUMN column_name;

Example:

ALTER TABLE Customer

DROP COLUMN grade;

This removes the grade column from the Customer table.

5. ALTER Command

LAB EXERCISES:

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

Ans-

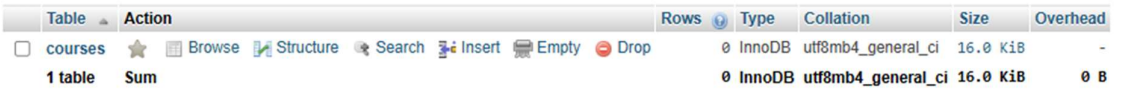


Table	Action	Rows	Type	Collation	Size	Overhead
courses	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_general_ci	16.0 KiB	-
1 table	Sum	0	InnoDB	utf8mb4_general_ci	16.0 KiB	0 B

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

Ans-

6. DROP Command

Theory Questions:

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

Ans-The DROP command in SQL is a Data Definition Language (DDL) command used to permanently delete database objects such as a table, view, index, or database from the system.

When you execute a DROP command, the entire structure and all data stored in that object are removed permanently — it cannot be recovered (unless a backup exists).

Syntax:

DROP TABLE table_name;

Example:

DROP TABLE Customer;

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

Ans-

Effect	Explanation
1. Permanent Data Loss	All the records (rows) stored in the table are deleted permanently.
2. Structure Removed	The table structure (column definitions, data types, constraints) is erased from the database.
3. Constraints Lost	Any PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, or DEFAULT constraints linked to the table are removed.
4. Relationships Broken	If other tables reference this table using FOREIGN KEYS, those relationships will be broken, possibly causing errors.
5. Space Freed	All storage space used by the table is released back to the database system.
6. Irreversible Action	The DROP operation cannot be rolled back (since DDL commands are auto-committed).

6. DROP Command

LAB EXERCISES:

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

Ans-

```
1 USE school_db;
2
3 DROP TABLE teachers;
4
```

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

```sql
USE school_db;

DROP TABLE students;

SHOW TABLES;

```

Ans-

7. Data Manipulation Language (DML)

Theory Questions:

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

Ans-INSERT Command:
Used to add new records (rows) into a table.
Syntax:

INSERT INTO table_name (column1, column2, ...)

VALUES (value1, value2, ...);

Example:

INSERT INTO students (id, name, age)

VALUES (1, 'Rahul', 20);

UPDATE Command:
Used to modify existing records in a table.
Syntax:

UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

Example:

UPDATE students

SET age = 21

WHERE id = 1;

DELETE Command:
Used to remove one or more records from a table.
Syntax:

DELETE FROM table_name

WHERE condition;

Example:

DELETE FROM students

WHERE id = 1;

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

Ans-The WHERE clause specifies which rows should be affected by the UPDATE or DELETE command.

- Without a WHERE clause:
  - UPDATE changes all records in the table.
  - DELETE removes all records from the table.
- Therefore, it is essential to use the WHERE clause carefully to avoid unwanted data loss or modification.

Example (with WHERE):

DELETE FROM students WHERE id = 1;

→ Deletes only the student with ID = 1.

Example (without WHERE):

DELETE FROM students;

→ Deletes all students from the table.

7. Data Manipulation Language (DML)

LAB EXERCISES:

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

←T→	course_id	course_name	course_duration
Edit Copy Delete	1	B.Tech	4 Years
Edit Copy Delete	2	MBA	2 Years
Edit Copy Delete	3	B.Sc	3 Years

Ans-

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

←T→	course_id	course_name	course_duration
Edit Copy Delete	1	B.Tech	5 Years
Edit Copy Delete	2	MBA	2 Years
Edit Copy Delete	3	B.Sc	3 Years

Ans-

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

	course_id	course_name	course_duration
Edit Copy Delete	1	B.Tech	5 Years
Edit Copy Delete	2	MBA	2 Years

Ans-

8. Data Query Language (DQL)

Theory Questions:

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

Ans-What is the SELECT statement, and how is it used to query data?

- The SELECT statement is the main command in Data Query Language (DQL).
- It is used to retrieve data from one or more tables in a database.
- You can use it to display all columns or only specific columns, apply conditions, sort results, or even perform calculations.

Syntax:

SELECT column1, column2, ...

FROM table_name

WHERE condition

ORDER BY column_name;

Example:

SELECT name, age

FROM students

WHERE age > 18;

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

Ans-WHERE Clause:

- Used to filter rows based on a specific condition.
- Only the records that meet the condition are retrieved.

Syntax:

SELECT * FROM table_name WHERE condition;

Example:

SELECT * FROM students WHERE age > 18;

➞ Retrieves only students older than 18.

ORDER BY Clause:

- Used to sort the result set (ascending or descending order) based on one or more columns.
- By default, sorting is ascending (ASC), but you can specify descending (DESC).

Syntax:

SELECT * FROM table_name ORDER BY column_name [ASC|DESC];

Example:

SELECT * FROM students ORDER BY name ASC;

8. Data Query Language (DQL)

LAB EXERCISES:

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

Ans-

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	course_id	int(11)			No	None			Change Drop More
2	course_name	varchar(100)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	course_duration	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More

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

Ans-

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)

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

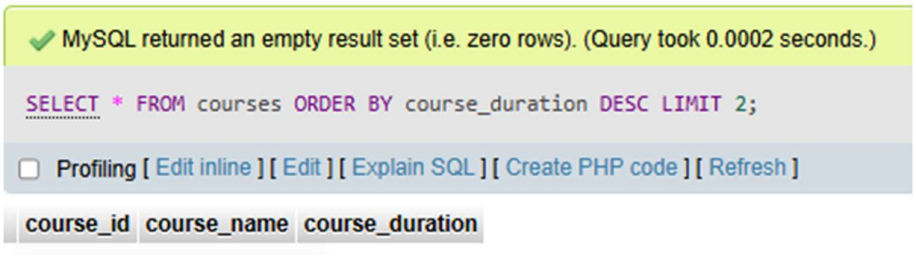
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

course_id	course_name	course_duration

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	course_id	int(11)			No	None			Change Drop More
2	course_name	varchar(100)	utf8mb4_general_ci		Yes	NULL			Change Drop More
3	course_duration	varchar(50)	utf8mb4_general_ci		Yes	NULL			Change Drop More

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

Ans-



9. Data Control Language (DCL)

Theory Questions:

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

Ans-Data Control Language (DCL) commands are used to control access to data stored in a database.
The two main DCL commands are GRANT and REVOKE.

- GRANT Command:
  Used to give (assign) specific permissions or privileges to users on database objects such as tables, views, or procedures.

Purpose: Allows users to perform certain operations (like SELECT, INSERT, UPDATE, DELETE) on database objects.

Syntax:

GRANT privilege_name

ON object_name

TO user_name;

Example:

GRANT SELECT, INSERT ON students TO user1;

REVOKE Command:
Used to **remove (take back)** permissions that were previously granted to a user.

Purpose: Restricts access to database objects when needed for security reasons.

Syntax:

REVOKE privilege_name

ON object_name

FROM user_name;

Example:

REVOKE INSERT ON students FROM user1;

2. How do you manage privileges using these commands?

Ans-Privileges (permissions) define what actions a user can perform on database objects. You can manage them using GRANT and REVOKE as follows:

Operation	Command Used	Description
Give privileges	GRANT	Allows users to access or modify data.
Take back privileges	REVOKE	Removes previously given permissions.

Example of Managing Privileges:

-- Grant privileges

GRANT SELECT, UPDATE ON employees TO user2;

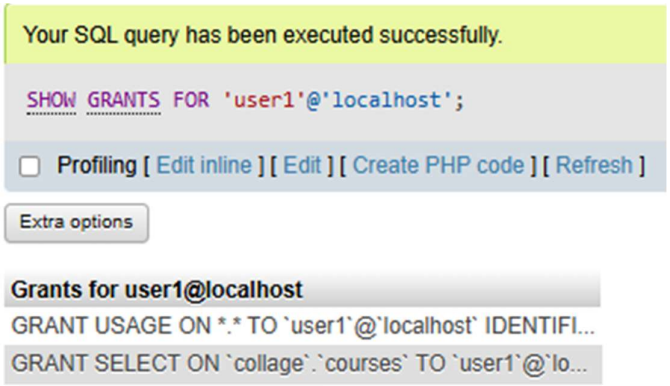
-- Revoke privileges

REVOKE UPDATE ON employees FROM user2;

9. Data Control Language (DCL)

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

Ans-



10. Transaction Control Language (TCL)

Theory Questions:

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

Ans-COMMIT Command:

- Used to save all the changes made by the current transaction permanently in the database.
- Once a COMMIT is issued, the changes cannot be undone.

Syntax:

COMMIT;

Example:

UPDATE students SET age = 21 WHERE id = 1;

COMMIT;

➞ The updated data is now permanently saved in the database.

ROLLBACK Command:

- Used to undo all the changes made in the current transaction (since the last COMMIT or SAVEPOINT).
- It restores the database to its previous consistent state.

Syntax:

ROLLBACK;

Example:

UPDATE students SET age = 21 WHERE id = 1;

ROLLBACK;

2. Explain how transactions are managed in SQL databases.

Ans-A transaction ensures that a set of database operations are executed in a safe and reliable manner following the ACID properties:

Property	Meaning	Description
A – Atomicity	All or nothing	Either all statements in a transaction execute successfully or none at all.
C – Consistency	Valid state	The database remains consistent before and after the transaction.
I – Isolation	Independence	Multiple transactions can occur without interfering with each other.
D – Durability	Permanent	Once committed, the changes are permanently stored in the database.

Transaction Management Commands:

1. BEGIN TRANSACTION / START TRANSACTION – Starts a new transaction.
2. COMMIT – Saves the changes permanently.
3. ROLLBACK – Cancels the changes made during the transaction.
4. SAVEPOINT – Sets a point within a transaction to which you can later roll back.

Example:

START TRANSACTION;

UPDATE accounts SET balance = balance - 500 WHERE acc_no = 101;

UPDATE accounts SET balance = balance + 500 WHERE acc_no = 102;

COMMIT;

ROLLBACK;

10. Transaction Control Language (TCL)

LAB EXERCISES:

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

Ans-

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

Ans-

11. SQL Joins

Theory Questions:

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

Ans-Concept of JOIN:

- A JOIN in SQL is used to combine data from two or more tables based on a related column (usually a common key, like id or student_id).
- Joins help retrieve meaningful combined information stored across multiple tables.

General Syntax:

SELECT table1.column_name, table2.column_name

FROM table1

JOIN table2

ON table1.common_column = table2.common_column;

Types of Joins in SQL:

Join Type	Description	Result Includes	Diagram Summary
INNER JOIN	Returns only matching rows from both tables.	Rows where the join condition is true in both tables.	Intersection of both tables.
LEFT JOIN (LEFT OUTER JOIN)	Returns all rows from the left table, and matching rows from the right table.	Unmatched rows from the left table show NULL for right-side columns.	All from left + matches from right.
RIGHT JOIN (RIGHT OUTER JOIN)	Returns all rows from the right table, and matching rows from the left table.	Unmatched rows from the right table show NULL for left-side columns.	All from right + matches from left.
FULL OUTER JOIN	Returns all rows from both tables, with NULLs where there is no match.	Combines results of LEFT and RIGHT JOIN.	All from both sides.

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

Ans-Joins connect tables through common columns (keys), allowing users to view related information together.

They are essential when data is normalized (spread across multiple related tables).

Example:
To display each student's name along with their marks:

SELECT students.name, marks.marks

FROM students

INNER JOIN marks

ON students.student_id = marks.student_id;

11. SQL Joins

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

Ans-

emp_id	emp_name	dept_name
101	Alice	Computer Science
102	Bob	Computer Science
103	Charlie	Mathematics

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

Ans-

dept_id	dept_name	emp_name
1	Computer Science	Alice
1	Computer Science	Bob
2	Mathematics	Charlie
3	Physics	*NULL*
4	Chemistry	*NULL*

12. SQL Group By

Theory Questions:

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

- **Ans-** The GROUP BY clause in SQL is used to group rows that have the same values in one or more columns.
- It is mainly used with aggregate functions such as:
    - COUNT() → counts rows
    - SUM() → adds values
    - AVG() → calculates average

- MAX() → finds maximum
- MIN() → finds minimum

This allows you to summarize data for each group.

Syntax:

SELECT column_name, aggregate_function(column_name)

FROM table_name

GROUP BY column_name;

Example:

Suppose we have a table named Sales:

customer	product	amount
Raj	Pen	50
Priya	Book	100
Raj	Pencil	30
Priya	Eraser	20
Amit	Book	70

To find total sales amount by each customer:

SELECT customer, SUM(amount) AS total_sales

FROM Sales

GROUP BY customer;

Result:

customer	total_sales
Raj	80
Priya	120
Amit	70

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

Ans-

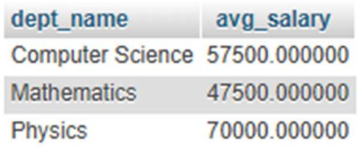
Feature	GROUP BY	ORDER BY
Purpose	Groups rows that have the same values into summary rows.	Sorts the result set in ascending or descending order.
Used With	Usually used with aggregate functions (SUM, COUNT, AVG, etc.).	Used for sorting data (does not aggregate).
Output Effect	Reduces number of rows (summarized results).	Keeps all rows, just rearranged.
Default Order	No specific order (unless ORDER BY is used).	Ascending (ASC) by default; can specify DESC.

12. SQL Group By

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

Ans-

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



Ans-

13. SQL Stored Procedure

Theory Questions:

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 group of SQL statements (such as SELECT, INSERT, UPDATE, DELETE) that are stored in the database and can be executed as a single unit.

It works like a function in programming — you create it once and can call it multiple times whenever needed.

Syntax Example:

```
CREATE PROCEDURE GetStudentDetails
AS
BEGIN
  SELECT * FROM students;
END;
```

To execute (or call) the procedure:

```
EXEC GetStudentDetails;
```

Difference between a Stored Procedure and a Standard SQL Query:

Feature	Stored Procedure	Standard SQL Query
Definition	A saved and named block of SQL code stored in the database.	A single SQL statement executed directly.
Execution	Precompiled; can be executed multiple times using a call.	Parsed and executed each time it runs.
Performance	Faster because it's precompiled and optimized by the database.	Slower for repetitive tasks since it compiles every time.
Reusability	Can be reused and called in applications or triggers.	Must be rewritten or re-executed manually each time.
Security	Provides controlled access to data (can limit permissions).	Gives direct access to tables and data.

2. Explain the advantages of using stored procedures.

Ans-Stored procedures provide many benefits for database management and application performance:

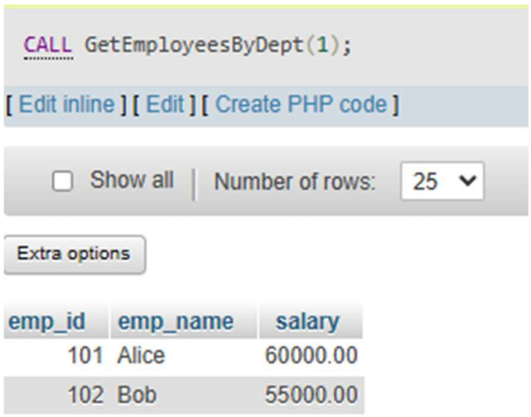
Advantage	Explanation
1. Improved Performance	Stored once and precompiled, so execution is faster than running multiple individual SQL queries.
2. Reusability	The same procedure can be reused by multiple programs or users, reducing code duplication.
3. Security	Permissions can be granted on the procedure instead of the underlying tables, protecting sensitive data.

Advantage	Explanation
4. Maintainability	Logic is stored in one place; changes to business rules can be made in the procedure without altering the application code.
5. Reduced Network Traffic	Instead of sending multiple SQL statements from the client to the server, one procedure call can perform all operations.
6. Modularity	Complex database logic can be broken into smaller, manageable stored procedures.

13. SQL Stored Procedure

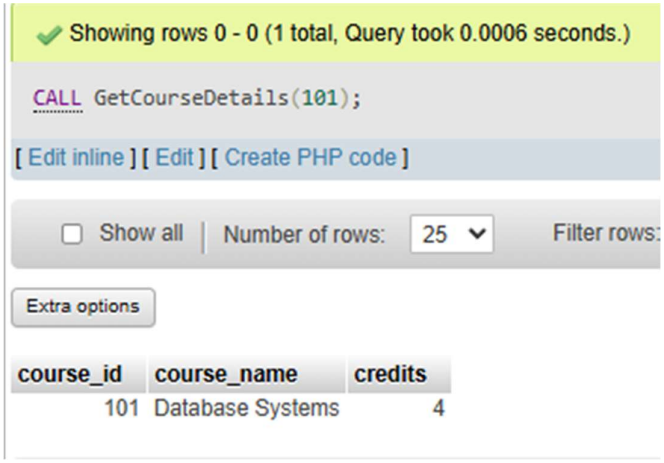
Lab 1: Write a stored procedure to retrieve all employees from the employees table based on department.

Ans-



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

Ans-



14. SQL View

Theory Questions:

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

Ans-A view in SQL is a virtual table that is created using the result of a SELECT query.
It does not store data physically; instead, it displays data stored in one or more actual tables.

In simple terms:
A view acts like a window to look at data from one or more tables in a customized way.

Syntax:

CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;

Example:

CREATE VIEW HighScorers AS

SELECT name, marks

FROM students

WHERE marks > 80;

To use the view:

SELECT * FROM HighScorers;

2. Explain the advantages of using viewsin SQL databases.

Ans-Views provide several benefits for database design, security, and maintenance:

Advantage	Explanation
1. Data Security	Views can hide sensitive columns or rows from users. You can grant access to the view instead of the entire table.
2. Simplifies Complex Queries	Complex joins or calculations can be stored as a view, making it easy to query without rewriting long SQL statements.
3. Data Abstraction	Provides a customized representation of data without changing the underlying tables.
4. Reusability and Convenience	Once created, a view can be reused like a regular table for SELECT queries.
5. Logical Data Independence	Changes in the table structure (like adding columns) don't affect users who query the view.

Advantage	Explanation
6. Consistency	Ensures users always see consistent and up-to-date data from the base tables.

Example:

If you want users to see only student names and marks (but not IDs), you can create:

CREATE VIEW Student Marks AS

SELECT name, marks FROM students;

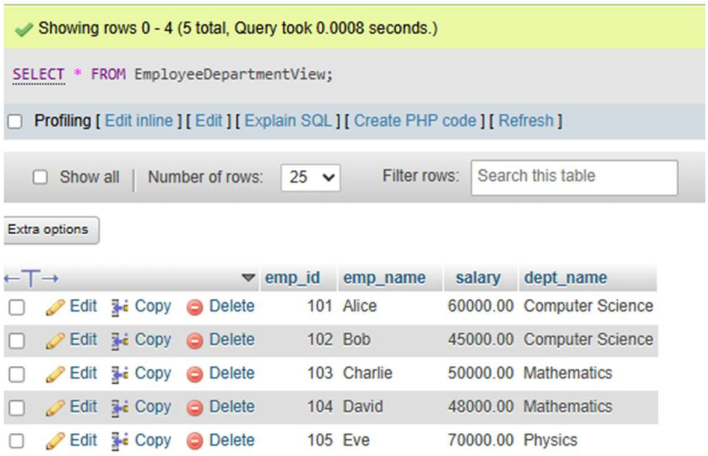
Now users can run:

SELECT * FROM Student Marks**;**

14. SQL View

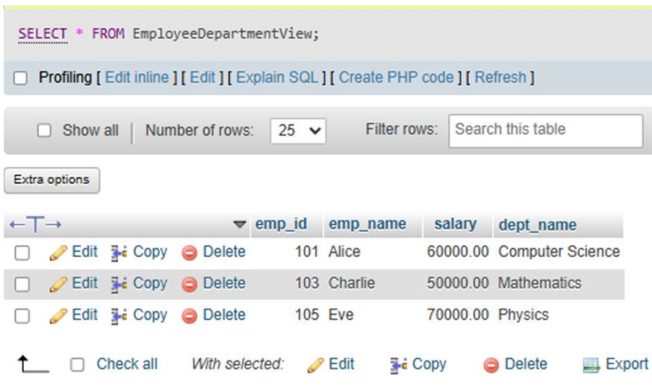
• Lab 1: Create a view to show all employees along with their department names.

Ans-



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

Ans-



15. SQL Triggers

Theory Questions:

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

Ans-A trigger in SQL is a special kind of stored procedure that is automatically executed (fired) by the database in response to specific events on a table or view — such as INSERT, UPDATE, or DELETE.

In short, a trigger is used to automate actions when certain changes occur in the database.

Syntax:

```
CREATE TRIGGER trigger_name

AFTER INSERT

ON table_name

FOR EACH ROW

BEGIN

   SQL statements to execute

END;
```

Purpose / Use of Triggers:

- To maintain data integrity automatically.
- To audit or log changes made to a table.
- To enforce business rules (e.g., prevent deletion of important records).
- To automatically update or validate data in related tables.

Types of Triggers (Based on Timing and Event):

A. Based on Timing:

1. BEFORE Trigger – Executes *before* the triggering action (INSERT, UPDATE, DELETE).
   ➞ Used to validate or modify data before saving it to the table.
2. AFTER Trigger – Executes *after* the triggering action.
   ➞ Used for logging, auditing, or updating related tables after data changes.

B. Based on Event:

1. INSERT Trigger – Fires when a new record is inserted into a table.
   ➞ Example: Automatically add an entry to an audit log when a new user is added.

2. UPDATE Trigger – Fires when a record is modified.
   ➞ Example: Track who updated a record or ensure certain fields are not changed.
3. DELETE Trigger – Fires when a record is deleted from a table.
   ➞ Example: Log deleted records or prevent deletion of critical data.

2. Explain the difference between INSERT, UPDATE, and DELETE triggers.

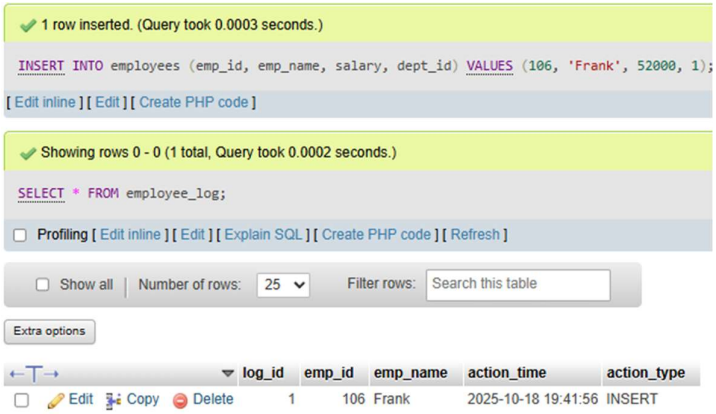
Ans-

Trigger Type	When It Fires	Used For	Example Use Case
INSERT Trigger	When a new row is inserted into a table.	To perform actions after new data is added.	Automatically add a record to an audit log or update a related table.
UPDATE Trigger	When existing data is modified.	To track or validate changes.	Keep a history of modifications or restrict updates to specific columns.
DELETE Trigger	When a row is deleted from a table.	To prevent or log deletions.	Archive deleted records or stop users from deleting important data.

15. SQL Triggers

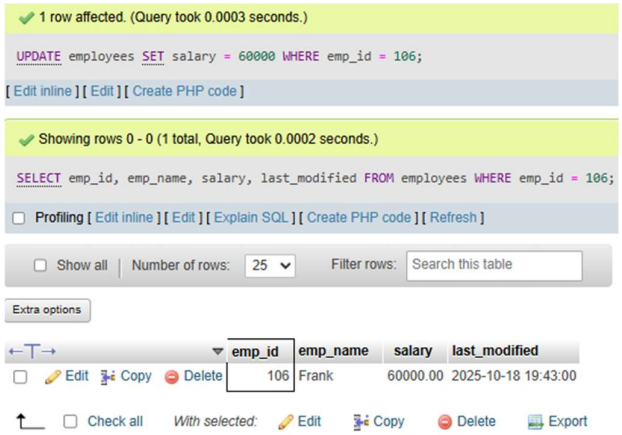
Lab 1: Create a trigger to automatically log changes to the employees table when a new employee is added.

Ans-



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

Ans-



16. Introduction to PL/SQL

Theory Questions:

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

Ans-PL/SQL (Procedural Language/Structured Query Language) is an extension of SQL developed by Oracle that adds programming features such as variables, loops, conditions, and error handling to SQL.

In short, SQL is a *query language* used to interact with the database, while PL/SQL is a *procedural language* that allows you to write programs (blocks of code) to control how SQL statements are executed.

How PL/SQL Extends SQL:

Feature	SQL	PL/SQL
Nature	Declarative (only specifies *what* to do).	Procedural (specifies *how* to do it).
Statements	Can execute one query at a time.	Can group multiple SQL statements with control structures.
Programming Logic	No loops, conditions, or variables.	Supports loops (FOR, WHILE), conditions (IF), and variables.
Error Handling	Limited error feedback.	Built-in exception handling using EXCEPTION blocks.
Reusability	SQL queries are executed individually.	PL/SQL can define procedures, functions, triggers, and packages for reuse.

2. List and explain the benefits of using PL/SQL.

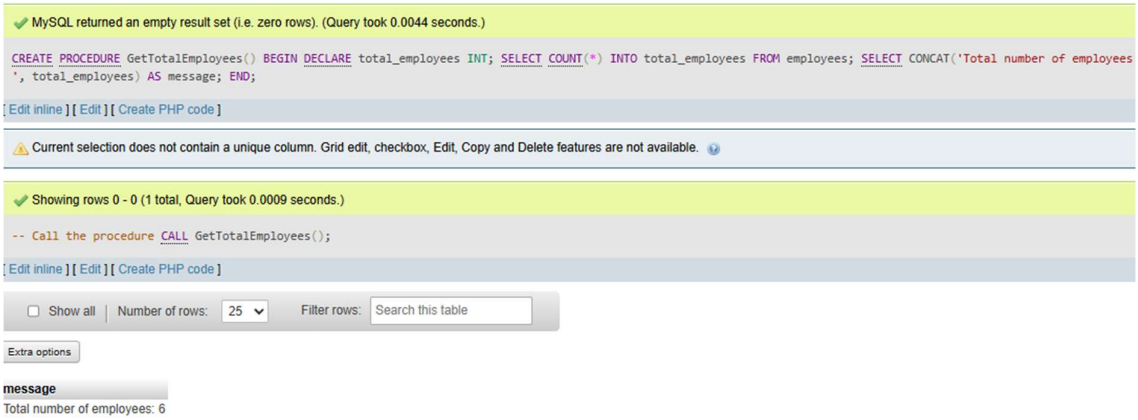
Ans-

Benefit	Explanation
1. Better Performance	Multiple SQL statements can be sent to the database at once, reducing communication overhead and improving speed.
2. Block Structure	Code is organized into logical blocks (DECLARE, BEGIN, EXCEPTION), making it easy to read, manage, and debug.
3. Supports Procedural Features	Allows the use of programming constructs like variables, loops, conditions, and functions — making SQL more powerful.
4. Error Handling (Exception Handling)	Provides built-in mechanisms to handle runtime errors gracefully using the EXCEPTION section.
5. Reusability and Modularity	You can store PL/SQL code as procedures, functions, and packages, which can be reused across applications.
6. Data Security	Sensitive business logic can be stored in the database instead of application code, reducing security risks.
7. Integration with SQL	PL/SQL fully supports SQL — you can run queries, updates, and DML operations seamlessly within PL/SQL blocks.

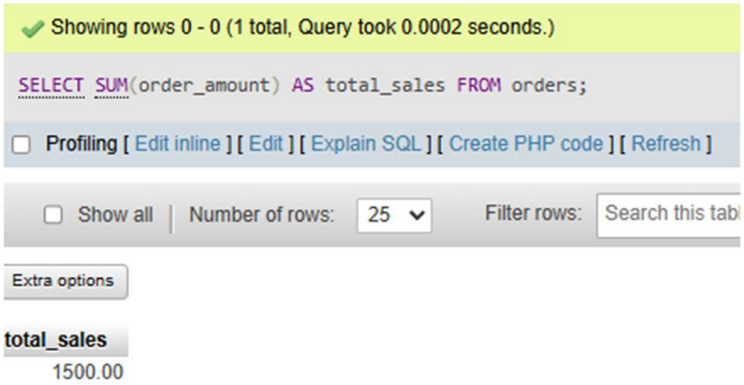
16. Introduction to PL/SQL

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

Ans-



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



Ans-

17. PL/SQL Control Structures

Theory Questions:

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

Ans-Control structures in PL/SQL are programming constructs that allow you to control the flow of execution in a PL/SQL block based on conditions or repetition.

They include:

- Conditional statements → IF-THEN, IF-THEN-ELSE, CASE
- Looping statements → LOOP, WHILE LOOP, FOR LOOP

These help perform decision-making and repetitive tasks within a program.

A. IF-THEN Statement:

- Executes a block of code only if a condition is true.
- Can be extended with ELSE and ELSIF for multiple conditions.

Syntax:

IF condition THEN

  -- statements to execute if condition is true

ELSIF another_condition THEN

  -- statements to execute if another_condition is true

ELSE

  -- statements to execute if all conditions are false

END IF;

Example:

```
DECLARE
  marks NUMBER := 75;
BEGIN
  IF marks >= 80 THEN
    DBMS_OUTPUT.PUT_LINE('Excellent');
  ELSIF marks >= 60 THEN
    DBMS_OUTPUT.PUT_LINE('Good');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Needs Improvement');
  END IF;
END;
```

Output: Good

B. LOOP Statement:

- Used to repeat a block of statements multiple times.
- Types:
    1. Simple LOOP – repeats indefinitely until EXIT is encountered.
    2. WHILE LOOP – repeats as long as a condition is true.
    3. FOR LOOP – repeats for a fixed number of times.

Simple LOOP Example:

```
DECLARE
  counter NUMBER := 1;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE('Counter: ' || counter);
    counter := counter + 1;
    EXIT WHEN counter > 5;
  END LOOP;
END;
```

Output: Counter: 1 ... Counter: 5

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

Ans-Decision Making: Using IF-THEN-ELSE or CASE, you can perform different actions based on conditions.

- Repetition: Using loops, you can process multiple rows or perform repetitive tasks without writing the same SQL multiple times.
- Modularity: Control structures allow grouping of SQL operations with logic and flow control, making programs easier to maintain.
- Dynamic Actions: You can check conditions, update tables, call procedures, and handle exceptions based on runtime data.

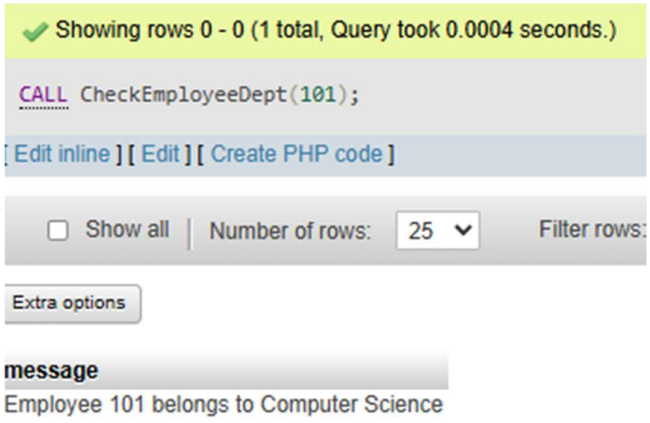
Example: Updating student grades automatically based on marks:

```
DECLARE
  student_id NUMBER := 1;
  marks NUMBER;
BEGIN
  SELECT marks INTO marks FROM students WHERE id = student_id;
  IF marks >= 90 THEN
    UPDATE students SET grade = 'A' WHERE id = student_id;
  ELSIF marks >= 75 THEN
    UPDATE students SET grade = 'B' WHERE id = student_id;
  ELSE
    UPDATE students SET grade = 'C' WHERE id = student_id;
  END IF;
END;
```

This uses IF-THEN-ELSE to assign grades automatically.

17. PL/SQL Control Structures

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



Ans-

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

Ans-

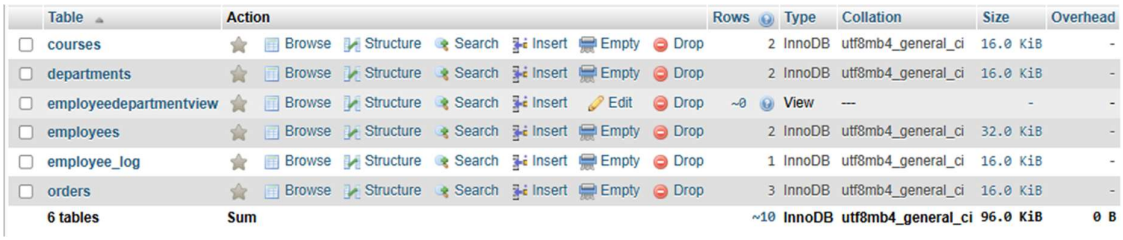


Table	Action	Rows	Type	Collation	Size	Overhead
courses	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
departments	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
employeedepartmentview	Browse Structure Search Insert Edit Drop	~0	View	---	-	-
employees	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
employee_log	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	16.0 KiB	-
orders	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_general_ci	16.0 KiB	-
6 tables	**Sum**	**~10**	**InnoDB**	**utf8mb4_general_ci**	**96.0 KiB**	**0 B**

18. SQL Cursors

Theory Questions:

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

Ans-A cursor in PL/SQL is a pointer that allows you to retrieve and manipulate rows returned by a SQL query one at a time.

- Useful when a query returns multiple rows and you want to process each row individually.

Types of Cursors:

Type	Definition	Key Points
Implicit Cursor	Automatically created by PL/SQL when a SELECT INTO, INSERT, UPDATE, or DELETE statement affects only one row.	- No explicit declaration needed. - Cursor attributes: %FOUND, %NOTFOUND, %ROWCOUNT, %ISOPEN.

Type	Definition	Key Points
Explicit Cursor	Must be declared, opened, fetched, and closed manually by the programmer. Used for queries that return multiple rows.	- Provides more control over row-by-row processing. - Steps: DECLARE → OPEN → FETCH → CLOSE.

Difference Between Implicit and Explicit Cursors:

Feature	Implicit Cursor	Explicit Cursor
Declaration	Not required	Must be declared explicitly
Row Handling	Single row	Multiple rows
Control	Limited	Full control (OPEN, FETCH, CLOSE)
Cursor Attributes	%FOUND, %NOTFOUND, %ROWCOUNT, %ISOPEN	Same attributes available but used explicitly

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

Ans-Query returns multiple rows that need to be processed one by one.

1. You want fine-grained control over the retrieval of rows.
2. You need to loop through the result set for operations like logging, calculations, or updates per row.
3. You want to use cursor attributes to check row status during iteration.

Example Use Case:

- Fetching all students with marks above 70 and printing their details individually.
- Implicit cursors cannot handle multiple rows; hence an explicit cursor is required
- **18. SQL Cursors**
- **• Lab 1: Write a PL/SQL block using an explicit cursor to retrieve and display employee details.**
- **Ans-**

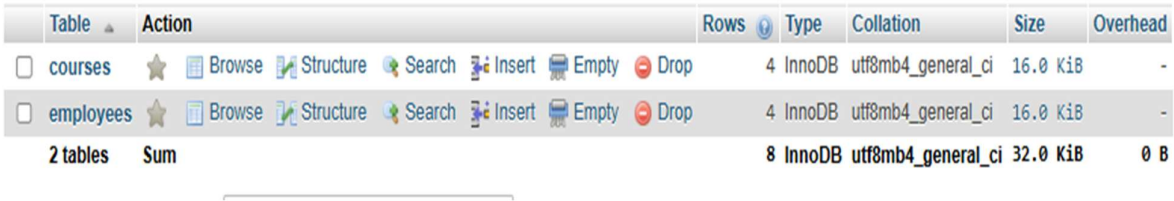


	Table	Action	Rows	Type	Collation	Size	Overhead
☐	courses	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
☐	employees	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
	2 tables	Sum	8	InnoDB	utf8mb4_general_ci	32.0 KiB	0 B

19. Rollback and Commit Savepoint

Theory Questions:

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

Ans-A SAVEPOINT is a marker or point within a transaction that allows you to roll back part of the transaction without affecting the entire transaction.

- Think of it as a checkpoint in your transaction.

Key Points:

- SAVEPOINT does not commit changes; it just marks a point for potential rollback.
- You can create multiple savepoints in a single transaction.
- Syntax:

SAVEPOINT savepoint_name;

ROLLBACK with SAVEPOINT:

- ROLLBACK can undo changes back to a specific savepoint without affecting earlier committed parts or the rest of the transaction.

Syntax:

ROLLBACK TO savepoint_name;

Example:

START TRANSACTION;

INSERT INTO students VALUES (1, 'Raj', 20);

SAVEPOINT sp1;

INSERT INTO students VALUES (2, 'Priya', 22);

ROLLBACK TO sp1;

Only the first insert remains

COMMIT;

After rollback, only Raj is inserted; Priya is removed.

COMMIT with SAVEPOINT:

- COMMIT finalizes all changes in the transaction.

- Once committed, all savepoints are removed and cannot be rolled back.

Example:

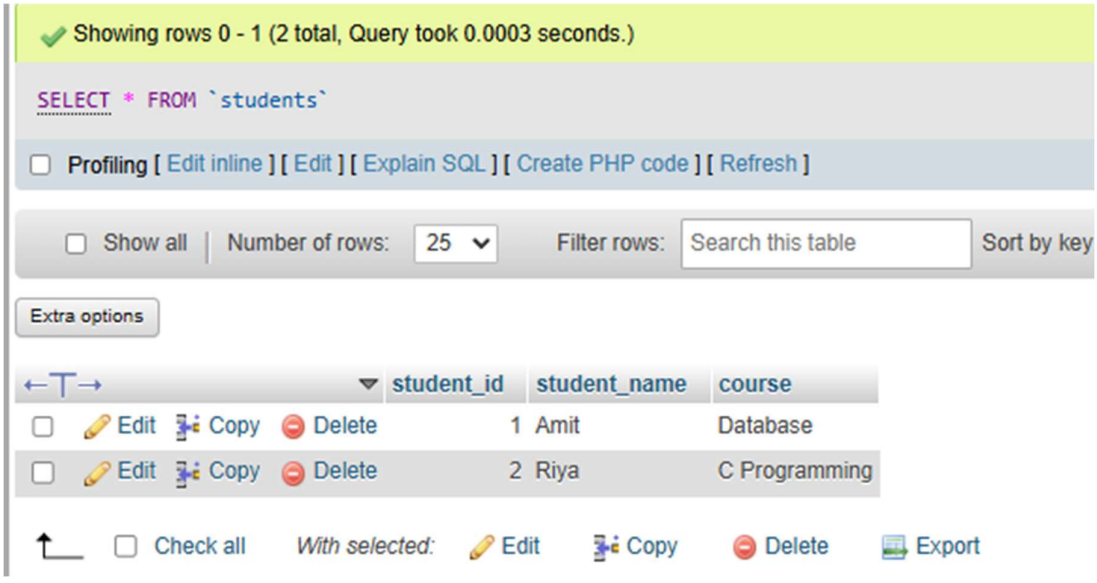
COMMIT;

All changes up to now are permanent; savepoints are gone.

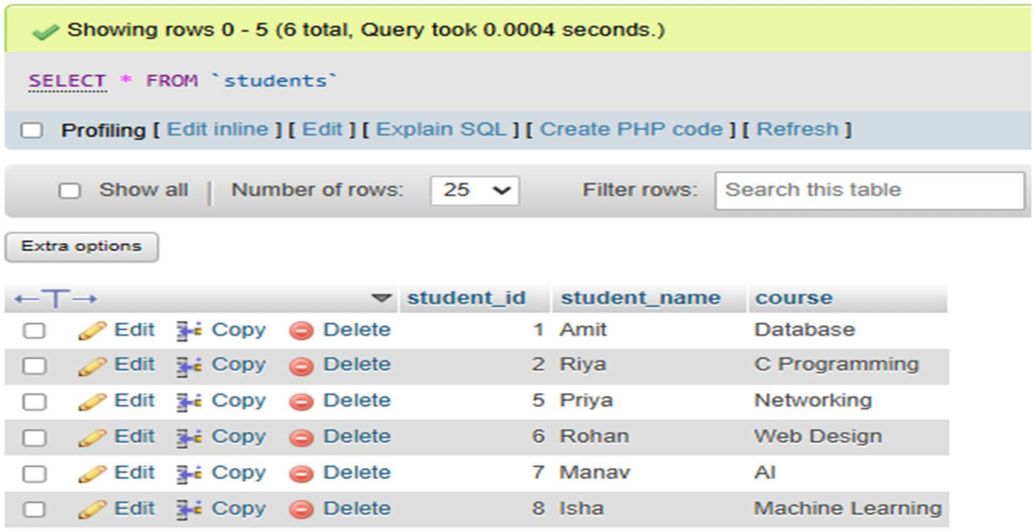
2. When is it useful to use savepoints in a database transaction?

Ams-Savepoints are useful when:

1. Partial Rollbacks are Needed:
    - You want to undo only some operations instead of the entire transaction.
2. Complex Transactions:
    - In transactions with multiple steps, savepoints allow you to recover gracefully from errors at intermediate stages.
3. Error Handling:
    - If an error occurs after a savepoint, you can rollback only to the last known good state, not the beginning.
4. Testing / Debugging Transactions:
    - Developers can mark multiple savepoints to check intermediate results before committing.
5. **19. Rollback and Commit Savepoint**
6. **Lab 1: Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.**
7. **Ans-**



8. **• Lab 2: Commit part of a transaction after using a savepoint and then rollback the remaining changes.**
9. **Ans-**



Showing rows 0 - 5 (6 total, Query took 0.0004 seconds.)

```
SELECT * FROM `students`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

←T→			student_id	student_name	course
Edit	Copy	Delete	1	Amit	Database
Edit	Copy	Delete	2	Riya	C Programming
Edit	Copy	Delete	5	Priya	Networking
Edit	Copy	Delete	6	Rohan	Web Design
Edit	Copy	Delete	7	Manav	AI
Edit	Copy	Delete	8	Isha	Machine Learning