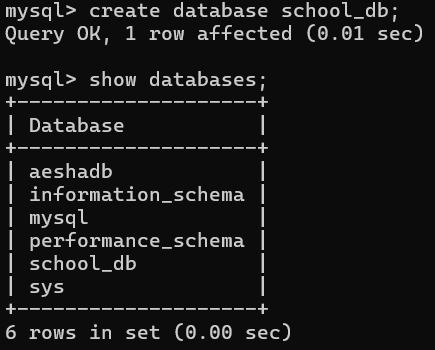
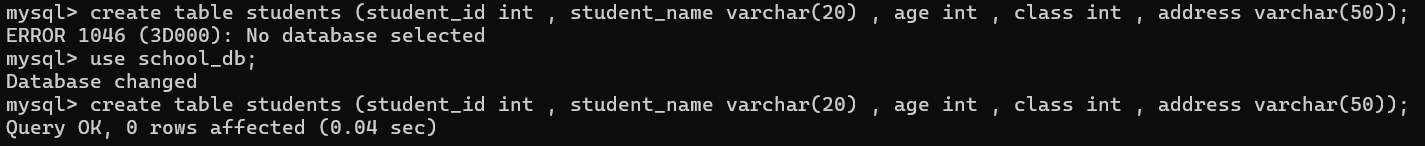
**Module 4 – Introduction to DBMS**

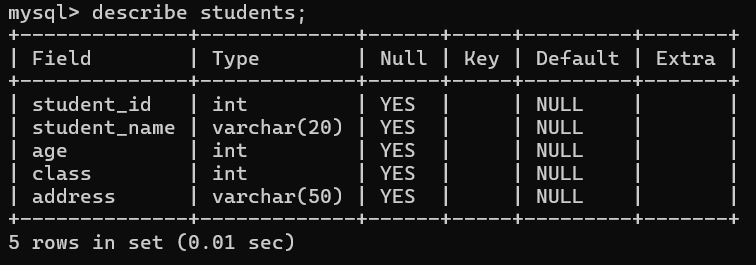
1. **Introduction to SQL**

**Lab Exercises :-**

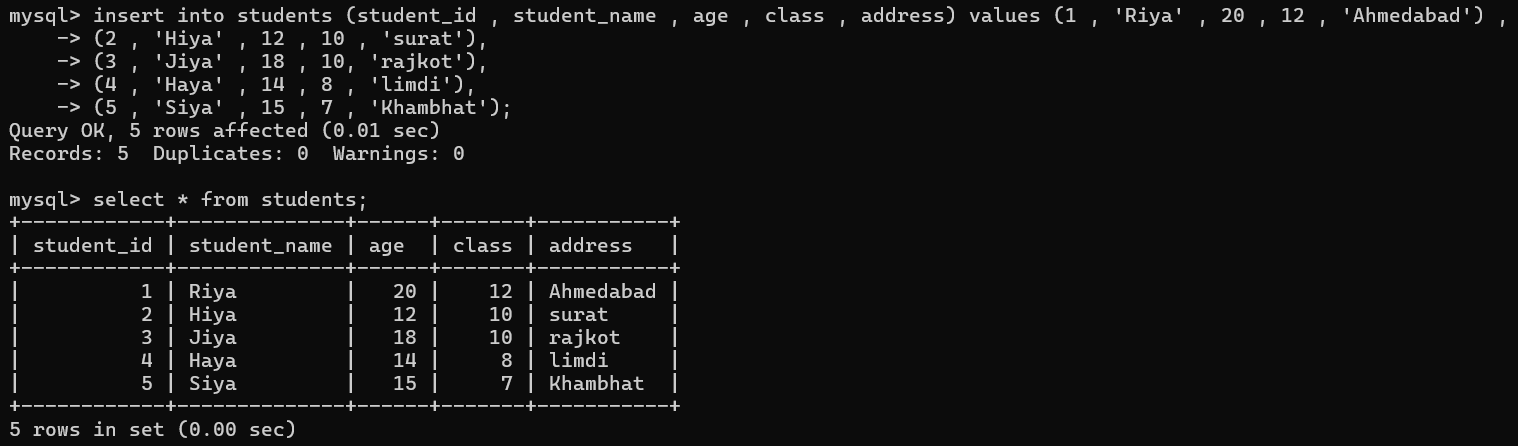
* Create a new database named school\_db and a table called students with the following columns: student\_id , student\_name, age, class, and address.

****

****

****

* Insert five records into the students table and retrieve all records using the SELECT statement.

****

**Theory Exercises :-**

* ***What is SQL, and why is it essential in database management?***
* SQL is Structural Query Language. It is the **language used to talk to databases** , with SQL you can **store, retrieve, update, and delete** data in a database.
* It helps you easily **insert, update, delete, and search** data.
* Data Security → You can give permissions (who can view or change data).
* Efficiency → Quickly works with large amounts of data.
* ***Explain the difference between DBMS and RDBMS.***

**DBMS** – Database Management System.

* It is a software that manages only database and enables users to create , manage , and interact with database.
* It includes tables which has rows (data) & columns (fields).

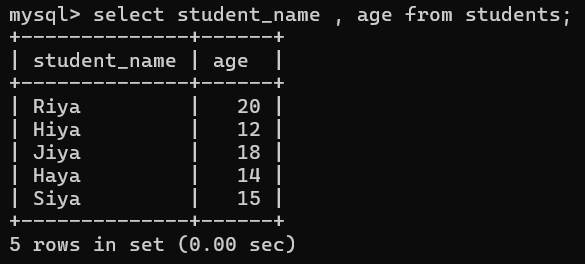
**RDBMS** – Relational Database Management System.

* It is a type of database that organizes data into tables.
* These tables are related to each other through common attributes.
* Eg:- MYSQL , Oracle , MongoDB
* ***Describe the role of SQL in managing relational databases.***
* **Data Definition (DDL):** SQL allows for the creation, modification, and deletion of database objects, including tables, views, and stored procedures.
* **Data Manipulation (DML):** SQL facilitates the manipulation of data within the database. This includes inserting new records, updating existing data, and deleting records.
* **Data Querying (DQL):** SQL enables users to retrieve specific data from one or more tables based on various criteria. This involves using clauses like SELECT, FROM, WHERE, JOIN, and GROUP BY.
* **Data Control (DCL):** SQL provides commands for managing database security and access permissions. This involves granting or revoking privileges to users or roles.
* **Transaction Control (TCL):** SQL supports transaction management . This involves commands like COMMIT to save changes permanently and ROLLBACK to undo changes in case of errors.

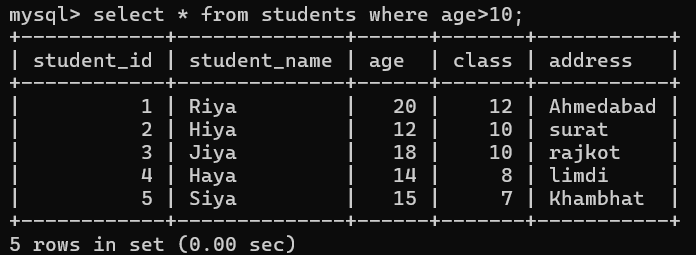
1. **SQL Syntax**

**Lab Exercises :-**

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



* Write SQL queries to retrieve all students whose age is greater than 10.



**Theory Exercises :-**

* ***What are the basic components of SQL syntax?***

The primary components include tables, queries, clauses, the SELECT statement, the INSERT command, data types, and expressions.

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

SELECT column1, column2, ...

FROM table\_name

WHERE condition

GROUP BY column

HAVING condition

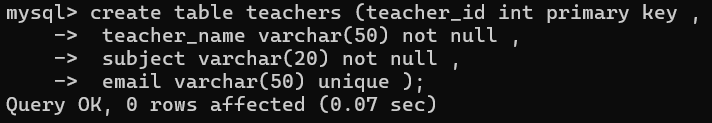
ORDER BY column ASC|DESC;

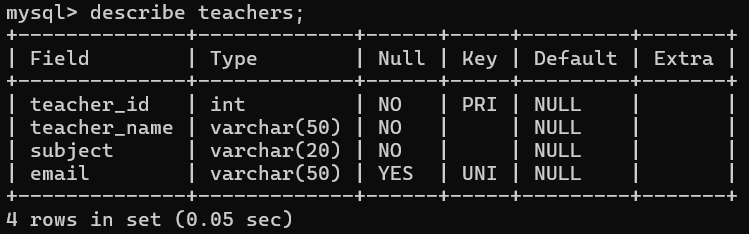
* ***Explain the role of clauses in SQL statements.***
* **Filtering Data:** Clauses like WHERE and HAVING are used to filter rows based on specified conditions. WHERE filters individual rows before grouping, while HAVING filters groups of rows after aggregation, often based on aggregate function results.
* **Grouping Data:** The GROUP BY clause groups rows that have the same values in specified columns into summary rows, allowing for aggregate calculations on these groups.
* **Ordering Data:** The ORDER BY clause sorts the result set in ascending or descending order based on one or more columns.
* **Limiting Results:** Clauses like LIMIT (in MySQL/PostgreSQL) restrict the number of rows returned by a query.
* **Joining Tables:** Clauses such as JOIN (e.g., INNER JOIN, LEFT JOIN) combine rows from two or more tables based on a related column between them.

1. **SQL Constraints**

**Lab Exercises :-**

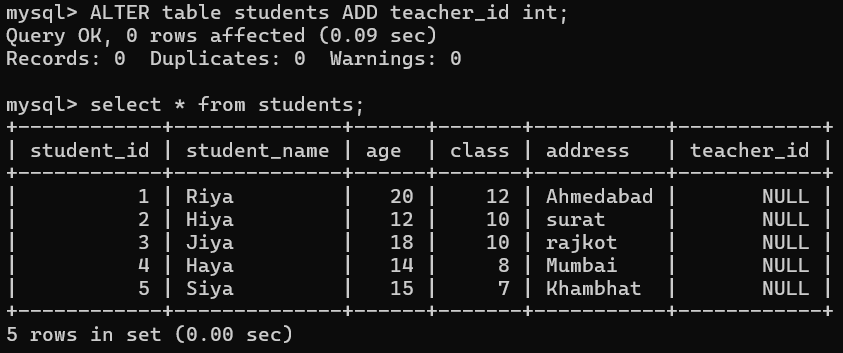
* Create a table teachers with the following columns: teacher\_id (Primary Key), teacher\_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).





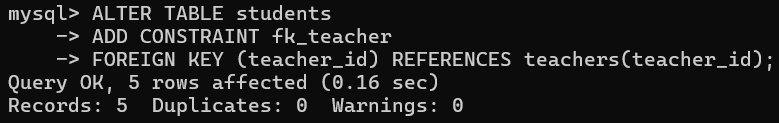
* Implement a FOREIGN KEY constraint to relate the teacher\_id from the teachers table with the students table.

ALTER command used to ADD teacher\_id field in students table.



This alter is used to add teachers\_id field as FOREIGN KEY in students table by references teachers

Table’s teacher\_id column.



**Theory Exercises :-**

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

SQL Constraints are used to specify rules for the data in a table.

1. **UNIQUE :-** By unique constraint we cannot add duplicate values into tables. It ensures that all values in a column are different.
2. **NOT NULL :-** It ensures that column cannot contain NULL values. This means a value must always be provided for that column.
3. **PRIMARY KEY :-** A combination of a NOT NULL and UNIQUE. Primary key identifies uniquely each record in a table.
4. **DEFAULT :-** It sets a default value for a column if no value is provided.
5. **CHECK :-** It ensures that the values in a column satisfies a specific condition.
6. **FOREIGN KEY :-** It is used to match two tables relationship and from that it will fetch data . A foreign key is a field (or collection of fields ) in one table , that refers to primary key in another table.

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

**Purpose:** A **PRIMARY KEY** uniquely identifies each record (row) within a single table. It ensures that no two rows in the table have the same value in the primary key column(s).

**Characteristics:**

* Must contain unique values.
* Cannot contain NULL values (it is inherently NOT NULL).
* A table can have only one PRIMARY KEY.
* Often, a clustered index is automatically created on the primary key, which can improve data retrieval performance.

**Purpose:** A **FOREIGN KEY** establishes a link or relationship between two tables. It refers to the PRIMARY KEY (or a UNIQUE key) in another table, known as the "parent" or "referenced" table.

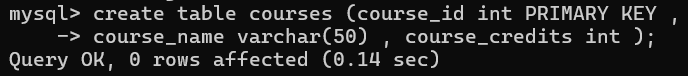
**Characteristics:**

* Can contain duplicate values.
* Can contain NULL values (unless explicitly defined as NOT NULL).
* A table can have multiple FOREIGN KEYs, referencing different parent tables.
* Does not automatically create an index, though creating an index on a foreign key is often beneficial for performance, especially during join operations.
* ***What is the role of NOT NULL and UNIQUE constraints?***
* **NOT NULL** constraints prevent null values from being entered into a column.
* **Unique** constraints ensure that the values in a set of columns are unique and not null for all rows in the table.
* The columns specified in a unique constraint must be defined as NOT NULL.

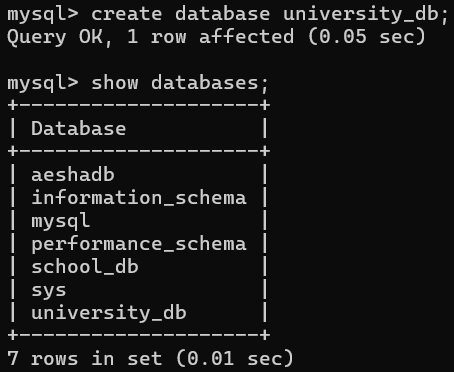
1. **Main SQL Commands and Sub-commands (DDL)**

**Lab Exercises :-**

* Create a table courses with columns: course\_id, course\_name, and course\_credits. Set the course\_id as the primary key.



* Use the CREATE command to create a database university\_db.



**Theory Exercises :-**

* ***Define the SQL Data Definition Language (DDL).***
* **DDL** is the part of SQL used to **define and manage the structure of the database** — like creating, changing, or deleting tables and databases.
* DDL commands deal with the **schema (design)** of the database, not the actual data. It’s about **how data is stored**, not about the actual data values.
* Common DDL Commands :- **CREATE** → To make new databases, tables, or objects.

**ALTER** → To change an existing table (add/remove/modify columns).

**DROP** → To permanently delete a table or database.

**TRUNCATE** → To remove all rows from a table but keep the structure.

* ***Explain the CREATE command and its syntax.***
* The **CREATE** command is a DDL (Data Definition Language) command. It is used to **create a new database** object such as a database, table, view, or index.
* Most commonly, we use it to **create tables** to store data.
* **Syntax :-** CREATE TABLE table\_name (

column1 datatype constraint,

column2 datatype constraint,

column3 datatype constraint,

... );

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

**Purpose of Data Types in Table Creation:** When you create a table, you must define **what kind of data** each column can hold.

* **Data types** ensure the **right kind of values** are stored in each column.
* Prevents wrong data (e.g., stopping someone from putting “ABC” in a marks column).
* Saves storage space by using the right type.
* Improves query performance.

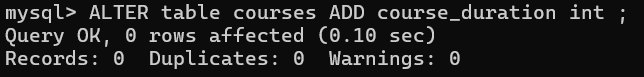
**Purpose of Constraints in Table Creation:** Constraints are **rules applied** to columns to **keep the data accurate** and **reliable**.

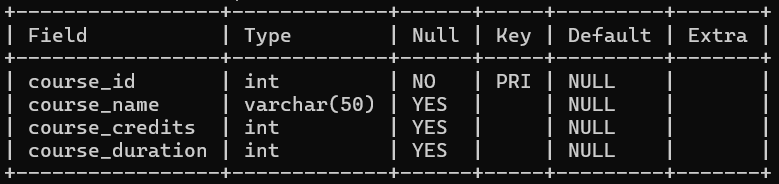
* Maintains data integrity (correctness).
* Prevents invalid or duplicate data.
* Enforces relationships between tables.

1. **ALTER Command**

**Lab Exercises :-**

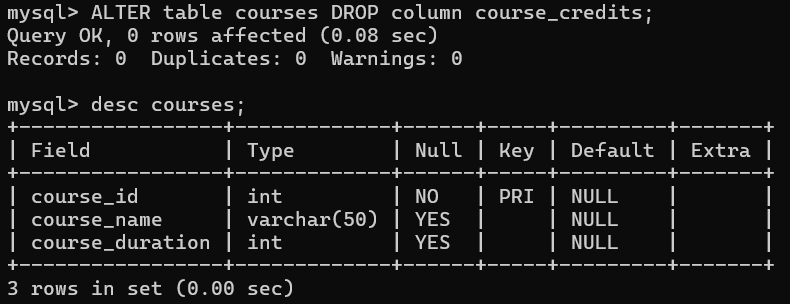
* Modify the courses table by adding a column course\_duration using the ALTER command.





* Drop the course\_credits column from the courses table.

Used ALTER command as want to drop a particular column so alter (modify) used.



**Theory Exercises :-**

* ***What is the use of the ALTER command in SQL?***
* The **ALTER** command is a **DDL (Data Definition Language)** command.
* It is used to **change the structure of an existing table** without deleting it.
* With ALTER, you can **add new columns, modify existing columns, or drop (remove) columns**.
* ***How can you add, modify, and drop columns from a table using ALTER?***
  1. **ADD Column** :- ALTER TABLE Students ADD age INT;

Adds a new column age to the Students table.

* 1. **MODIFY Column** :- ALTER TABLE Students MODIFY name VARCHAR(100);

Changes the size of the name column to 100 characters.

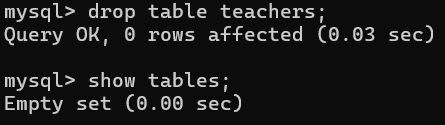
* 1. **DROP Column** :- ALTER TABLE Students DROP COLUMN age;

Removes the age column from the table.

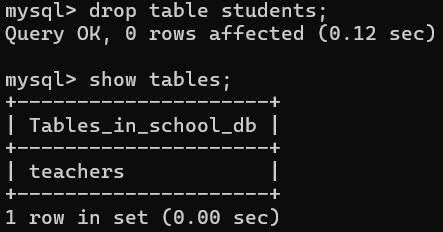
1. **DROP Command**

**Lab Exercises :-**

* Drop the teachers table from the school\_db database.



* Drop the students table from the school\_db database and verify that the table has been removed.



**Theory Exercises :-**

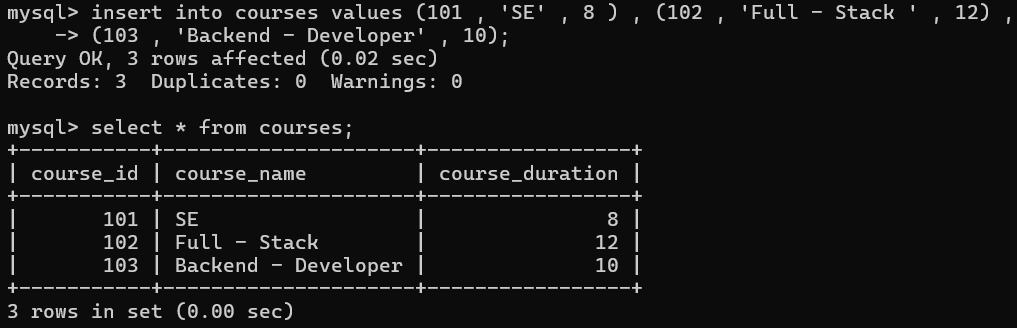
* ***What is the function of the DROP command in SQL?***
* The function of DROP command is to **removed table permanently** from database.
* By using drop command if want to drop a table having **FK constraint** then first have to **drop** its **child table** and then **parent table** will be dropped.
* **Syntax:-** DROP table table\_name;
* For database :- DROP database database\_name;
* ***What are the implications of dropping a table from a database?***

When you remove a table, you also delete the data stored in it, the indexes or constraints on the columns (including all the referential constraints placed on its columns), and any access privileges granted on the table.

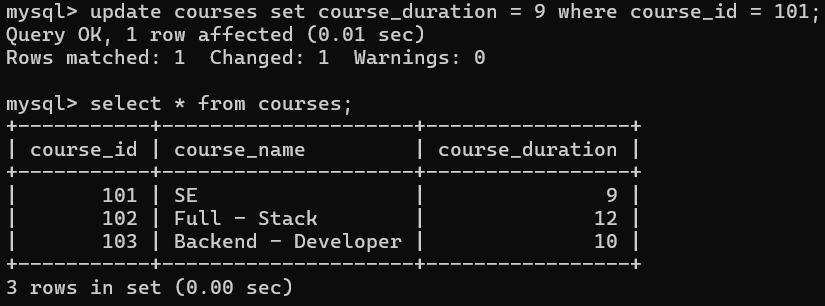
1. **Data Manipulation Language (DML)**

**Lab Exercises :-**

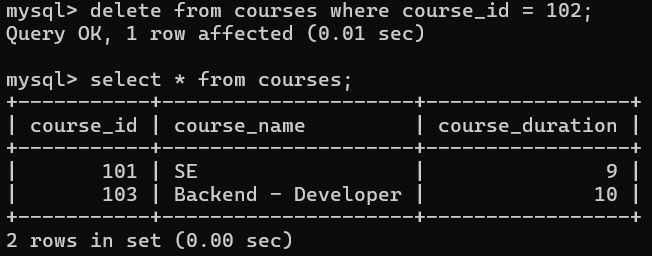
* Insert three records into the courses table using the INSERT command.



* Update the course duration of a specific course using the UPDATE command.



* Delete a course with a specific course\_id from the courses table using the DELETE command.



**Theory Exercises :-**

* ***Define the INSERT, UPDATE, and DELETE commands in SQL.***
* **INSERT :-** This command is used to **add new rows** (records) into an existing table.

Syntax:- insert into table\_name values ( values to add);

* **UPDATE :-** This command is used to **modify existing data** in one or more rows of a table.

Syntax :- update table\_name

set column1 = new\_value1, column2 = new\_value2, ...

where condition;

* **DELETE :-** This command is used to **remove one or more rows** from a table.

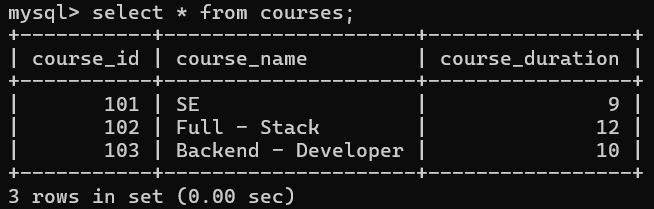
Syntax :- delete from table\_name where condition;

* ***What is the importance of the WHERE clause in UPDATE and DELETE operations?***
* The **WHERE clause decides which rows will be changed or removed.**
* If you **don’t use WHERE**, the command will affect **all rows in the table**.
* For ex:- update students set marks = 100; // this will set marks=100 for all students.
* For ex:- update students set marks = 100 where student\_id = 5; // this will set marks = 100 for only student with id = 5.
* For ex:- delete from students; // this will delete all students from the table.
* For ex:- delete from students where marks < 40; // this will delete data for marks < 40 having students only.

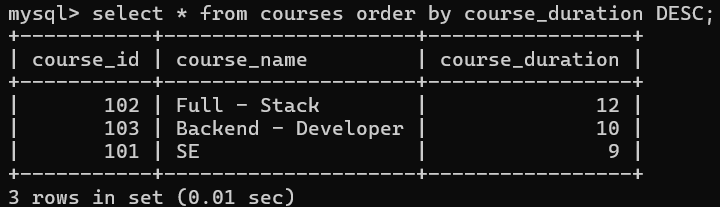
1. **Data Query Language (DQL)**

**Lab Exercises :-**

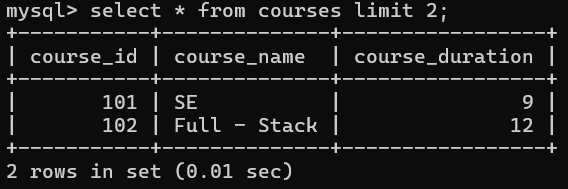
* Retrieve all courses from the courses table using the SELECT statement.



* Sort the courses based on course\_duration in descending order using ORDER BY.



* Limit the results of the SELECT query to show only the top two courses using LIMIT.



**Theory Exercises :-**

* ***What is the SELECT statement, and how is it used to query data?***
* **SELECT statement** is used to **query (fetch/retrieve) data** from one or more tables in a database.
* It does **not change data** — it only shows the data you ask for.
* ***SELECT \* FROM Students;***

Shows all rows and columns from the Students table.

* ***SELECT name, marks FROM Students;***

Shows only the name and marks columns.

* ***SELECT name, marks FROM Students ORDER BY marks DESC;***

Shows students sorted by marks (highest first).

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

**WHERE Clause**

* The **WHERE** clause is used to **filter rows** in a table.
* It decides **which records** should be included in the result.
* SELECT name, marks FROM Students WHERE marks > 50;

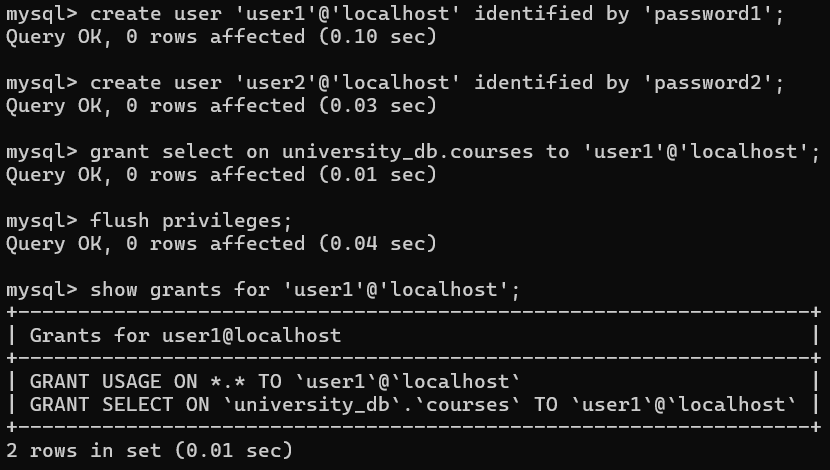
**ORDER BY Clause**

* The **ORDER BY** clause is used to **sort the result** (ascending or descending).
* Default is ascending (ASC), descending is written as DESC.
* SELECT name, marks FROM Students ORDER BY marks DESC;

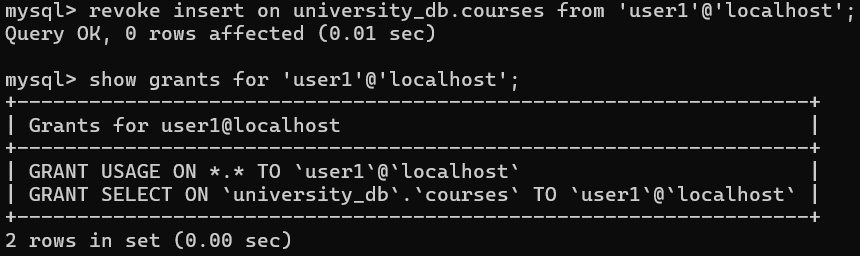
1. **Data Control Language (DCL)**

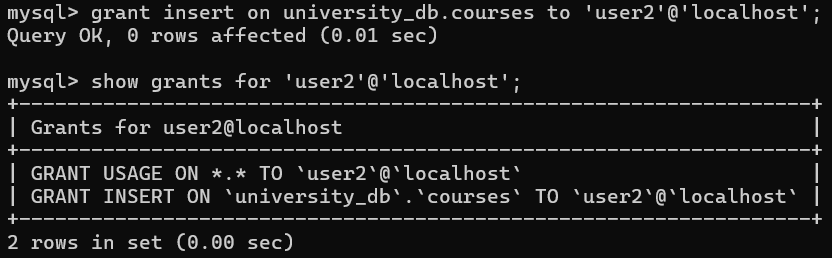
**Lab Exercises :-**

* Create two new users user1 and user2 and grant user1 permission to SELECT from the courses table. (flush privileges used to apply the changes)



* Revoke the INSERT permission from user1 and give it to user2.





**Theory Exercises :-**

* ***What is the purpose of GRANT and REVOKE in SQL?***
* GRANT :- The **GRANT** command is used to **give permissions** (privileges) to users.
* With it, you can allow a user to **access or perform actions** on a database or table.
* REVOKE :- The **REVOKE** command is used to **take back permissions** from a user.
* It removes previously granted access.
* ***How do you manage privileges using these commands?***

**Granting privileges (giving access):**

Ex :- GRANT SELECT, INSERT ON school.courses TO 'user1'@'localhost';

* This means **user1** can view (SELECT) and add (INSERT) data in the courses table.

**Revoking privileges (removing access):**

Ex :- REVOKE INSERT ON school.courses FROM 'user1'@'localhost';

* Now **user1** cannot insert data anymore but can still select.

**Checking privileges:**

Ex :- SHOW GRANTS FOR 'user1'@'localhost';

**Applying changes:**

Ex :- FLUSH PRIVILEGES;

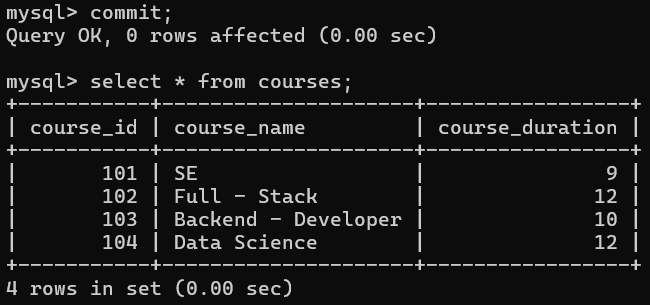
* This refreshes the system to make sure the new privileges are active.

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

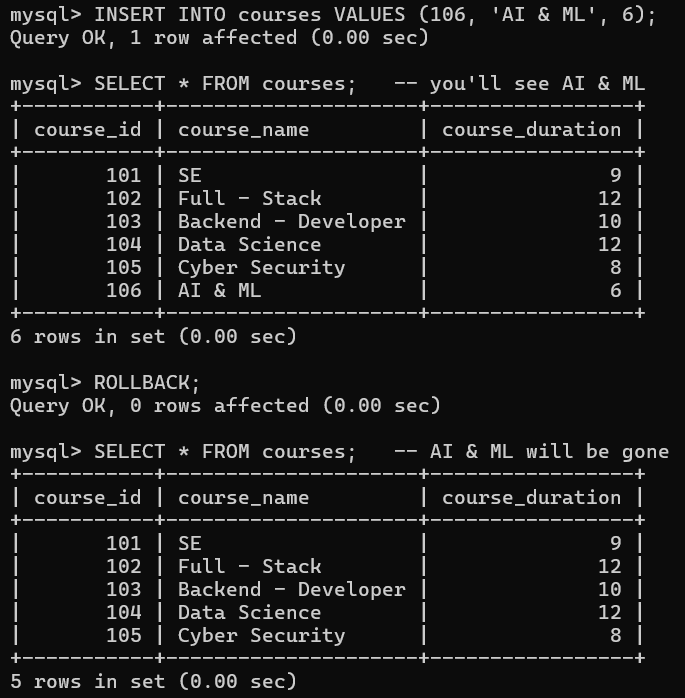
**Lab Exercises :-**

* Insert a few rows into the courses table and use COMMIT to save the changes.

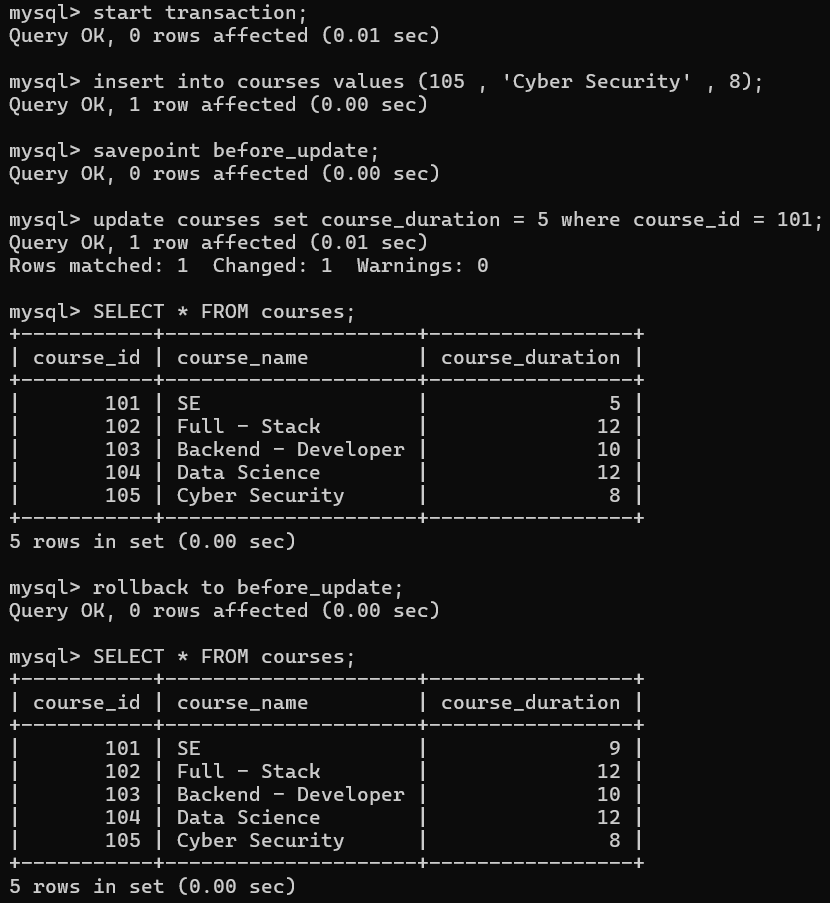




* Insert additional rows, then use ROLLBACK to undo the last insert operation.



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



**Theory Exercises :-**

* ***What is the purpose of the COMMIT and ROLLBACK commands in SQL?***
* **COMMIT :-** Saves all the changes made by the current transaction **permanently** into the database. After a COMMIT, you cannot undo the changes with ROLLBACK.
* Ex :- INSERT INTO courses VALUES (106, 'AI & ML', 11);
* The new row is permanently stored.
* **ROLLBACK :-** Undoes all the changes made in the current transaction **since the last COMMIT or SAVEPOINT**.
* Ex :- INSERT INTO courses VALUES (107, 'DevOps', 10);

ROLLBACK;

* The inserted row is discarded (not saved).
* ***Explain how transactions are managed in SQL databases.***
* A **transaction** is a **sequence of one or more SQL operations** (such as INSERT, UPDATE, DELETE) that are executed as a **single unit of work**. Either **all operations succeed** (commit) or **all fail** (rollback).
* SQL provides commands to **manage transactions**:

**START TRANSACTION / BEGIN** → Start a new transaction.

**COMMIT** → Save all changes permanently.

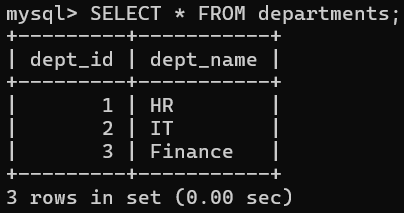
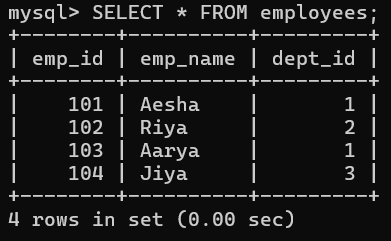
**ROLLBACK** → Undo changes made in the current transaction.

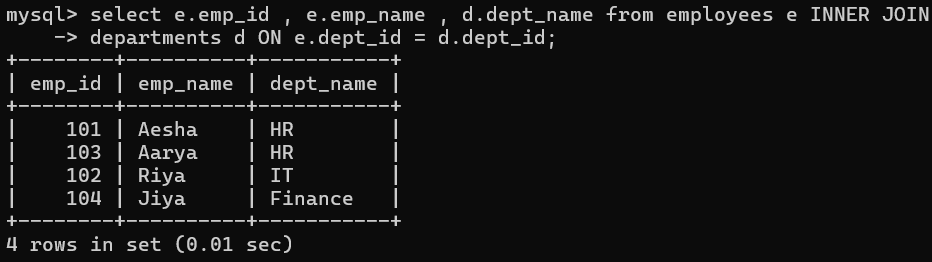
**SAVEPOINT** → Create a checkpoint inside a transaction to rollback to a specific point.

1. **SQL Joins**

**Lab Exercises :-**

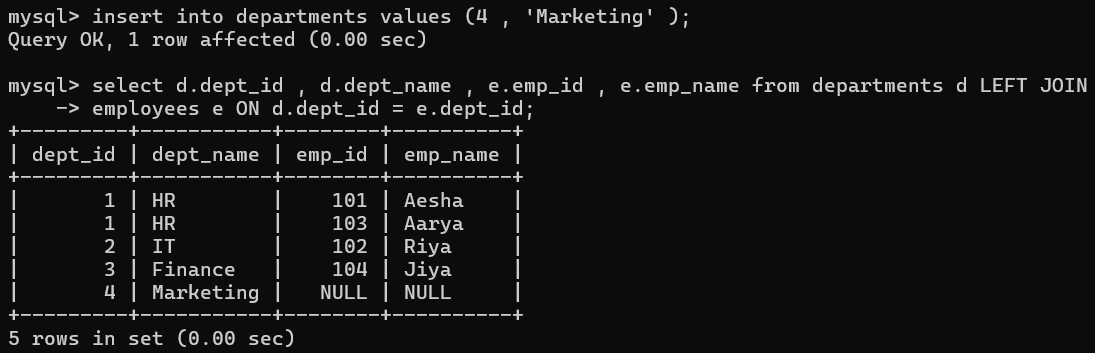
* Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.



* Use a LEFT JOIN to show all departments, even those without employees.

So here I have inserted a new record in departments table and not related dept\_id is there in employees table so on left join it will give null values.



**Theory Exercises :-**

* ***Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?***
* **JOIN :-** Join clause is used to combine rows from two or more tables based on same data. (PRIMARY KEY and FOREIGN KEY relationship )
* **INNER JOIN :-** Returns records that have **matching values** in **both the tables** .
* **LEFT JOIN :-** Returns all records from **left table** and **common records** from **both** the tables.
* **RIGHT JOIN :-** Returns all records from **right table** and **common records** from **both** the tables.
* **FULL OUTER JOIN :-** Returns all records when there is a **match** in either **left or right** table. If **no match** found in one table , **NULL** values are returned for columns for that table.
* ***How are joins used to combine data from multiple tables?***

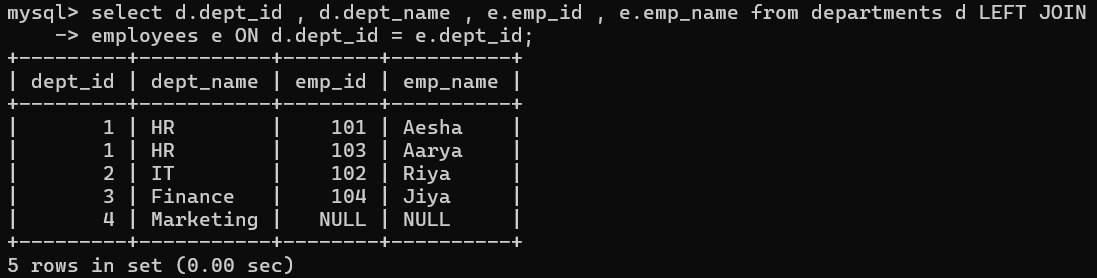
For example, I have:- A **departments** table (dept\_id, dept\_name)

An **employees** table (emp\_id, emp\_name, dept\_id)

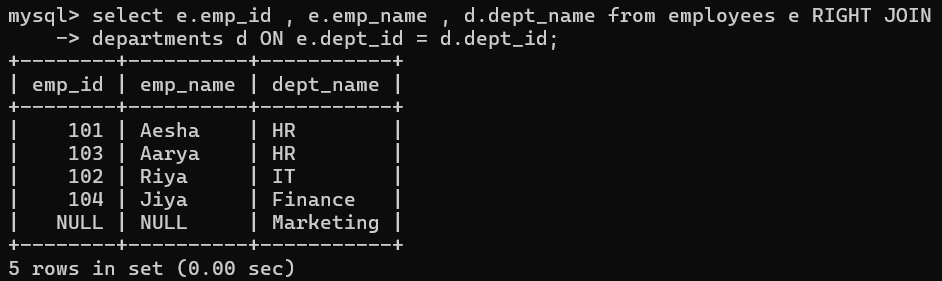
You can join them on the **dept\_id** column.

In INNER JOIN :- show in lab exercise 1st example.

In LEFT JOIN :- It works like this

****

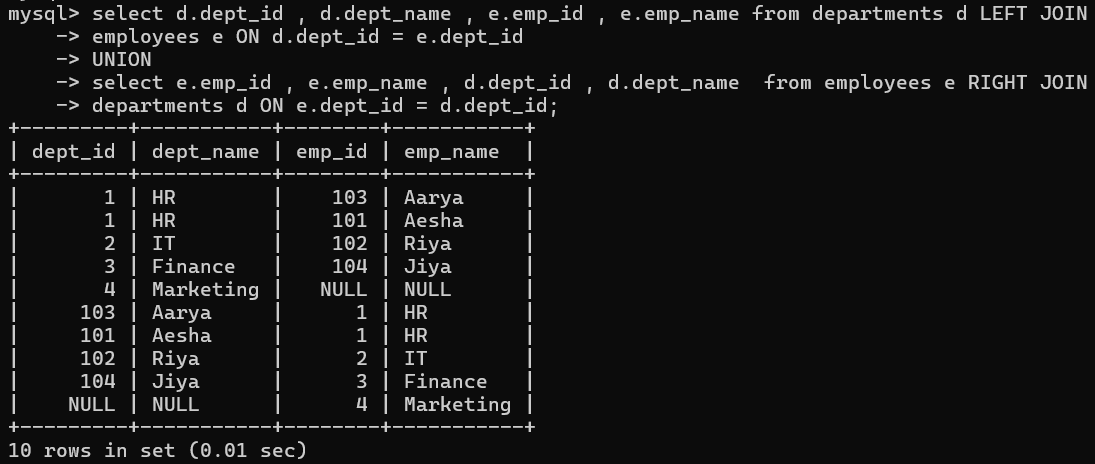
In RIGHT JOIN :- Returns all rows from the right table, and matching rows from the left table.



In FULL OUTER JOIN :- Employees with a matching department → shown together.

Departments with no employees → still shown, employee columns will be **NULL**.

Employees without a department → still shown, department columns will be **NULL**.

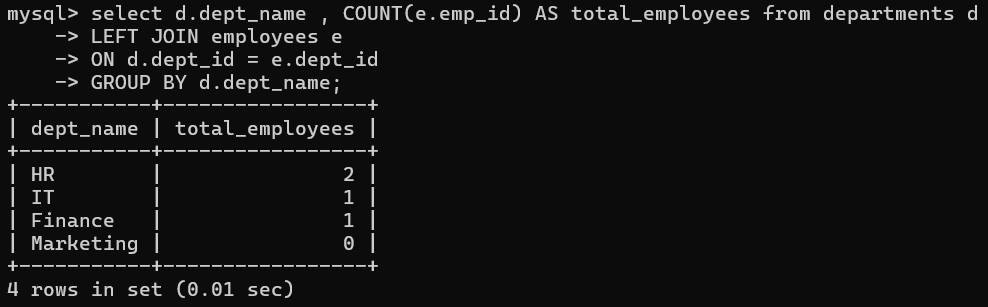


1. **SQL Group By**

**Lab Exercises :-**

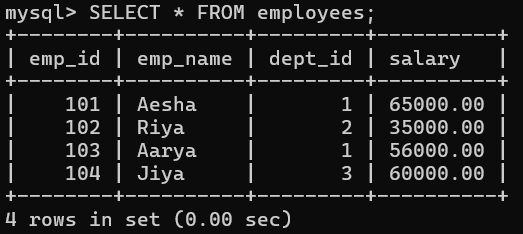
* Group employees by department and count the number of employees in each department using GROUP BY.

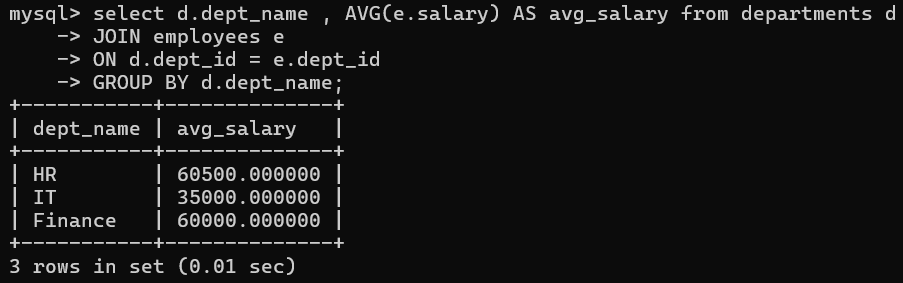
Show all departments, even those without employees → **LEFT JOIN** is used.



* Use the AVG aggregate function to find the average salary of employees in each department.

Added salary field (using alter ) and inserted values (using update).





**Theory Exercises :-**

* ***What is the GROUP BY clause in SQL? How is it used with aggregate functions?***
* The **GROUP BY** clause is used to **arrange rows** into **groups** based on the values of one or more columns.
* It is mainly used with **aggregate functions** like COUNT(), SUM(), AVG(), MAX(), MIN() to perform calculations for each group.
* ***Explain the difference between GROUP BY and ORDER BY.***

**GROUP BY :-** Used to **group rows** that have the same values in specified columns.

* Often used with **aggregate functions** (COUNT, SUM, AVG, etc.).
* Ex :- SELECT dept\_id, COUNT(\*) FROM employees GROUP BY dept\_id;
* Groups employees by department and counts them.

**ORDER BY :-** Used to **sort the result set** (ascending by default, descending with DESC).

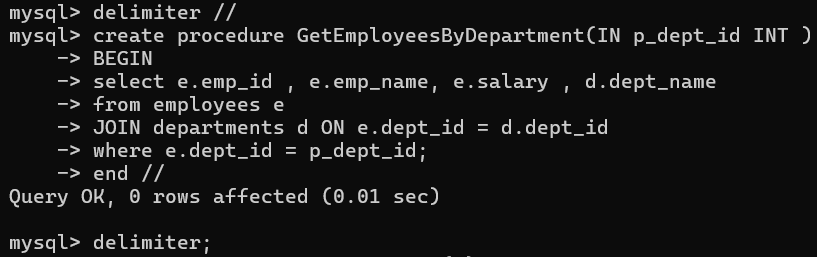
* Does **not group**, only changes the order of rows.
* Ex :- SELECT emp\_name, salary FROM employees ORDER BY salary DESC;
* Sorts employees by salary (highest first).

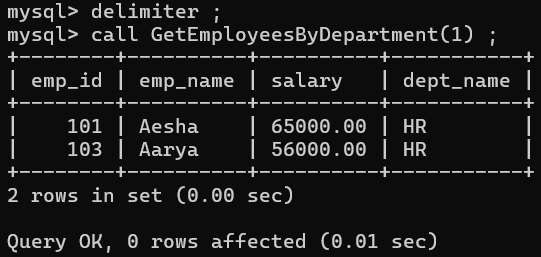
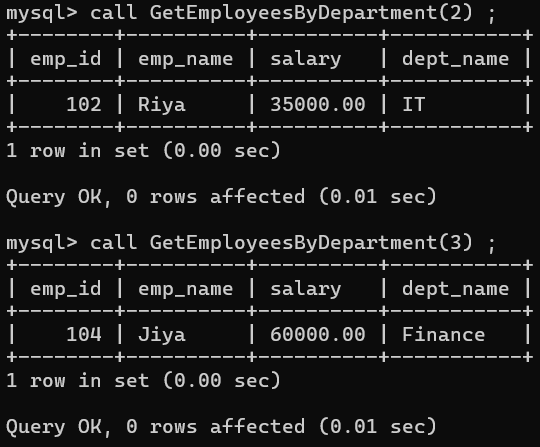
1. **SQL Stored Procedure**

**Lab Exercises :-**

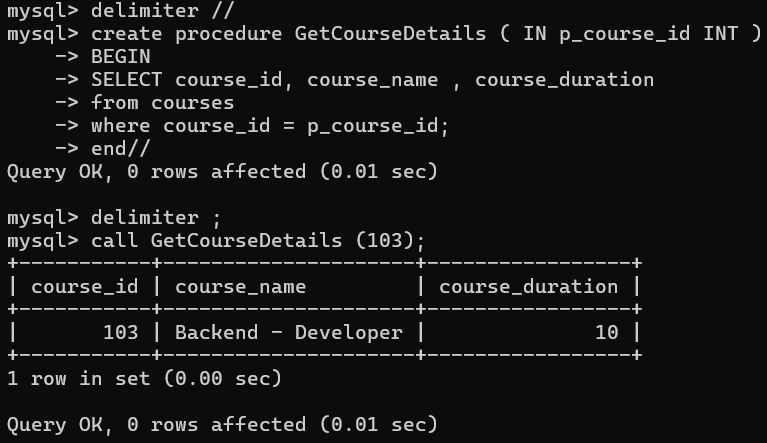
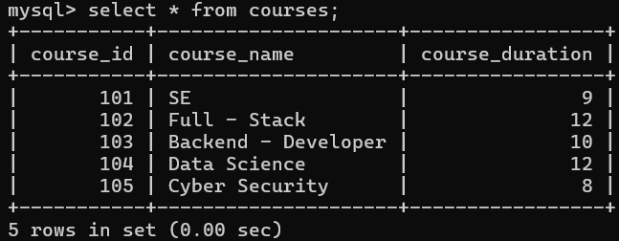
* Write a stored procedure to retrieve all employees from the employees table based on department.

(Delimiter is just a symbol (like ; or //) that tells MySQL **where one command ends**.)so after ending procedure set delimiter to **;**



* Write a stored procedure that accepts course\_id as input and returns the course details.



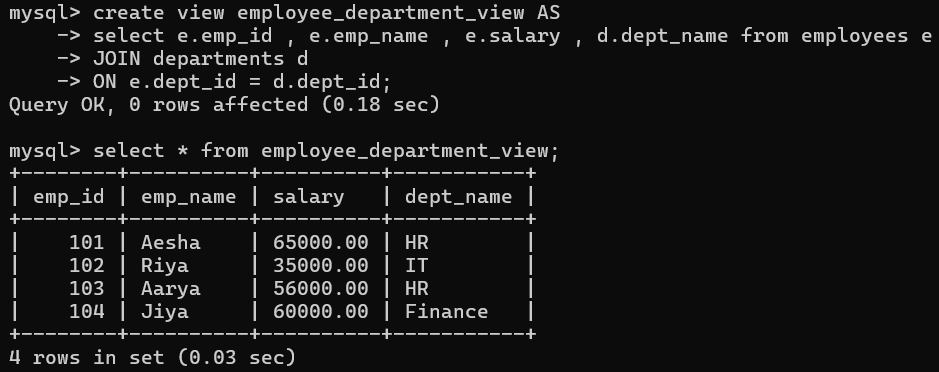
**Theory Exercises :-**

* ***What is a stored procedure in SQL, and how does it differ from a standard SQL query?***
* A **Stored Procedure** is a precompiled set of one or more SQL statements (like SELECT, INSERT, UPDATE, DELETE) that is **stored in the database** and can be executed by just calling it.
* It is like a **function** in programming that you can reuse whenever you need.
* **Accept Parameters** – Can take input (IN), return output (OUT), or both (INOUT).
* While **Standard SQL** statement are executed on demand . It must be written / executed every time. SQL is a one – time manual command.
* ***Explain the advantages of using stored procedures.***
* Faster execution (precompiled).
* Reusable & modular.
* More secure (limit table access).
* Easy to maintain (update once, use everywhere).
* Reduce network traffic (one call instead of many queries).

1. **SQL View**

**Lab Exercises :-**

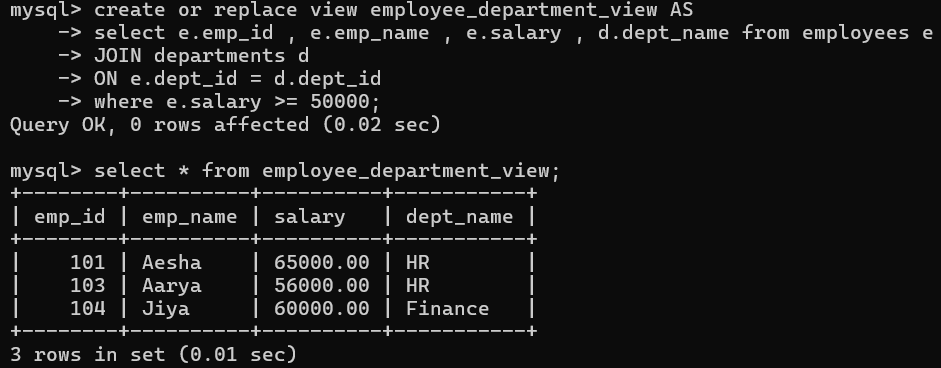
* Create a view to show all employees along with their department names.



To see the created view :- 

* Modify the view to exclude employees whose salaries are below $50,000.

To modify view **create or replace** command is used



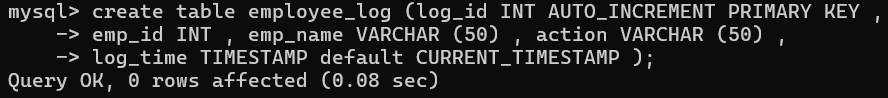
**Theory Exercises :-**

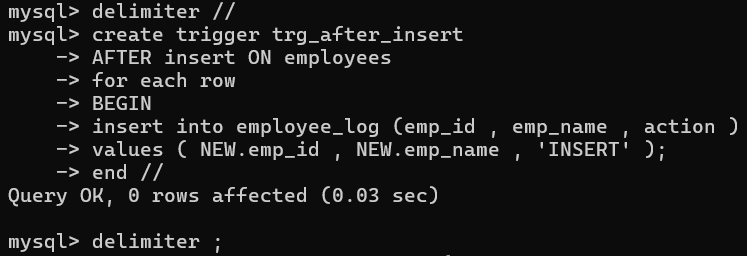
* ***What is a view in SQL, and how is it different from a table?***
* A **View** is a **virtual table** based on the result of an SQL query.
* It does **not store data physically**, it just shows data from one or more tables.
* **Table** contains actual records while **View** displays data from tables.
* Creation of table **create table** while for view **create view.**
* ***Explain the advantages of using views in SQL databases.***
* **Simplifies complex queries** – Saves a query and reuses it easily.
* **Data security** – Restrict access to specific columns/rows instead of full table.
* **Data abstraction** – Hides table structure, shows only necessary info.
* **Consistency** – Ensures users always see the same query result.

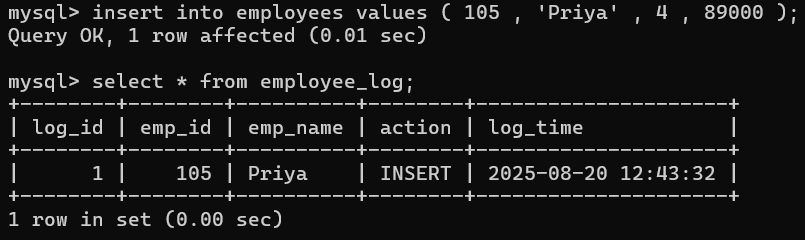
1. **SQL Triggers**

**Lab Exercises :-**

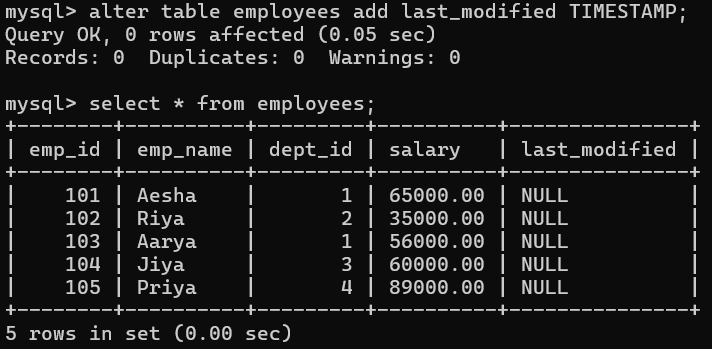
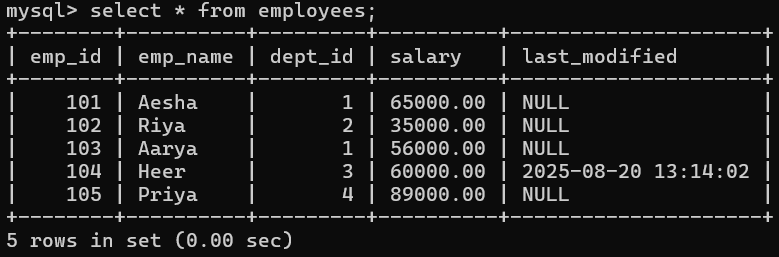
* Create a trigger to automatically log changes to the employees table when a new employee is added.

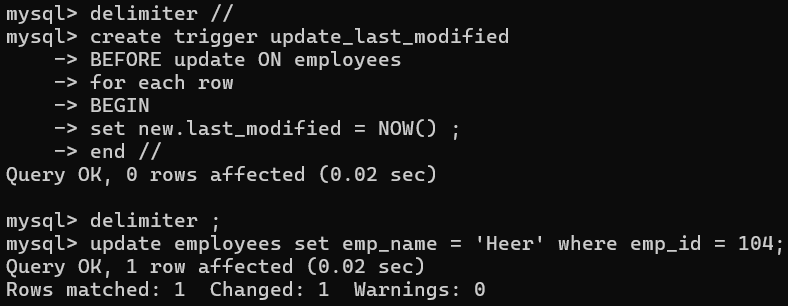






* Create a trigger to update the last\_modified timestamp whenever an employee record is updated.





**Theory Exercises :-**

* ***What is a trigger in SQL? Describe its types and when they are used.***
* **Trigger** is a special type of stored procedure that **automatically executed** or “fired” when a **specific event** occurs in a database. Events like insert , update , delete .
* Triggers are automatically invoked by DBMS.
* **BEFORE Trigger** :- Executes before the triggering event occurs.

Example: Validate data before inserting.

* **AFTER Trigger** :- Executes after the triggering event occurs.

Example: Log changes into an audit table.

When they are used :- To **maintain audit logs** (track who updated/deleted data).

To **synchronize tables** automatically.

To **prevent invalid transactions**.

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

**1. INSERT Trigger :-** Fires **when a new row is inserted** into a table.

* Can be **BEFORE INSERT** (validate data before adding) or **AFTER INSERT** (notify after adding).  
  Ex :- Log new employee details whenever a new employee is added.

**2. UPDATE Trigger :-** Fires **when an existing row is updated**.

* Can be **BEFORE UPDATE** (check new values) or **AFTER UPDATE** (record changes).  
  Ex :- Track salary changes of employees.

**3. DELETE Trigger :-** Fires **when a row is deleted**.

* Can be **BEFORE DELETE** (prevent deletion if needed) or **AFTER DELETE** (store old data in archive table).  
  Ex :- Move deleted employee details to a history table.

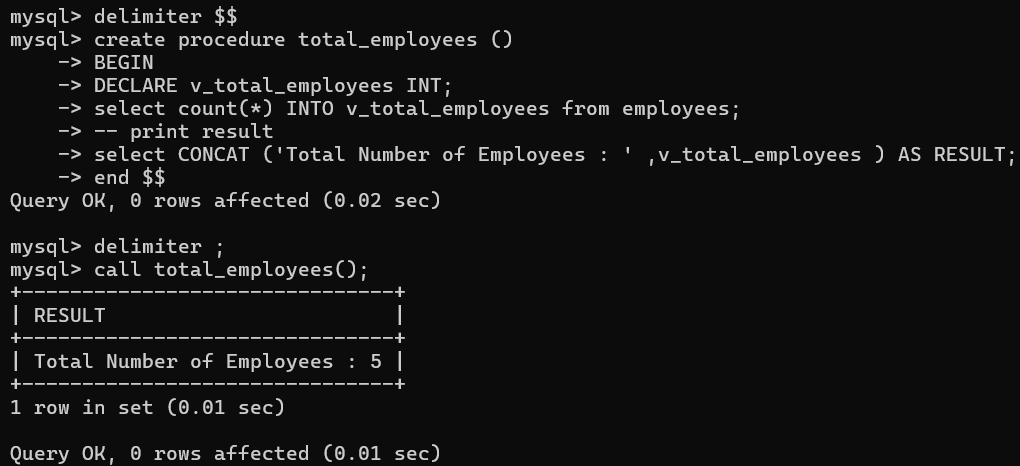
1. **Introduction to PL/SQL**

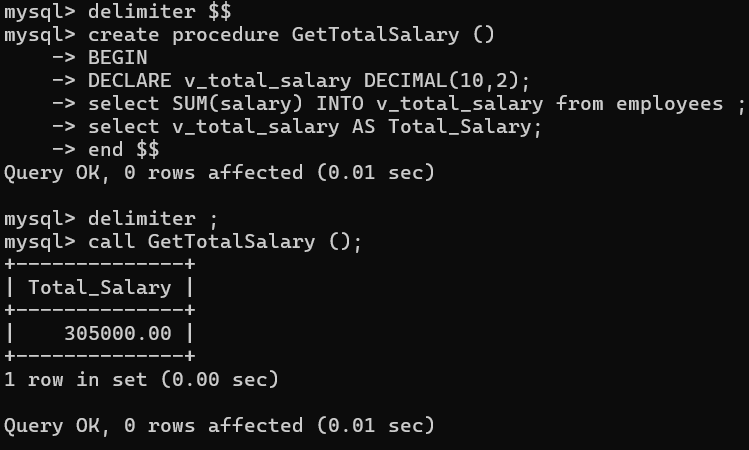
**Lab Exercises :-**

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

To see all the created procedures in current database :- 

To see a particular procedure :- **SHOW CREATE PROCEDURE total\_employees ;**



* Create a PL/SQL block that calculates the total salary from an employees table.

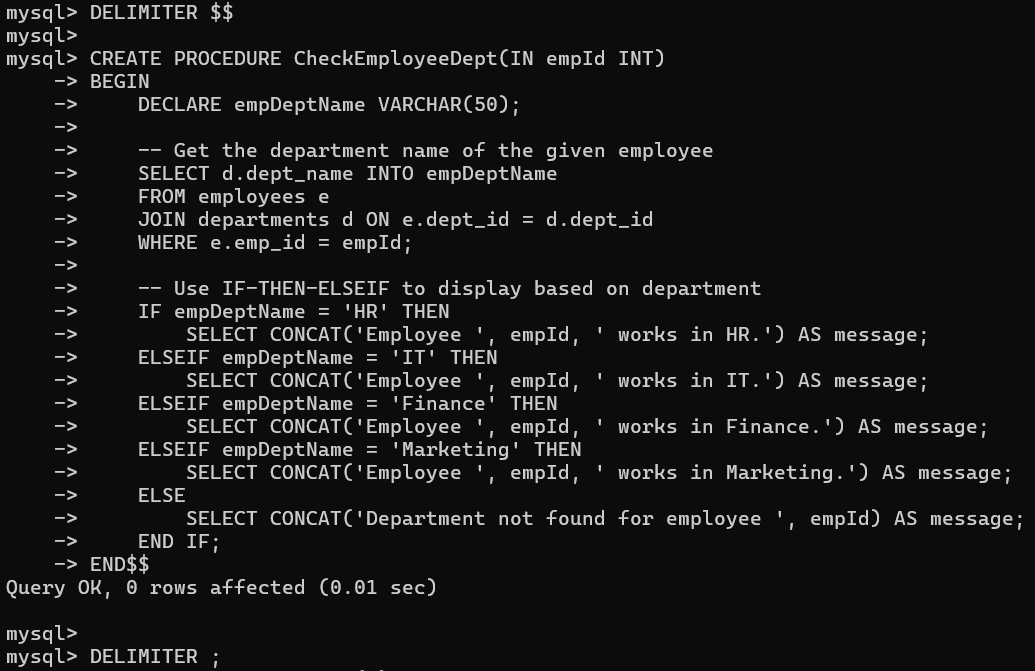
**Theory Exercises :-**

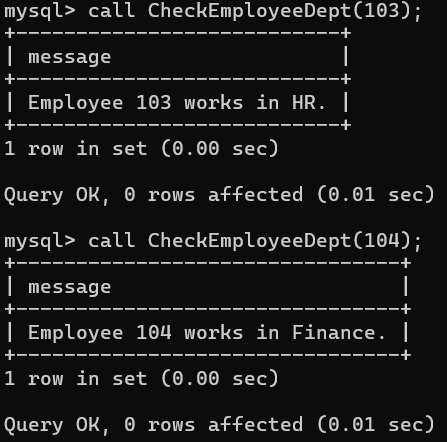
* ***What is PL/SQL, and how does it extend SQL's capabilities?***
* **Procedural language extensions to SQL** is a block – structured language that **combines** the **power of SQL** with the **features** of **procedural programming** such as loops , conditions , functions and error handling.
* To use PL/SQL in MYSQL we have to create stored procedures and then have to write PL/SQL block of code in it .
* ***List and explain the benefits of using PL/SQL.***
* Block-structured (easy to manage)
* Faster performance (less network traffic)
* Supports loops & conditions (procedural)
* Secure (hides logic, controls access)
* Portable within Oracle

1. **PL/SQL Control Structures**

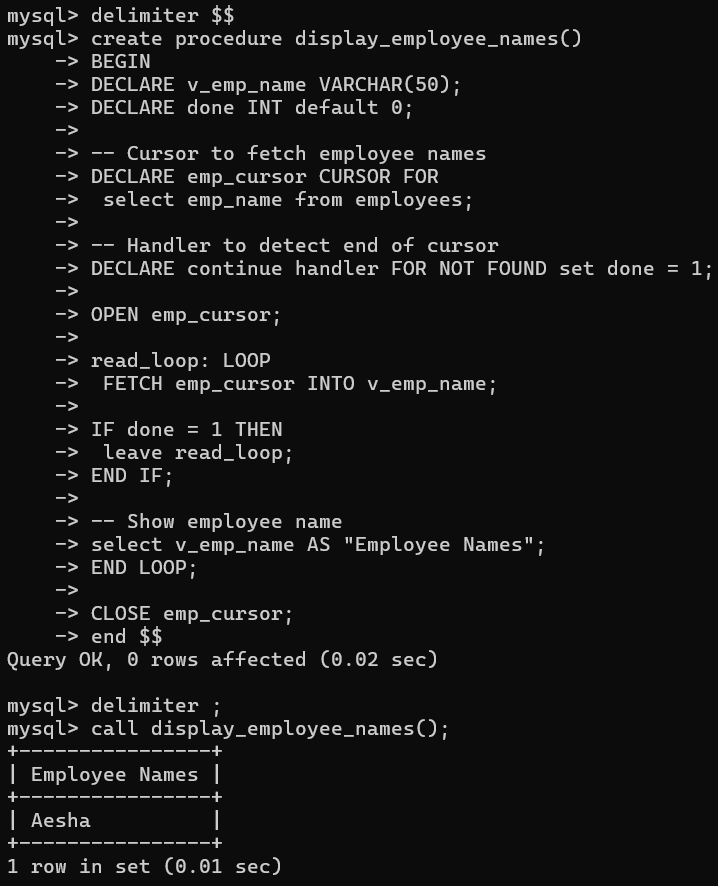
**Lab Exercises :-**

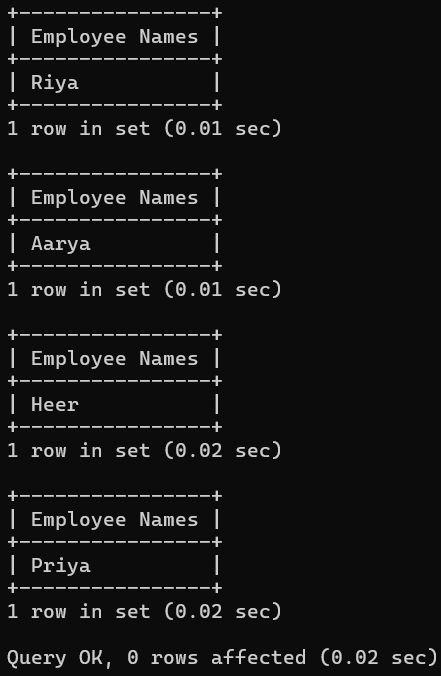
* Write a PL/SQL block using an IF-THEN condition to check the department of an employee.





* Use a FOR LOOP to iterate through employee records and display their names.





**Theory Exercises :-**

* ***What are control structures in PL/SQL? Explain the IF-THEN and LOOP control structures.***
* Control structures let you control the flow of execution in a PL/SQL block.
* **IF-THEN** **:-** Used for decision-making.

IF condition THEN Ex :- IF sal>80000 THEN

-- statements set pcustomerlevel = ‘platinum’;

Select pcustomerlevel;

END IF; END IF;

* **LOOP :-** Repeats a block of statements until explicitly exited.

LOOP

-- statements

EXIT WHEN condition;

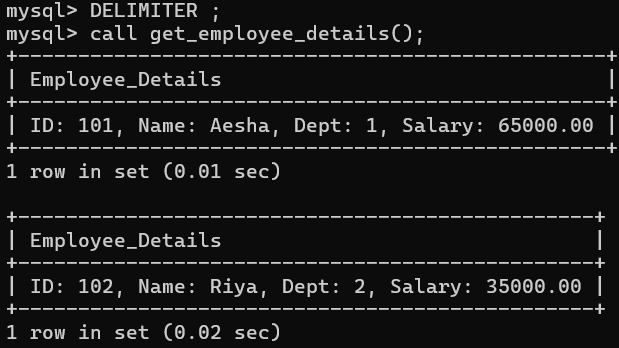
END LOOP;

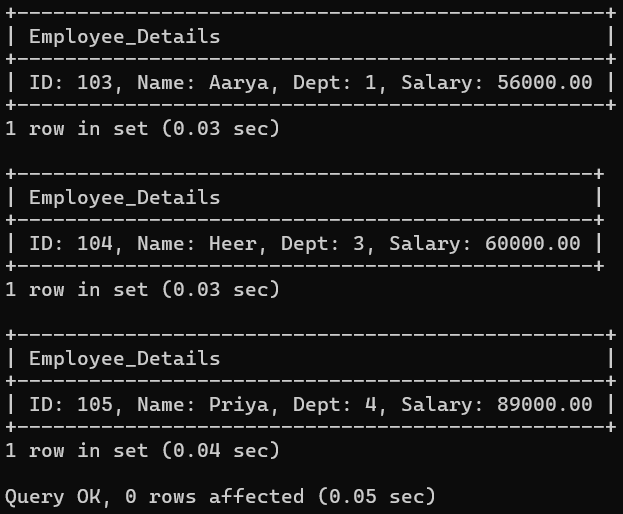
* ***How do control structures in PL/SQL help in writing complex queries?***
* **Decision-making** → Using **IF-THEN** , you can apply conditions dynamically (e.g., apply different logic for different departments).
* **Iteration** → Using **LOOP**, **WHILE**, or **FOR**, you can repeatedly execute SQL statements (e.g., process each row one by one).
* **Flexibility** → Combine **SQL** with **procedural** **logic**, making it possible to handle exceptions, calculations, and conditional flows.

1. **SQL Cursors**

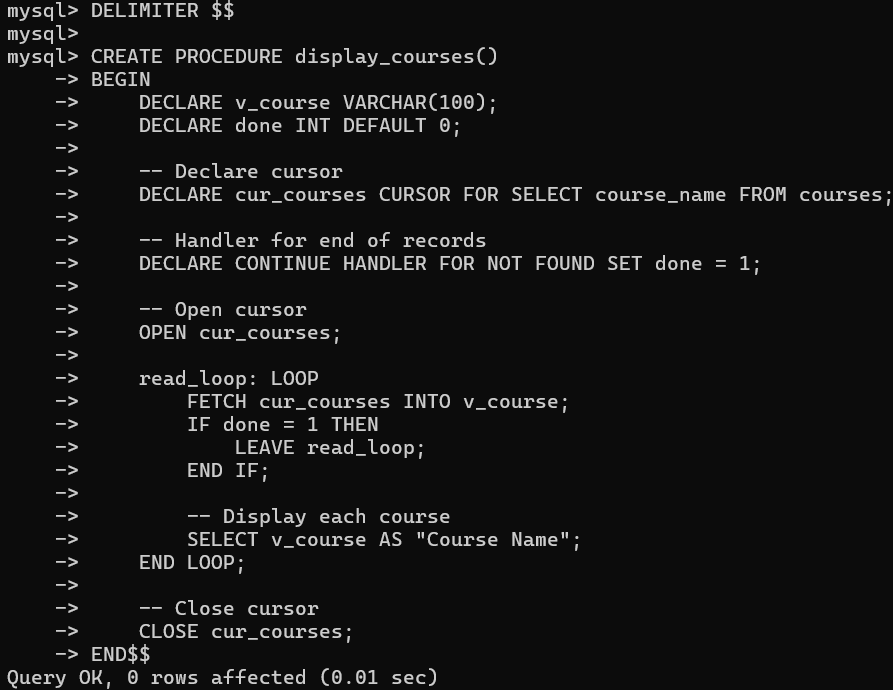
**Lab Exercises :-**

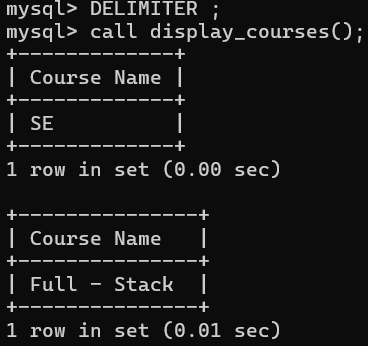
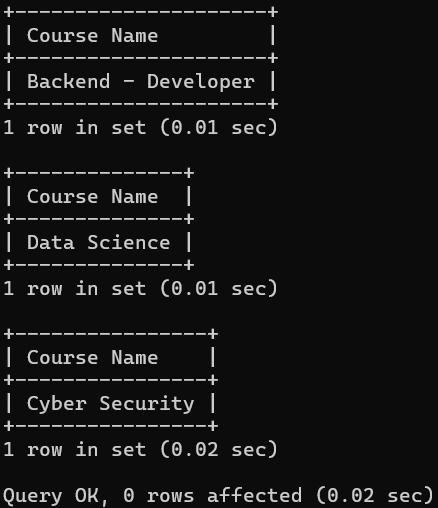
* Write a PL/SQL block using an explicit cursor to retrieve and display employee details.





* Create a cursor to retrieve all courses and display them one by one.



**Theory Exercises :-**

* ***What is a cursor in PL/SQL? Explain the difference between implicit and explicit cursors.***
* A **Cursor** is database object (like pointer) that allows for **row-by-row processing** of a result set returned by a query.
* SQL queries operate on entire sets of data , cursors enable you to iterate through individual rows and perform specific operations on each one.
* ***Implicit Cursor :-*** Automatically created by the database (MySQL, Oracle, etc.) whenever you run a **single SQL statement** like INSERT, UPDATE, DELETE, or SELECT ... INTO.
* **Control:** You don’t declare or open it yourself. The database engine handles it behind the scenes.
* **Use Case:** When you are executing a **single-row query** or a DML statement.
* ***Explicit Cursor :-*** A cursor that you **manually declare, open, fetch, and close**.
* **Control:** You write code to control how rows are fetched (one by one, or in loops).
* **Use Case:** Needed when the query **returns multiple rows** (cannot be handled by implicit cursors).
* ***When would you use an explicit cursor over an implicit one?***

**Implicit cursor :-** SELECT emp\_name INTO v\_name FROM employees WHERE emp\_id=101; (only one row).

**Explicit cursor :-** Retrieve all employees and display them one by one (multi-row).

**You would use an explicit cursor when:**

* **The query returns multiple rows** – implicit cursors can only handle single-row queries.
* **You need to process rows one by one** (row-by-row operations).
* **You need to loop through results** and perform calculations or logic on each record.
* In MySQL we only work with **explicit cursors** (inside stored procedures/functions).
* The term "implicit cursor" is more of a **PL/SQL (Oracle)** concept, not a MySQL feature.