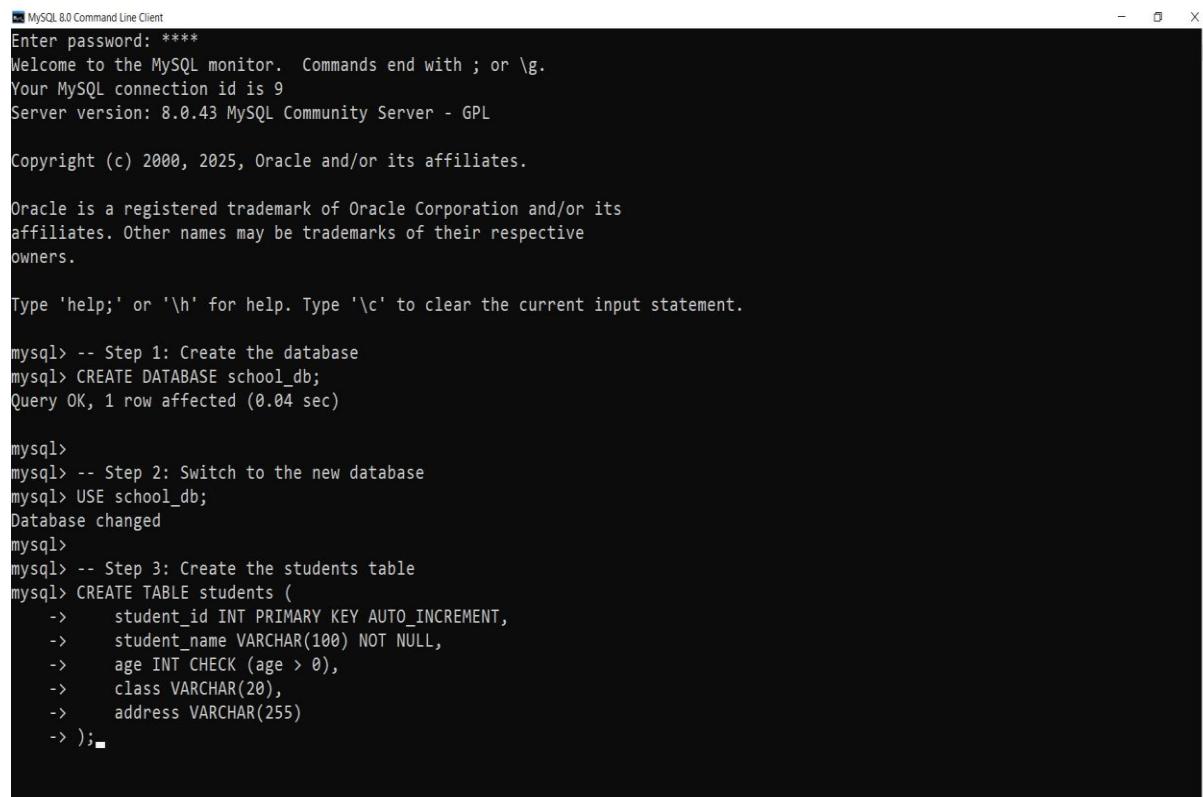


MODULE-4 DBMS

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.



The screenshot shows a terminal window for MySQL 8.0 Command Line Client. It starts with the MySQL welcome message and copyright notice. Then, it shows the creation of a database named 'school_db' and switching to it. Finally, it creates a table named 'students' with five columns: 'student_id' (primary key, auto-increment), 'student_name' (VARCHAR(100), not null), 'age' (INT, check constraint age > 0), 'class' (VARCHAR(20)), and 'address' (VARCHAR(255)).

```
MySQL 8.0 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.43 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> -- Step 1: Create the database
mysql> CREATE DATABASE school_db;
Query OK, 1 row affected (0.04 sec)

mysql>
mysql> -- Step 2: Switch to the new database
mysql> USE school_db;
Database changed
mysql>
mysql> -- Step 3: Create the students table
mysql> CREATE TABLE students (
    ->     student_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     student_name VARCHAR(100) NOT NULL,
    ->     age INT CHECK (age > 0),
    ->     class VARCHAR(20),
    ->     address VARCHAR(255)
    -> );■
```

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

```
MySQL 8.0 Command Line Client
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.43 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> -- Step 1: Open (use) the database
mysql> USE school_db;
Database changed
mysql>
mysql> -- Step 2: Insert 5 records into the students table
mysql> INSERT INTO students (student_name, age, class, address) VALUES
    -> ('Aarav Sharma', 14, '8A', 'Delhi'),
    -> ('Priya Patel', 15, '9B', 'Ahmedabad'),
    -> ('Rohan Mehta', 13, '7C', 'Mumbai'),
    -> ('Sneha Reddy', 16, '10A', 'Hyderabad'),
    -> ('Karan Singh', 12, '6B', 'Chandigarh');
Query OK, 5 rows affected (0.03 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
mysql> -- Step 3: Retrieve all records to check inserted data
mysql> SELECT * FROM students;
```

```
MySQL 8.0 Command Line Client
mysql>
mysql> -- Step 3: Retrieve all records to check inserted data
mysql> SELECT * FROM students;
+-----+-----+-----+-----+
| student_id | student_name | age | class | address |
+-----+-----+-----+-----+
| 1 | Aarav Sharma | 14 | 8A | Delhi |
| 2 | Priya Patel | 15 | 9B | Ahmedabad |
| 3 | Rohan Mehta | 13 | 7C | Mumbai |
| 4 | Sneha Reddy | 16 | 10A | Hyderabad |
| 5 | Karan Singh | 12 | 6B | Chandigarh |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

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

```
MySQL 8.0 Command Line Client

mysql> SELECT student_name, age
-> FROM students;
+-----+-----+
| student_name | age |
+-----+-----+
| Aarav Sharma | 14 |
| Priya Patel | 15 |
| Rohan Mehta | 13 |
| Sneha Reddy | 16 |
| Karan Singh | 12 |
+-----+-----+
5 rows in set (0.00 sec)

mysql> -
```

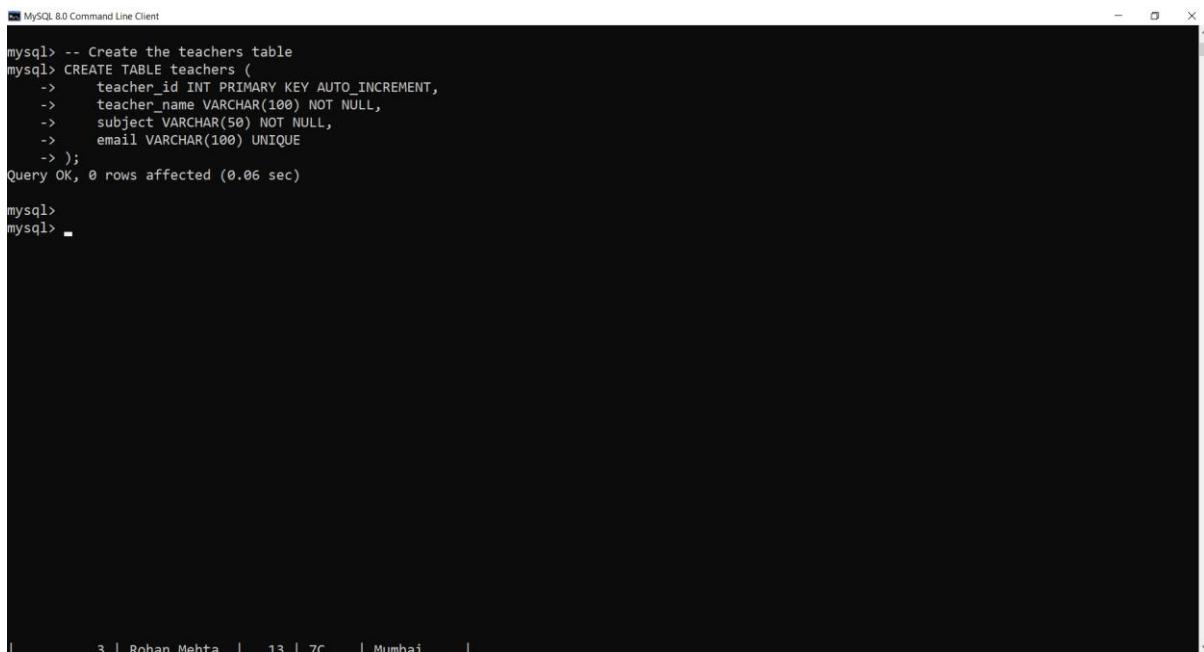
4. Write SQL queries to retrieve all students whose age is greater than 10.

```
MySQL 8.0 Command Line Client

mysql> SELECT *
-> FROM students
-> WHERE age > 10;
+-----+-----+-----+-----+
| student_id | student_name | age | class | address |
+-----+-----+-----+-----+
| 1 | Aarav Sharma | 14 | 8A | Delhi |
| 2 | Priya Patel | 15 | 9B | Ahmedabad |
| 3 | Rohan Mehta | 13 | 7C | Mumbai |
| 4 | Sneha Reddy | 16 | 10A | Hyderabad |
| 5 | Karan Singh | 12 | 6B | Chandigarh |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

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



The screenshot shows a terminal window titled "MySQL 8.0 Command Line Client". The user has run the following SQL command to create a table named "teachers":

```
mysql> -- Create the teachers table
mysql> CREATE TABLE teachers (
    ->     teacher_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     teacher_name VARCHAR(100) NOT NULL,
    ->     subject VARCHAR(50) NOT NULL,
    ->     email VARCHAR(100) UNIQUE
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> -
```

The command creates a table with four columns: teacher_id (INT, primary key, auto-increment), teacher_name (VARCHAR(100), NOT NULL), subject (VARCHAR(50), NOT NULL), and email (VARCHAR(100), UNIQUE). The query is successful, with 0 rows affected in 0.06 seconds.

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

```
MySQL 8.0 Command Line Client

mysql> -- Step 1: Alter the students table to add teacher_id column
mysql> ALTER TABLE students
    -> ADD teacher_id INT;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> -- Step 2: Add FOREIGN KEY constraint linking to teachers table
mysql> ALTER TABLE students
    -> ADD CONSTRAINT fk_teacher
        -> FOREIGN KEY (teacher_id) REFERENCES teachers(teacher_id);
Query OK, 5 rows affected (0.10 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
```

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

```
MySQL 8.0 Command Line Client

mysql> ^C
mysql> -- Create the courses table
mysql> CREATE TABLE courses (
    ->     course_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     course_name VARCHAR(100) NOT NULL,
    ->     course_credits INT CHECK (course_credits > 0)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> -
```

8. Use the CREATE command to create a database university_db.

```
MySQL 8.0 Command Line Client
-> );
Query OK, 0 rows affected (0.02 sec)

mysql> -- Create the university_db database
mysql> CREATE DATABASE university_db;
Query OK, 1 row affected (0.01 sec)

mysql>
```

```
MySQL 8.0 Command Line Client
-> );
Query OK, 0 rows affected (0.02 sec)

mysql> -- Create the university_db database
mysql> CREATE DATABASE university_db;
Query OK, 1 row affected (0.01 sec)

mysql> -- Open the database
mysql> USE university_db;
Database changed
mysql>
```

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

```
MySQL 8.0 Command Line Client
mysql> use school_db;
Database changed
mysql> -- Add a new column course_duration to courses table
mysql> ALTER TABLE courses
-> ADD course_duration VARCHAR(50);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql>
```

10. Drop the course_credits column from the **courses** table.

```
MySQL 8.0 Command Line Client
mysql> ^C
mysql> -- Drop the course_credits column
mysql> ALTER TABLE courses
-> DROP COLUMN course_credits;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> ■
```

11. Drop the **teachers** table from the **school_db** database.

Step 1: Find the foreign key name:

```
MySQL 8.0 Command Line Client

mysql> -- Select the database
mysql> USE school_db;
Database changed
mysql> -- Drop the teachers table
mysql> DROP TABLE teachers;
ERROR 3730 (HY000): Cannot drop table 'teachers' referenced by a foreign key constraint 'fk_teacher' on table 'students'.
mysql> -- Show foreign keys on students table
mysql> SHOW CREATE TABLE students;
+-----+-----+
| Table | Create Table
+-----+-----+
| students | CREATE TABLE `students` (
  `student_id` int NOT NULL AUTO_INCREMENT,
  `student_name` varchar(100) NOT NULL,
  `age` int DEFAULT NULL,
  `class` varchar(20) DEFAULT NULL,
  `address` varchar(255) DEFAULT NULL,
  `teacher_id` int DEFAULT NULL,
  PRIMARY KEY (`student_id`),
  KEY `fk_teacher` (`teacher_id`),
  CONSTRAINT `fk_teacher` FOREIGN KEY (`teacher_id`) REFERENCES `teachers` (`teacher_id`),
  CONSTRAINT `students_chk_1` CHECK ((`age` > 0))
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Step 2: Drop the foreign key constraint:

```
MySQL 8.0 Command Line Client

mysql> -- Remove the foreign key from students
mysql> ALTER TABLE students
      -> DROP FOREIGN KEY fk_teacher;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

Step 3: Drop the teachers table:

```
MySQL 8.0 Command Line Client

mysql> -- Drop the teachers table safely
mysql> DROP TABLE teachers;
Query OK, 0 rows affected (0.04 sec)

mysql> ■
```

12. Drop the students table from the school_db database and verify that the table has been removed

```
MySQL 8.0 Command Line Client

mysql> -- Step 1: Select the database
mysql> USE school_db;
Database changed
mysql>
mysql> -- Step 2: Drop the students table
mysql> DROP TABLE students;
Query OK, 0 rows affected (0.03 sec)

mysql>
mysql> -- Step 3: Verify that the table is removed
mysql> SHOW TABLES;
+-----+
| Tables_in_school_db |
+-----+
| courses           |
+-----+
1 row in set (0.02 sec)

mysql> ■
```

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

```
MySQL> -- Insert 3 new courses into the courses table
MySQL> INSERT INTO courses (course_name, course_duration) VALUES
    -> ('Physics', '6 Months'),
    -> ('Economics', '1 Year'),
    -> ('Data Science', '9 Months');
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

MySQL>
MySQL> -- Verify the records
MySQL> SELECT * FROM courses;
+-----+-----+-----+
| course_id | course_name | course_duration |
+-----+-----+-----+
|      1 | Physics     | 6 Months       |
|      2 | Economics   | 1 Year        |
|      3 | Data Science | 9 Months      |
+-----+-----+-----+
3 rows in set (0.00 sec)

MySQL> ■
```

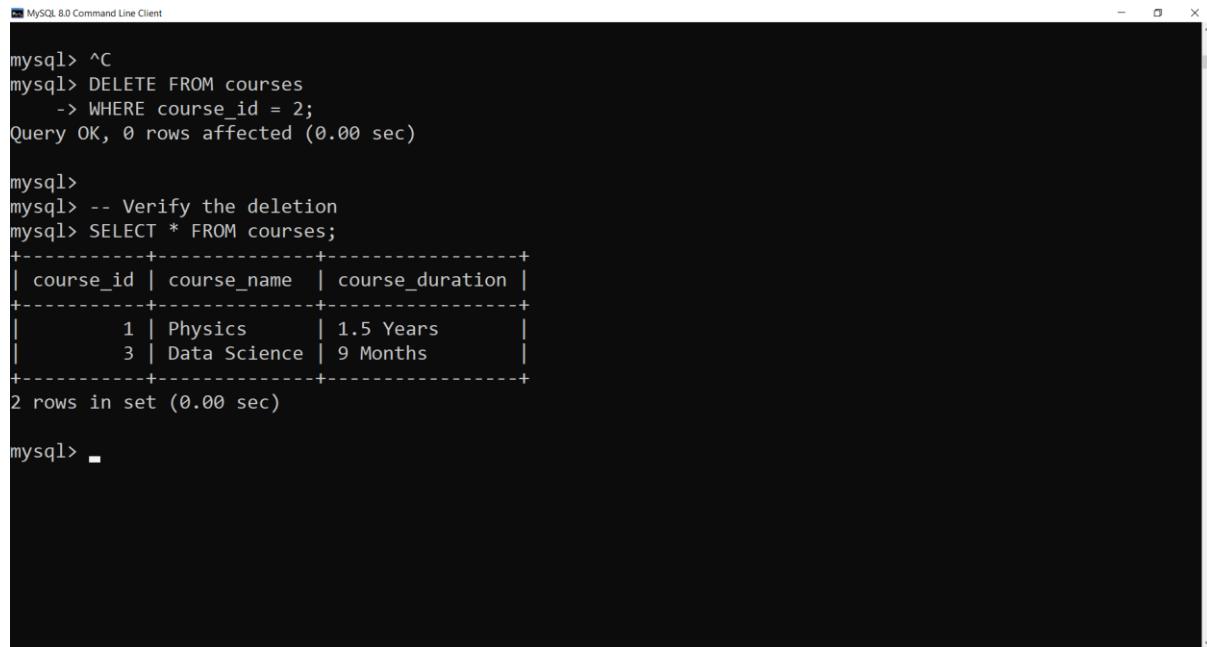
14. : Update the course duration of a specific course using the UPDATE command

```
MySQL> UPDATE courses
    -> SET course_duration = '1.5 Years'
    -> WHERE course_id = 1;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

MySQL>
MySQL> -- Verify the update
MySQL> SELECT * FROM courses WHERE course_id = 1;
+-----+-----+-----+
| course_id | course_name | course_duration |
+-----+-----+-----+
|      1 | Physics     | 1.5 Years      |
+-----+-----+-----+
1 row in set (0.00 sec)

MySQL> ■
```

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



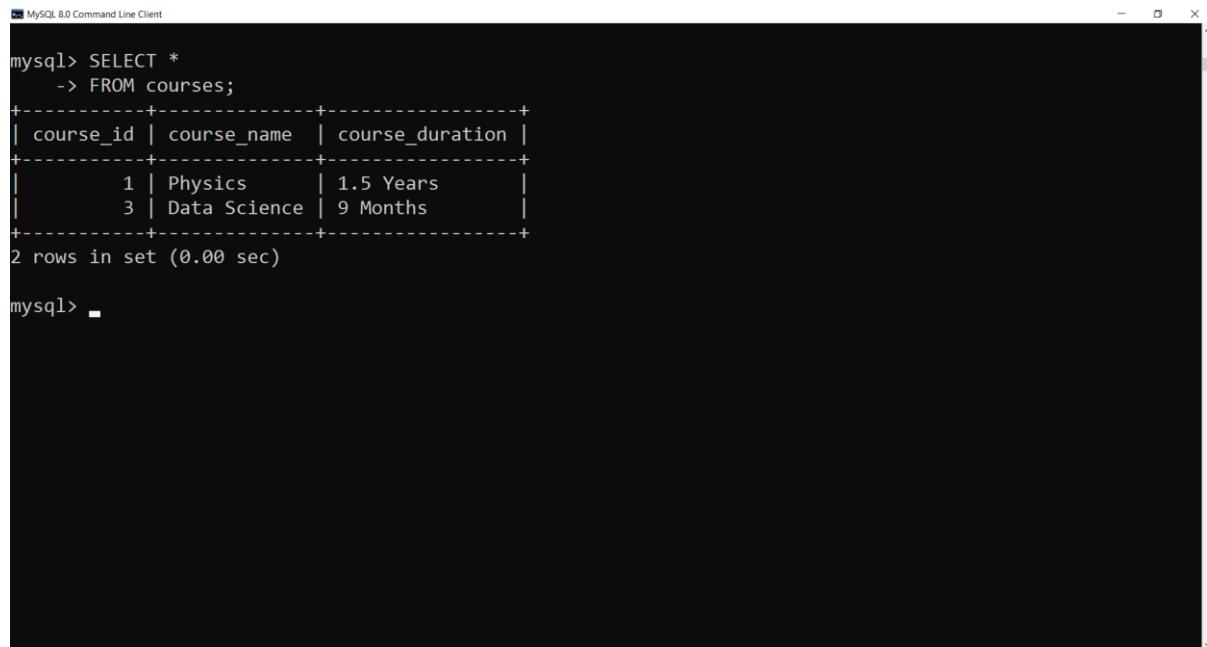
```
MySQL 8.0 Command Line Client

mysql> ^C
mysql> DELETE FROM courses
      -> WHERE course_id = 2;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> -- Verify the deletion
mysql> SELECT * FROM courses;
+-----+-----+-----+
| course_id | course_name | course_duration |
+-----+-----+-----+
|      1 | Physics     | 1.5 Years      |
|      3 | Data Science | 9 Months       |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> .
```

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



```
MySQL 8.0 Command Line Client

mysql> SELECT *
      -> FROM courses;
+-----+-----+-----+
| course_id | course_name | course_duration |
+-----+-----+-----+
|      1 | Physics     | 1.5 Years      |
|      3 | Data Science | 9 Months       |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> .
```

17. Sort the courses based on course_duration in descending order using ORDER BY.

```
MySQL 8.0 Command Line Client

mysql> ^C
mysql> SELECT *
   -> FROM courses
   -> ORDER BY course_duration DESC;
+-----+-----+-----+
| course_id | course_name | course_duration |
+-----+-----+-----+
|      3 | Data Science | 9 Months
|      1 | Physics      | 1.5 Years
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

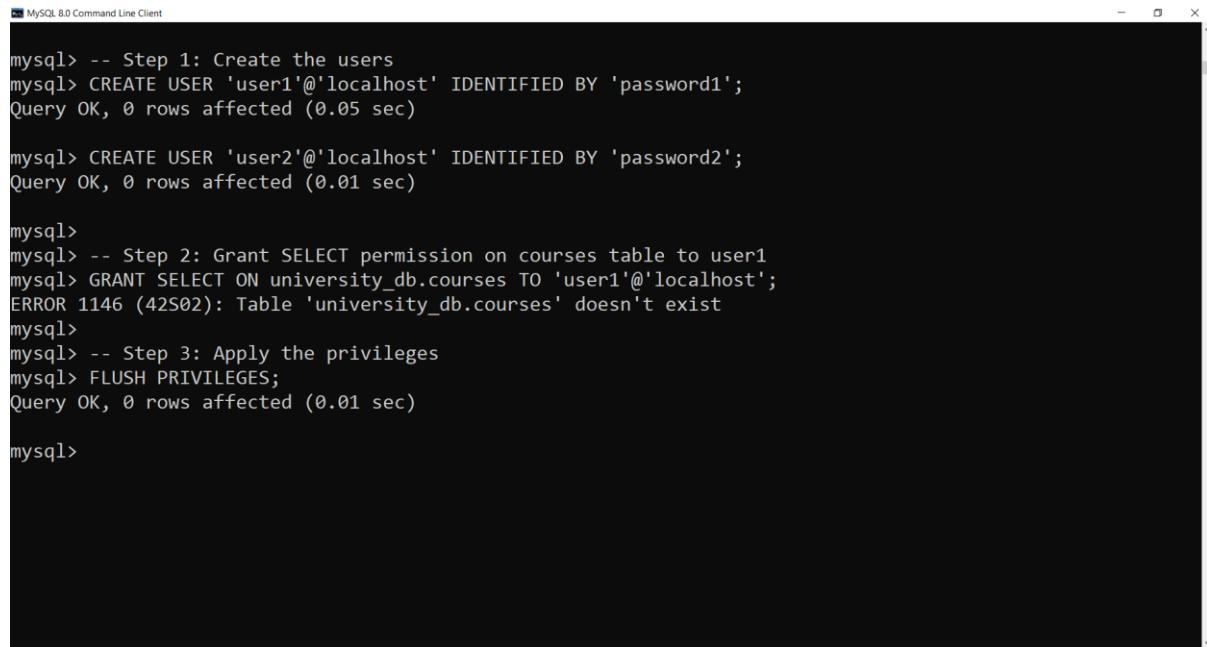
18. Limit the results of the SELECT query to show only the top two courses using LIMIT.

```
MySQL 8.0 Command Line Client

mysql> ^C
mysql> ^C
mysql> SELECT course_id, course_name, course_duration
   -> FROM courses
   -> ORDER BY course_duration DESC
   -> LIMIT 2;
+-----+-----+-----+
| course_id | course_name | course_duration |
+-----+-----+-----+
|      3 | Data Science | 9 Months
|      1 | Physics      | 1.5 Years
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

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



```
MySQL 8.0 Command Line Client

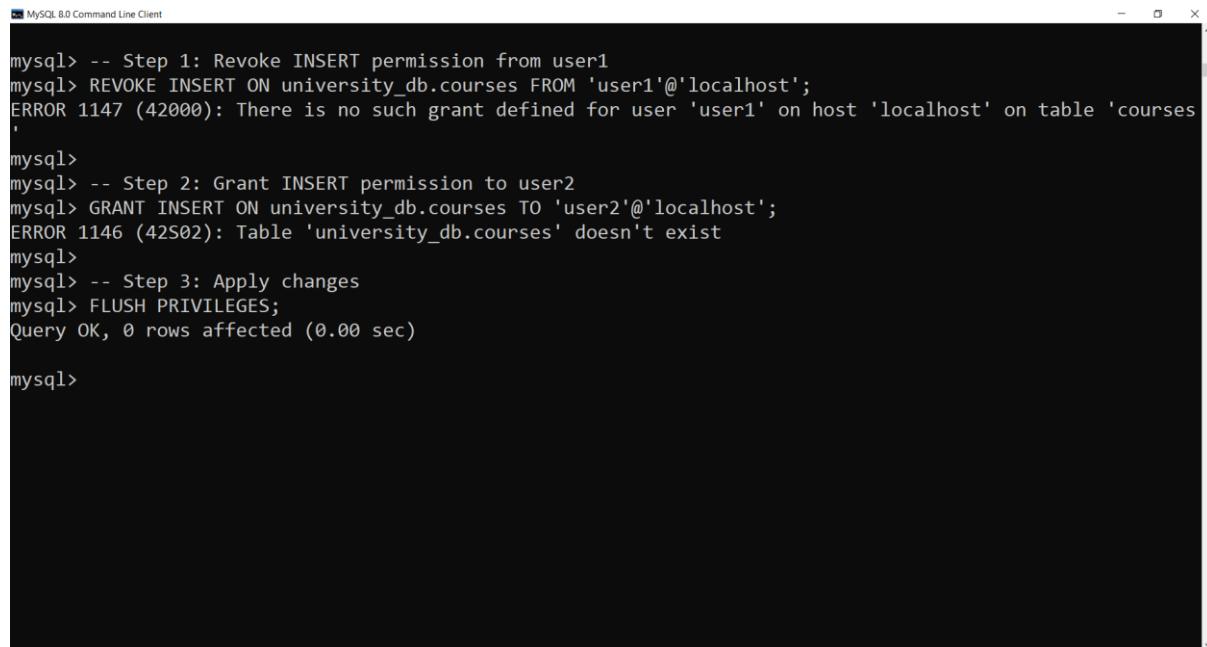
mysql> -- Step 1: Create the users
mysql> CREATE USER 'user1'@'localhost' IDENTIFIED BY 'password1';
Query OK, 0 rows affected (0.05 sec)

mysql> CREATE USER 'user2'@'localhost' IDENTIFIED BY 'password2';
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> -- Step 2: Grant SELECT permission on courses table to user1
mysql> GRANT SELECT ON university_db.courses TO 'user1'@'localhost';
ERROR 1146 (42S02): Table 'university_db.courses' doesn't exist
mysql>
mysql> -- Step 3: Apply the privileges
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

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



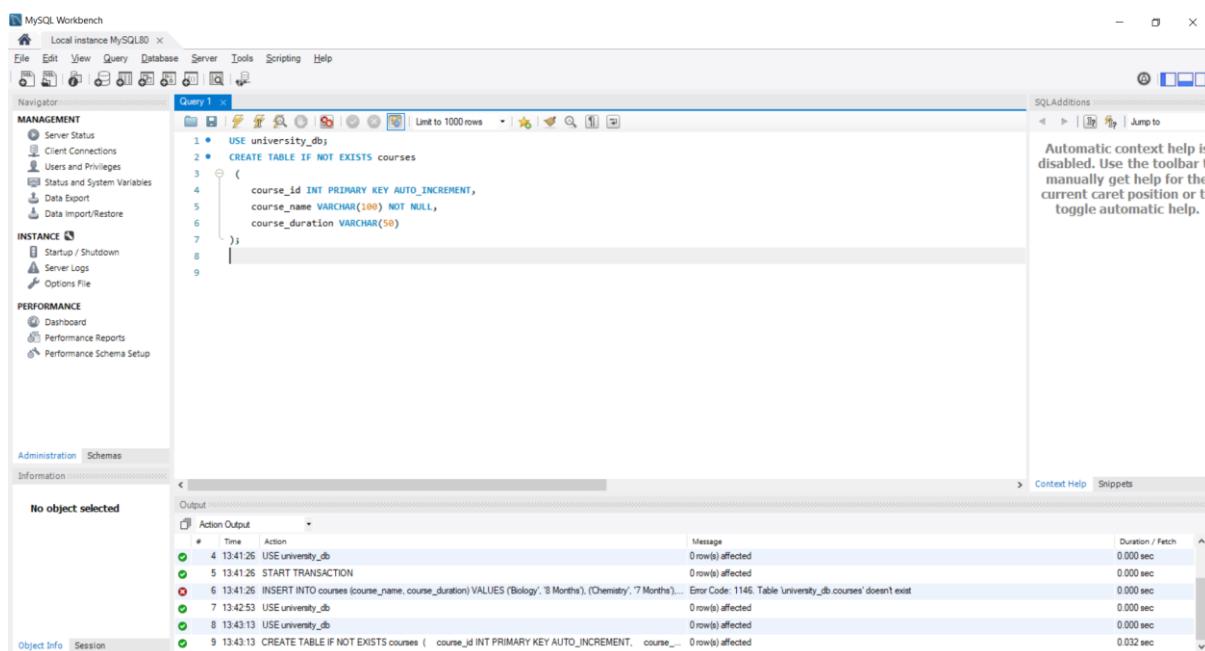
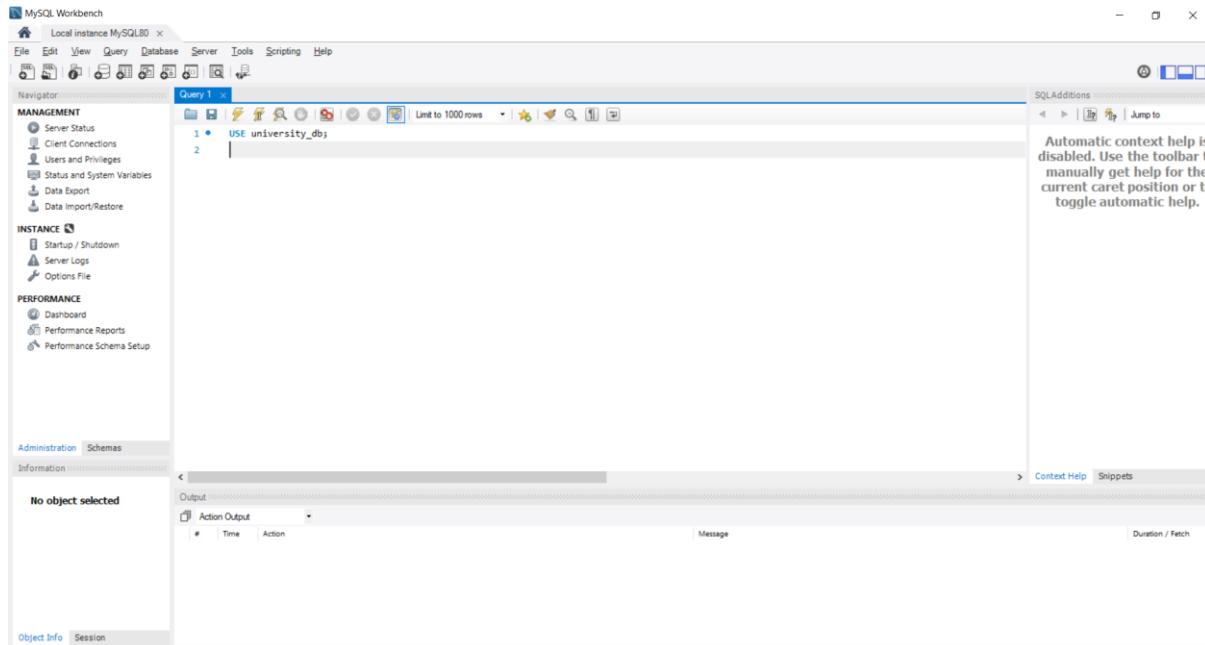
```
MySQL 8.0 Command Line Client

mysql> -- Step 1: Revoke INSERT permission from user1
mysql> REVOKE INSERT ON university_db.courses FROM 'user1'@'localhost';
ERROR 1147 (42000): There is no such grant defined for user 'user1' on host 'localhost' on table 'courses'
'

mysql>
mysql> -- Step 2: Grant INSERT permission to user2
mysql> GRANT INSERT ON university_db.courses TO 'user2'@'localhost';
ERROR 1146 (42S02): Table 'university_db.courses' doesn't exist
mysql>
mysql> -- Step 3: Apply changes
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

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



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the following SQL code:


```

1 • USE university_db
2 • CREATE TABLE IF NOT EXISTS courses
3 • (
4     course_id INT PRIMARY KEY AUTO_INCREMENT,
5     course_name VARCHAR(100) NOT NULL,
6     course_duration VARCHAR(50)
7 );
8 • start TRANSACTION;
9

```
- Output Window:** Shows the execution results:

#	Time	Action	Message	Duration / Fetch
10	13:43:49	USE university_db	0 row(s) affected	0.000 sec
11	13:43:49	CREATE TABLE IF NOT EXISTS courses (course_id INT PRIMARY KEY AUTO_INCREMENT, course_name VARCHAR(100) NOT NULL, course_duration VARCHAR(50))	0 row(s) affected, 1 warning(s): 1050 Table 'courses' already exists	0.000 sec
12	13:43:49	start TRANSACTION	Error Code: 1054. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'start TRANSACTION' at line 1	0.015 sec
13	13:44:05	USE university_db	0 row(s) affected	0.016 sec
14	13:44:05	CREATE TABLE IF NOT EXISTS courses (course_id INT PRIMARY KEY AUTO_INCREMENT, course_name VARCHAR(100) NOT NULL, course_duration VARCHAR(50))	0 row(s) affected, 1 warning(s): 1050 Table 'courses' already exists	0.000 sec
15	13:44:05	stat TRANSACTION	0 row(s) affected	0.000 sec

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the following SQL code:


```

9 • INSERT INTO courses (course_name, course_duration) VALUES
10 ('Biology', '8 Months'),
11 ('Chemistry', '7 Months'),
12 ('Psychology', '1 Year');
13 • COMMIT;
14 • SELECT * FROM courses;
15

```
- Result Grid:** Displays the inserted data:

course_id	course_name	course_duration
1	Biology	8 Months
2	Chemistry	7 Months
3	Psychology	1 Year
4	Biology	8 Months
5	Chemistry	7 Months
6	Psychology	1 Year
7	Biology	8 Months
8	Chemistry	7 Months
9	Psychology	1 Year
- Output Window:** Shows the execution results:

#	Time	Action	Message	Duration / Fetch
25	13:45:03	USE university_db	0 row(s) affected	0.000 sec
26	13:45:03	CREATE TABLE IF NOT EXISTS courses (course_id INT PRIMARY KEY AUTO_INCREMENT, course_name VARCHAR(100) NOT NULL, course_duration VARCHAR(50))	0 row(s) affected, 1 warning(s): 1050 Table 'courses' already exists	0.000 sec
27	13:45:03	start TRANSACTION	0 row(s) affected	0.000 sec
28	13:45:03	INSERT INTO courses (course_name, course_duration) VALUES ('Biology', '8 Months'), ('Chemistry', '7 Months'), ('Psychology', '1 Year')	3 row(s) affected Records: 3 Duplicates: 0 Warnings: 0	0.000 sec
29	13:45:03	COMMIT	0 row(s) affected	0.000 sec
30	13:45:03	SELECT * FROM courses LIMIT 0, 1000	9 row(s) returned	0.000 sec / 0.000 sec

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

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

No object selected

Administration Schemas Information

Object Info Session Output

Query 1

```

15 • START TRANSACTION;
16 • INSERT INTO courses (course_name, course_duration) VALUES
    ('Philosophy', '6 Months'),
    ('Sociology', '1 Year');
18 • ROLLBACK;
20 • SELECT * FROM courses;
21
22
23
24
25
26

```

Result Grid | Filter Rows | Edit: | Export/Import: | Wrap Cell Content: |

course_id	course_name	course_duration
1	Biology	8 Months
2	Chemistry	7 Months
3	Psychology	1 Year
4	Biology	8 Months
5	Chemistry	7 Months
6	Psychology	1 Year
7	Biology	8 Months
8	Chemistry	7 Months
9	Psychology	1 Year
10	Biology	8 Months
11	Chemistry	7 Months
12	Psychology	1 Year
13	Biology	8 Months
14	Chemistry	7 Months
15	Psychology	1 Year
16	Biology	8 Months

courses 4 courses 5

Context Help Snippets

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

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

No object selected

Administration Schemas Information

Object Info Session Output

Query 1

```

25
26 -- Update some courses
27 • UPDATE courses
28 SET course_duration = '10 Months'
29 WHERE course_name = 'Biology';
30
31 • UPDATE courses
32 SET course_duration = '8 Months'
33 WHERE course_name = 'Chemistry';
34
35 -- Create a SAVEPOINT before making further changes
36 • SAVEPOINT before_chemistry_update;
37
38 -- Make more updates
39 • UPDATE courses
40 SET course_duration = '2 Years'
41 WHERE course_name = 'Chemistry';
42

```

Result Grid | Filter Rows | Edit: | Export/Import: | Wrap Cell Content: |

course_id	course_name	course_duration
1	Biology	8 Months
2	Chemistry	7 Months
3	Psychology	1 Year
4	Biology	8 Months
5	Chemistry	7 Months
6	Psychology	1 Year
7	Biology	8 Months
8	Chemistry	7 Months
9	Psychology	1 Year

courses 4 courses 5

Context Help Snippets

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Query 1

```

34 -- Create a SAVEPOINT before making further changes
35 • SAVEPOINT before_chemistry_update;
36
37
38 -- Make more updates
39 • UPDATE courses
40   SET course_duration = '2 Years'
41   WHERE course_name = 'Chemistry';
42
43 -- Roll back only the last update to the SAVEPOINT
44 • ROLLBACK TO SAVEPOINT before_chemistry_update;
45
46 -- Commit the remaining changes
47 • COMMIT;
48
49 -- Verify the results
50 • SELECT * FROM courses;
51

```

Result Grid

course_id	course_name	course_duration
1	Biology	8 Months
2	Chemistry	7 Months
3	Psychology	1 Year
4	Biology	8 Months
5	Chemistry	7 Months
6	Psychology	1 Year
7	Biology	8 Months
8	Chemistry	7 Months
9	Psychology	1 Year

No object selected

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

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

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Query 1

```

26 -- Update some courses
27 • UPDATE courses
28   SET course_duration = '10 Months'
29   WHERE course_name = 'Biology';
30
31 • UPDATE courses
32   SET course_duration = '8 Months'
33   WHERE course_name = 'Chemistry';
34
35 -- Create a SAVEPOINT before making further changes
36 • SAVEPOINT before_chemistry_update;
37
38 -- Make more updates
39 • UPDATE courses
40   SET course_duration = '2 Years'
41   WHERE course_name = 'Chemistry';
42
43 -- Roll back only the last update to the SAVEPOINT
44 • ROLLBACK TO SAVEPOINT before_chemistry_update;
45
46 -- Verify the results
47 • SELECT * FROM courses;
48

```

Result Grid

course_id	course_name	course_duration
1	Biology	8 Months
2	Chemistry	7 Months
3	Psychology	1 Year
4	Biology	8 Months
5	Chemistry	7 Months
6	Psychology	1 Year
7	Biology	8 Months
8	Chemistry	7 Months
9	Psychology	1 Year

No object selected

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Client Status
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Query 1

```

41 WHERE course_name = 'Chemistry';
42
43 -- Roll back only the last update to the SAVEPOINT
44 • ROLLBACK TO SAVEPOINT before_chemistry_update;
45
46 -- Commit the remaining changes
47 • COMMIT;
48
49 -- Verify the results
50 • SELECT * FROM courses;
51 • CREATE TABLE departments (
52     department_id INT PRIMARY KEY AUTO_INCREMENT,
53     department_name VARCHAR(100) NOT NULL
54 );
55 • CREATE TABLE employees (
56     employee_id INT PRIMARY KEY AUTO_INCREMENT,
57     employee_name VARCHAR(100) NOT NULL,
58     department_id INT,

```

Administration Schemas

Information

No object selected

Result Grid | Filter Rows | Edit: | Export/Import: | Wrap Cell Content: |

course_id	course_name	course_duration
1	Biology	8 Months
2	Chemistry	7 Months
3	Psychology	1 Year
4	Biology	8 Months
5	Chemistry	7 Months
6	Psychology	1 Year
7	Biology	8 Months
8	Chemistry	7 Months
9	Psychology	1 Year

courses 8 | courses 9 x

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Client Status
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Query 1

```

56     employee_id INT PRIMARY KEY AUTO_INCREMENT,
57     employee_name VARCHAR(100) NOT NULL,
58     department_id INT,
59     FOREIGN KEY (department_id) REFERENCES departments(department_id)
60 );
61 -- Insert departments
62 • INSERT INTO departments (department_name) VALUES
63     ('HR'),
64     ('Finance'),
65     ('IT'),
66     ('Marketing');
67
68 -- Insert employees
69 • INSERT INTO employees (employee_name, department_id) VALUES
70     ('Aarav Sharma', 1),
71     ('Priya Patel', 2),
72     ('Rohan Mehta', 3),
73     ('Sneha Reddy', 3),

```

Administration Schemas

Information

No object selected

Result Grid | Filter Rows | Edit: | Export/Import: | Wrap Cell Content: |

course_id	course_name	course_duration
1	Biology	8 Months
2	Chemistry	7 Months
3	Psychology	1 Year
4	Biology	8 Months
5	Chemistry	7 Months
6	Psychology	1 Year
7	Biology	8 Months
8	Chemistry	7 Months
9	Psychology	1 Year

courses 8 | courses 9 x

Object Info Session Output

SQLAdditions

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays sections like MANAGEMENT, INSTANCE, and PERFORMANCE. The central Query Editor pane contains the following SQL code:

```
-- Insert employees
INSERT INTO employees (employee_name, department_id) VALUES
('Aarav Sharma', 1),
('Priya Patel', 2),
('Rohan Mehta', 3),
('Sneha Reddy', 3),
('Karan Singh', 4);
SELECT e.employee_id, e.employee_name, d.department_name
FROM employees e
INNER JOIN departments d
ON e.department_id = d.department_id;
```

The bottom Results Grid pane shows the output of the last SELECT statement:

course_id	course_name	course_duration
1	Biology	8 Months
2	Chemistry	7 Months
3	Psychology	1 Year
4	Biology	8 Months
5	Chemistry	7 Months
6	Psychology	1 Year
7	Biology	8 Months
8	Chemistry	7 Months
9	Psychology	1 Year

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

The screenshot shows the MySQL Command Line Client window. The command entered is:

```
mysql> SELECT d.department_id, d.department_name, e.employee_id, e.employee_name
-> FROM departments d
-> LEFT JOIN employees e
-> ON d.department_id = e.department_id;
```

The resulting table output is:

department_id	department_name	employee_id	employee_name
1	HR	1	Aarav Sharma
2	Finance	2	Priya Patel
3	IT	3	Rohan Mehta
3	IT	4	Sneha Reddy
4	Marketing	5	Karan Singh
5	Sales	NULL	NULL

Text at the bottom of the client window:

```
6 rows in set (0.00 sec)

mysql> ■
```

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

```
MySQL 8.0 Command Line Client

mysql> SELECT d.department_name, COUNT(e.employee_id) AS num_employees
-> FROM departments d
-> LEFT JOIN employees e
-> ON d.department_id = e.department_id
-> GROUP BY d.department_name;
+-----+-----+
| department_name | num_employees |
+-----+-----+
| HR             |           1 |
| Finance        |           1 |
| IT             |           2 |
| Marketing      |           1 |
| Sales           |           0 |
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

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

```
MySQL 8.0 Command Line Client

mysql> ALTER TABLE employees
-> ADD COLUMN salary DECIMAL(10,2);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

```
MySQL 8.0 Command Line Client

mysql> UPDATE employees
   -> SET salary = CASE employee_name
   ->      WHEN 'Aarav Sharma' THEN 50000
   ->      WHEN 'Priya Patel' THEN 60000
   ->      WHEN 'Rohan Mehta' THEN 55000
   ->      WHEN 'Sneha Reddy' THEN 58000
   ->      WHEN 'Karan Singh' THEN 52000
   -> END;
Query OK, 5 rows affected (0.00 sec)
Rows matched: 5  Changed: 5  Warnings: 0

mysql> SELECT d.department_name, AVG(e.salary) AS avg_salary
   -> FROM departments d
   -> LEFT JOIN employees e
   -> ON d.department_id = e.department_id
   -> GROUP BY d.department_name;
+-----+-----+
| department_name | avg_salary |
+-----+-----+
| HR             | 50000.00000 |
| Finance        | 60000.00000 |
| IT             | 56500.00000 |
| Marketing       | 52000.00000 |
| Sales           |      NULL    |
+-----+-----+
5 rows in set (0.00 sec)
```

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

```
MySQL 8.0 Command Line Client
+-----+-----+
5 rows in set (0.00 sec)

mysql> DELIMITER $$

mysql> CREATE PROCEDURE GetEmployeesByDepartment(IN dept_name VARCHAR(100))
   -> BEGIN
   ->     SELECT employee_id, employee_name, salary
   ->     FROM employees
   ->     WHERE department_id = (
   ->         SELECT department_id
   ->         FROM departments
   ->         WHERE department_name = dept_name
   ->     );
   -> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql>
```

```
MySQL 8.0 Command Line Client
->      FROM departments
->      WHERE department_name = dept_name
->    );
-> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL GetEmployeesByDepartment('IT');
+-----+-----+-----+
| employee_id | employee_name | salary |
+-----+-----+-----+
|          3 | Rohan Mehta   | 55000.00 |
|          4 | Sneha Reddy   | 58000.00 |
+-----+-----+-----+
2 rows in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

mysql>
```

29. Write a stored procedure that accepts course_id as input and returns the course details.

```
MySQL 8.0 Command Line Client
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER $$

mysql> CREATE PROCEDURE GetCourseDetails(IN cid INT)
-> BEGIN
->   SELECT course_id, course_name, course_duration
->   FROM courses
->   WHERE course_id = cid;
-> END $$
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> DELIMITER ;
mysql> ■
```

```
MySQL 8.0 Command Line Client

mysql>
mysql> DELIMITER ;
mysql> CALL GetCourseDetails(1);
ERROR 1146 (42S02): Table 'company_db.courses' doesn't exist
mysql> -
```

Query OK, 0 rows affected (0.00 sec)

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

```
MySQL 8.0 Command Line Client

mysql> ^C
mysql> -- Create the view
mysql> CREATE VIEW EmployeeDepartments AS
-> SELECT e.employee_id, e.employee_name, e.salary, d.department_name
-> FROM employees e
-> INNER JOIN departments d
-> ON e.department_id = d.department_id;
Query OK, 0 rows affected (0.01 sec)

mysql> -- Retrieve all employees with their department names
mysql> SELECT * FROM EmployeeDepartments;
+-----+-----+-----+-----+
| employee_id | employee_name | salary | department_name |
+-----+-----+-----+-----+
| 1 | Aarav Sharma | 50000.00 | HR |
| 2 | Priya Patel | 60000.00 | Finance |
| 3 | Rohan Mehta | 55000.00 | IT |
| 4 | Sneha Reddy | 58000.00 | IT |
| 5 | Karan Singh | 52000.00 | Marketing |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Query OK, 0 rows affected (0.00 sec)

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

```
MySQL 8.0 Command Line Client

mysql> -- Drop the existing view
mysql> DROP VIEW IF EXISTS EmployeeDepartments;
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> -- Recreate the view with salary filter
mysql> CREATE VIEW EmployeeDepartments AS
-> SELECT e.employee_id, e.employee_name, e.salary, d.department_name
-> FROM employees e
-> INNER JOIN departments d
-> ON e.department_id = d.department_id
-> WHERE e.salary >= 50000;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM EmployeeDepartments;
+-----+-----+-----+-----+
| employee_id | employee_name | salary | department_name |
+-----+-----+-----+-----+
| 1 | Aarav Sharma | 50000.00 | HR
| 2 | Priya Patel | 60000.00 | Finance
| 3 | Rohan Mehta | 55000.00 | IT
| 4 | Sneha Reddy | 58000.00 | IT
| 5 | Karan Singh | 52000.00 | Marketing
+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

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

```
MySQL 8.0 Command Line Client

mysql> ^C
mysql> CREATE TABLE employee_log (
->     log_id INT PRIMARY KEY AUTO_INCREMENT,
->     employee_id INT,
->     employee_name VARCHAR(100),
->     department_id INT,
->     salary DECIMAL(10,2),
->     action_time DATETIME DEFAULT CURRENT_TIMESTAMP,
->     action VARCHAR(50)
-> );
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER $$

mysql> CREATE TRIGGER after_employee_insert
-> AFTER INSERT ON employees
-> FOR EACH ROW
-> BEGIN
->     INSERT INTO employee_log (employee_id, employee_name, department_id, salary, action)
->     VALUES (NEW.employee_id, NEW.employee_name, NEW.department_id, NEW.salary, 'INSERTED');
-> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> -
```

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

```
MySQL 8.0 Command Line Client
mysql>
mysql> DELIMITER ;
mysql> ^C
mysql> ^C
mysql> ^C
mysql>
mysql> ALTER TABLE employees
-> ADD COLUMN last_modified DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP;
Query OK, 0 rows affected (0.02 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> DELIMITER $$

mysql> CREATE TRIGGER before_employee_update
-> BEFORE UPDATE ON employees
-> FOR EACH ROW
-> BEGIN
->     SET NEW.last_modified = NOW();
-> END $$
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> DELIMITER ;
mysql> -
```

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

The screenshot shows a web browser window with the URL chatgpt.com/c/68d10a07-7840-832f-83bb-8b4b529078fe. The page title is "GitHub Dashboard". On the left, there's a sidebar with a list of chats: "New chat", "Search chats", "Library", "Sora", "GPTs", "Chats", and a selected item "PL/SQL employee count". The main content area contains a SQL code block:

```
sql
DECLARE
    v_total_employees NUMBER; -- variable to store count
BEGIN
    -- Count employees
    SELECT COUNT(*)
    INTO v_total_employees
    FROM employees;

    -- Print result
    DBMS_OUTPUT.PUT_LINE('Total number of employees: ' || v_total_employees);
END;
/
```

Below the code, there's a "Notes:" section with a checked checkbox:

- `DBMS_OUTPUT.PUT_LINE` is used to print the result in Oracle.
- Before running this block in SQL*Plus or SQL Developer, make sure to enable output using:

```
sql
SET SERVEROUTPUT ON;
```

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

table.

Here's a clean PL/SQL block that calculates the total sales from an `orders` table (assuming it has a column like `order_amount` or `total_price`):

```
sql
DECLARE
    v_total_sales NUMBER(10,2); -- variable to store sales total
BEGIN
    -- calculate total sales
    SELECT NVL(SUM(order_amount), 0)
    INTO v_total_sales
    FROM orders;

    -- Print result
    DBMS_OUTPUT.PUT_LINE('Total Sales: ' || v_total_sales);
END;
/
```

Explanation:

- `SUM(order_amount)` → adds up all sales values.

+ Ask anything

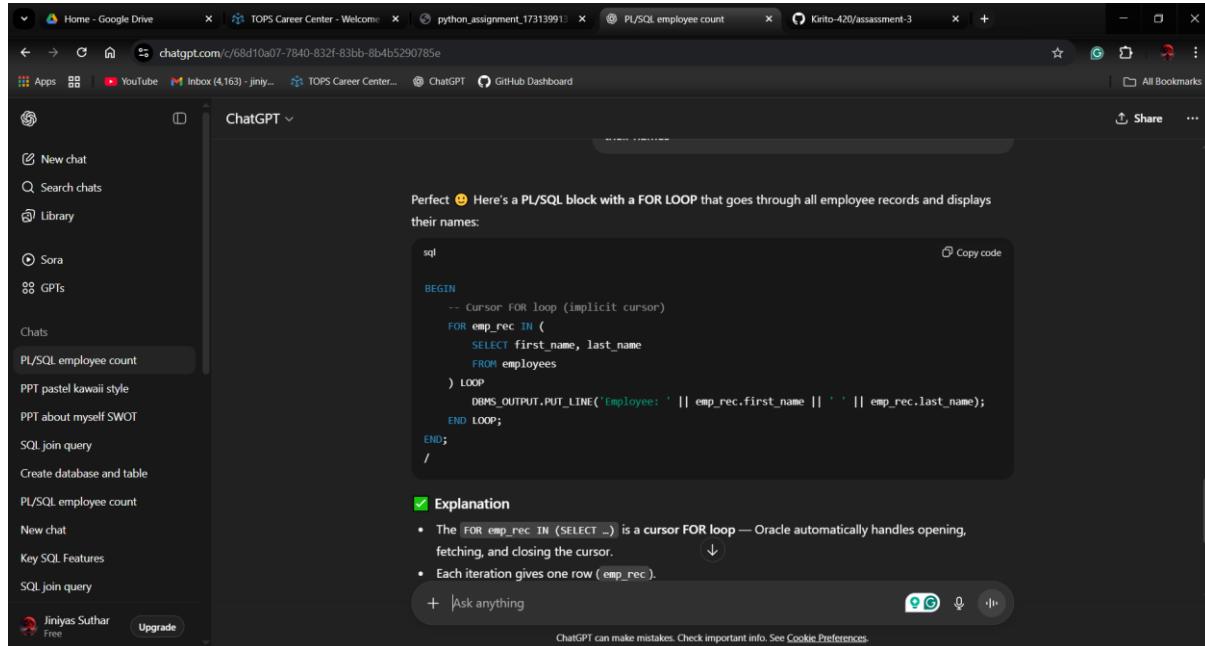
36. : Write a PL/SQL block using an IF-THEN condition to check the department of an employee.

```
sql
DECLARE
    v_employee_id employees.employee_id%TYPE := 101; -- employee to check
    v_department_id employees.department_id%TYPE; -- variable for dept
BEGIN
    -- Get department of the given employee
    SELECT department_id
    INTO v_department_id
    FROM employees
    WHERE employee_id = v_employee_id;

    -- Check department using IF-THEN
    IF v_department_id = 10 THEN
        DBMS_OUTPUT.PUT_LINE('Employee ' || v_employee_id || ' works in Administration.');
    ELSIF v_department_id = 20 THEN
        DBMS_OUTPUT.PUT_LINE('Employee ' || v_employee_id || ' works in Marketing.');
    ELSIF v_department_id = 30 THEN
        DBMS_OUTPUT.PUT_LINE('Employee ' || v_employee_id || ' works in Purchasing.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Employee ' || v_employee_id || ' works in another department.');
    END IF;
END;
/
```

+ Ask anything

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



Perfect 😊 Here's a PL/SQL block with a FOR LOOP that goes through all employee records and displays their names:

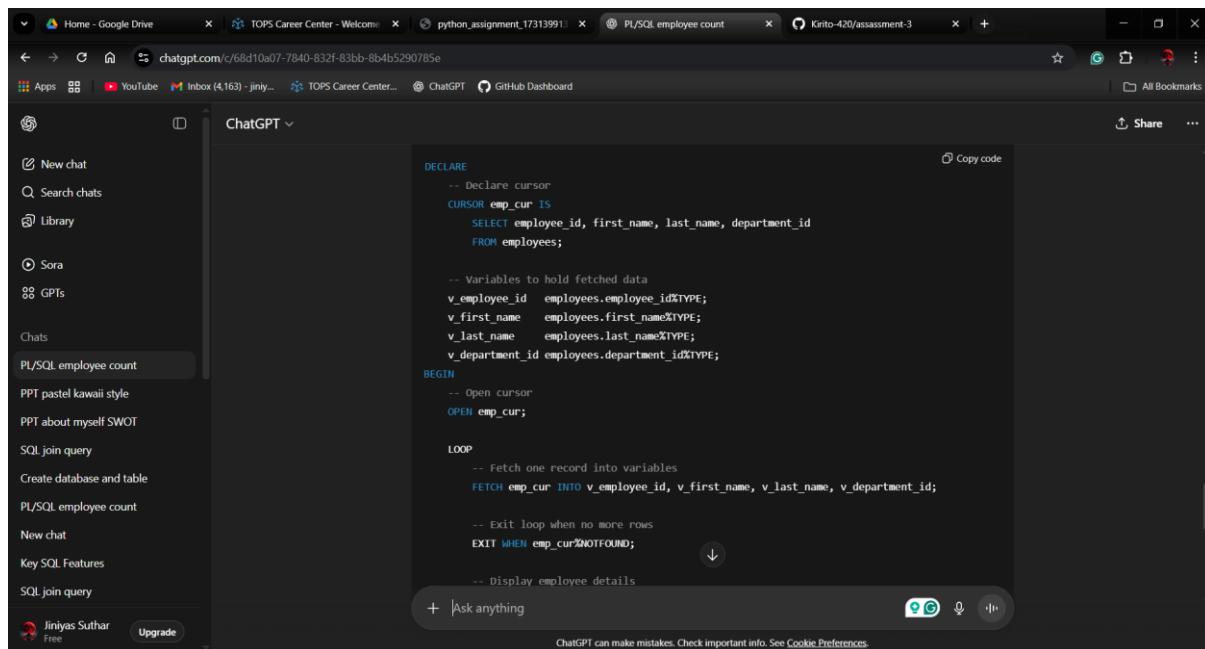
```
sql
BEGIN
    -- Cursor FOR loop (implicit cursor)
    FOR emp_rec IN (
        SELECT first_name, last_name
        FROM employees
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Employee: ' || emp_rec.first_name || ' ' || emp_rec.last_name);
    END LOOP;
END;
/
```

Explanation

- The `FOR emp_rec IN (SELECT ...)` is a cursor FOR loop — Oracle automatically handles opening, fetching, and closing the cursor.
- Each iteration gives one row (`emp_rec`).

+ Ask anything

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



```
DECLARE
    -- Declare cursor
    CURSOR emp_cur IS
        SELECT employee_id, first_name, last_name, department_id
        FROM employees;

    -- Variables to hold fetched data
    v_employee_id employees.employee_id%TYPE;
    v_first_name employees.first_name%TYPE;
    v_last_name employees.last_name%TYPE;
    v_department_id employees.department_id%TYPE;

BEGIN
    -- Open cursor
    OPEN emp_cur;

    LOOP
        -- Fetch one record into variables
        FETCH emp_cur INTO v_employee_id, v_first_name, v_last_name, v_department_id;

        -- Exit loop when no more rows
        EXIT WHEN emp_cur%NOTFOUND;

        -- Display employee details
    END LOOP;
END;
```

The screenshot shows the ChatGPT interface with a dark theme. On the left, there's a sidebar with various chat options like 'New chat', 'Search chats', 'Library', and a list of recent chats including 'PL/SQL employee count', 'PPT pastel kawaii style', etc. The main area displays a block of PL/SQL code:

```

-- Exit loop when no more rows
EXIT WHEN emp_cur%NOTFOUND;

-- Display employee details
DBMS_OUTPUT.PUT_LINE('Employee ID: ' || v_employee_id ||
                     ', Name: ' || v_first_name || ' ' || v_last_name ||
                     ', Department ID: ' || v_department_id);

END LOOP;

-- Close cursor
CLOSE emp_cur;
END;
/

```

Below the code, there's an 'Explanation' section with a checked checkbox:

- CURSOR emp_cur IS ... defines the cursor.
- OPEN emp_cur starts it.
- FETCH emp_cur INTO ... retrieves rows.
- EXIT WHEN emp_cur%NOTFOUND ensures loop ends when no more rows.
- DBMS_OUTPUT.PUT_LINE prints employee details.

At the bottom, there's a search bar with '+ Ask anything' and a note: 'ChatGPT can make mistakes. Check important info. See [Cookie Preferences](#)'.

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

The screenshot shows the ChatGPT interface with a dark theme. On the left, there's a sidebar with various chat options like 'New chat', 'Search chats', 'Library', and a list of recent chats including 'PL/SQL employee count', 'PPT pastel kawaii style', etc. The main area displays a block of PL/SQL code:

```

DECLARE
    -- Declare cursor
    CURSOR course_cur IS
        SELECT course_id, course_name
        FROM courses;

    -- Variables to hold fetched data
    v_course_id courses.course_id%TYPE;
    v_course_name courses.course_name%TYPE;

BEGIN
    -- Open cursor
    OPEN course_cur;

    LOOP
        -- Fetch one record into variables
        FETCH course_cur INTO v_course_id, v_course_name;

        -- Exit loop when no more rows
        EXIT WHEN course_cur%NOTFOUND;

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

```

At the bottom, there's a search bar with '+ Ask anything' and a note: 'ChatGPT can make mistakes. Check important info. See [Cookie Preferences](#)'.

40. Perform a transaction where you create a savepoint, insert records, then rollback to the savepoint.

```
BEGIN
    -- First insert
    INSERT INTO employees (employee_id, first_name, last_name, department_id)
    VALUES (3001, 'Alex', 'Johnson', 10);

    -- Create savepoint
    SAVEPOINT sp_insert;

    -- Second insert (this will be rolled back)
    INSERT INTO employees (employee_id, first_name, last_name, department_id)
    VALUES (3002, 'Maria', 'Lopez', 20);

    DBMS_OUTPUT.PUT_LINE('Two employees inserted. Now rolling back to savepoint...');

    -- Rollback to savepoint (removes Maria, keeps Alex)
    ROLLBACK TO sp_insert;

    DBMS_OUTPUT.PUT_LINE('Rollback completed. Only Alex remains inserted.');

    -- Commit the remaining changes
    COMMIT;
END;
/
```

41. Commit part of a transaction after using a savepoint and then rollback the remaining changes.

```
Here's a PL/SQL demo:

sql
BEGIN
    -- Insert first record
    INSERT INTO employees (employee_id, first_name, last_name, department_id)
    VALUES (4001, 'John', 'Smith', 10);

    -- Savepoint after first insert
    SAVEPOINT sp_first;

    -- Insert second record
    INSERT INTO employees (employee_id, first_name, last_name, department_id)
    VALUES (4002, 'Emma', 'Brown', 20);

    -- Commit everything up to this point (John + Emma become permanent)
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Committed John and Emma');

    -- Insert third record (not committed yet)
    INSERT INTO employees (employee_id, first_name, last_name, department_id)
    VALUES (4003, 'Liam', 'Davis', 30);
```

-- Insert fourth record (not committed yet)
INSERT INTO employees (employee_id, first_name, last_name, department_id)
VALUES (4084, 'Sophia', 'Wilson', 40);

DBMS_OUTPUT.PUT_LINE('Inserted Liam and Sophia (not committed yet).');

-- Rollback uncommitted changes (Liam + Sophia removed, John + Emma remain)
ROLLBACK;

DBMS_OUTPUT.PUT_LINE('Rolled back uncommitted changes. Only John and Emma remain in table.');
END;
/

Explanation
1. Insert John → Savepoint.
2. Insert Emma → `COMMIT` → John + Emma are permanent.
3. Insert Liam and Sophia → not committed yet.
4. `ROLLBACK` → removes Liam + Sophia but keeps John + Emma (since they were already committed).
⚠ Important: Once you `COMMIT`, you cannot roll back a savepoint from before it — only uncommitted changes can be rolled back.

+ Ask anything

ChatGPT can make mistakes. Check important info. See [Cookie Preferences](#).