SQL LAB-4
RDBMS, MYSQL

NAME: Sai Sujan C
ID: AF0366848

QUESTIONS

Lab 1: Database Schema:

Consider a simple database with one tables: BankAccount

BankAccount Table:

● Columns: account_id (Primary Key), account_holder_name, account_balance

Task 1: Insert Data

Write an SQL INSERT statement to insert data into the BankAccount table.

Task 2: Retrieving Data

Write an SQL SELECT statement to retrieve the account_holder_name and account_balance of all account holders from the BankAccount table.

Task 3: Filtering Data

Write an SQL SELECT statement to retrieve the account_holder_name and account_balance where the account_balance is more than 30,000.

Task 4: Updating Data

Write an SQL UPDATE statement to change the account_balance of the account holder whose ID is 101.

ChatGPT Exercise

Using ChatGPT generates SQL queries of the below problem .

Scenario 1: In an employee database, you want to retrieve information about employees who belong to the "Sales" department and have a salary greater than 50,000.

Scenario 2: An employee has resigned, and you need to remove their record from the "employees" table. Write an SQL DELETE query for this.

Scenario 3: You want to delete all orders placed before '2022-01-01' that are still in the 'Pending' status. Write an SQL DELETE query for this.

Scenario 4: You want to remove all products from the "Discontinued" category as they are no longer available. Write an SQL DELETE query for this.

Scenario 5: Employees in the "Sales" department are getting a bonus, and you want to add 1000 to the bonus column for all employees in that department. Write an SQL

UPDATE query for this

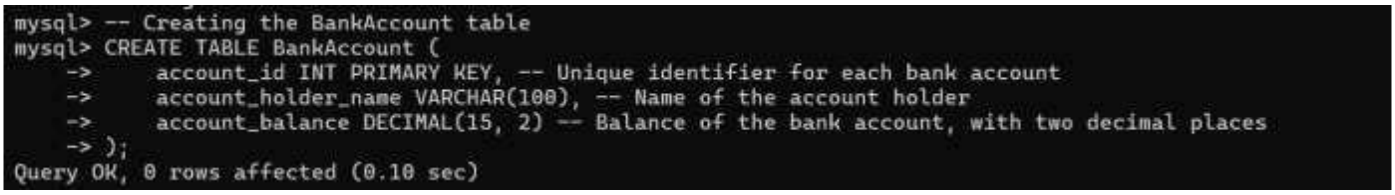
Lab 1: Database Schema:

Consider a simple database with one tables: BankAccount

BankAccount Table:

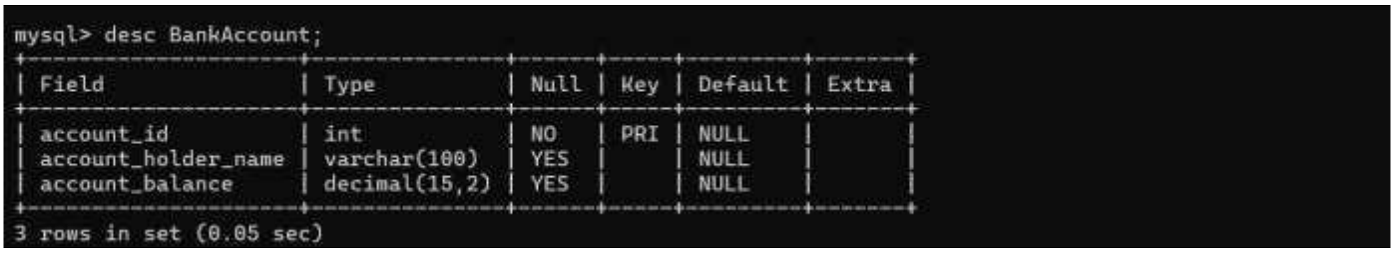
● Columns: account_id (Primary Key), account_holder_name, account_balance

Code:-



```
mysql> -- Creating the BankAccount table
mysql> CREATE TABLE BankAccount (
    ->     account_id INT PRIMARY KEY, -- Unique identifier for each bank account
    ->     account_holder_name VARCHAR(100), -- Name of the account holder
    ->     account_balance DECIMAL(15, 2) -- Balance of the bank account, with two decimal places
    -> );
Query OK, 0 rows affected (0.10 sec)
```

Output:-

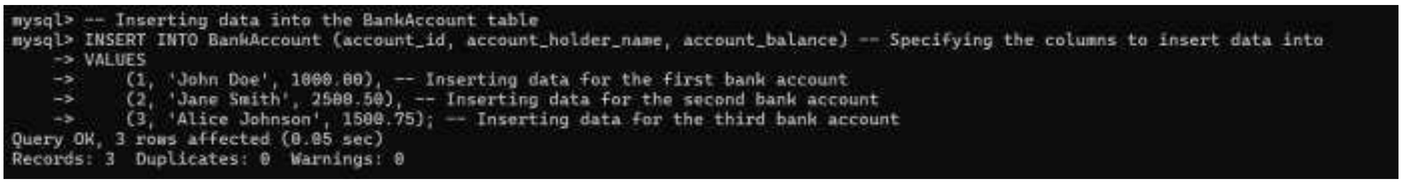


```
mysql> desc BankAccount;
+---------------------+---------------+------+-----+---------+-------+
| Field               | Type          | Null | Key | Default | Extra |
+---------------------+---------------+------+-----+---------+-------+
| account_id          | int           | NO   | PRI | NULL    |       |
| account_holder_name | varchar(100)  | YES  |     | NULL    |       |
| account_balance     | decimal(15,2) | YES  |     | NULL    |       |
+---------------------+---------------+------+-----+---------+-------+
3 rows in set (0.05 sec)
```

Task 1: Insert Data

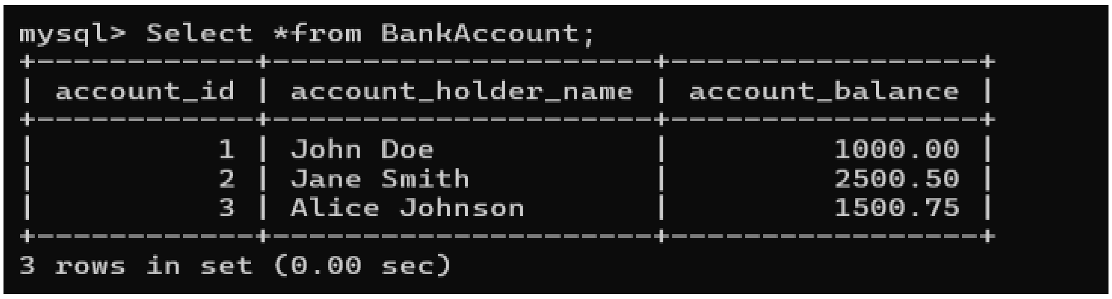
Write an SQL INSERT statement to insert data into the BankAccount table.

Code:-



```
mysql> -- Inserting data into the BankAccount table
mysql> INSERT INTO BankAccount (account_id, account_holder_name, account_balance) -- Specifying the columns to insert data into
    -> VALUES
    ->     (1, 'John Doe', 1000.00), -- Inserting data for the first bank account
    ->     (2, 'Jane Smith', 2500.50), -- Inserting data for the second bank account
    ->     (3, 'Alice Johnson', 1500.75); -- Inserting data for the third bank account
Query OK, 3 rows affected (0.05 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

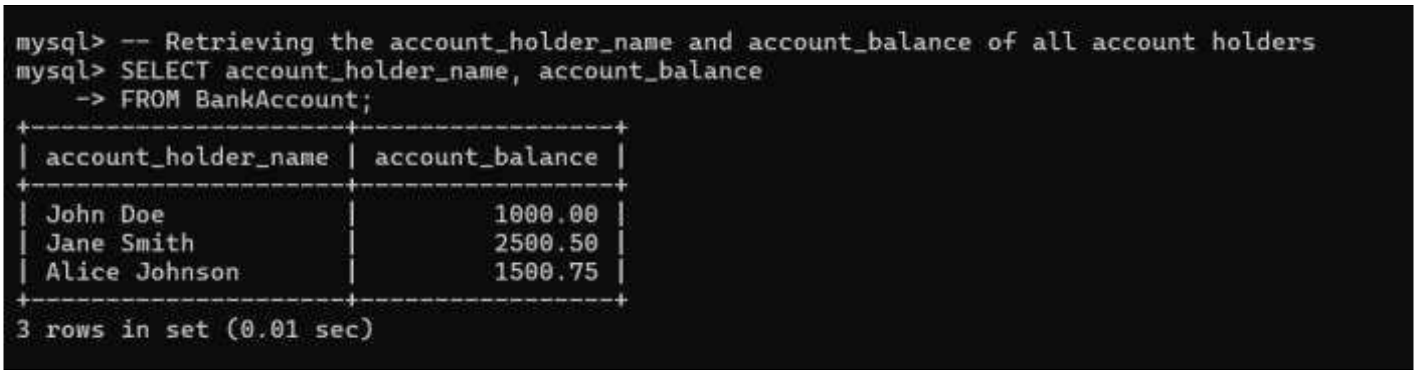
Output:-



```
mysql> Select *from BankAccount;
+------------+---------------------+-----------------+
| account_id | account_holder_name | account_balance |
+------------+---------------------+-----------------+
|          1 | John Doe            |         1000.00 |
|          2 | Jane Smith          |         2500.50 |
|          3 | Alice Johnson       |         1500.75 |
+------------+---------------------+-----------------+
3 rows in set (0.00 sec)
```

Task 2: Retrieving Data

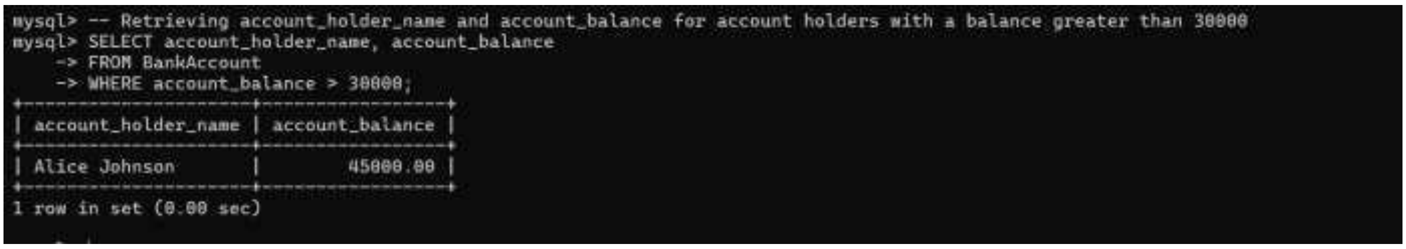
Write an SQL SELECT statement to retrieve the account_holder_name and account_balance of all account holders from the BankAccount table.



```
mysql> -- Retrieving the account_holder_name and account_balance of all account holders
mysql> SELECT account_holder_name, account_balance
    -> FROM BankAccount;
+---------------------+-----------------+
| account_holder_name | account_balance |
+---------------------+-----------------+
| John Doe            |         1000.00 |
| Jane Smith          |         2500.50 |
| Alice Johnson       |         1500.75 |
+---------------------+-----------------+
3 rows in set (0.01 sec)
```

Task 3: Filtering Data

Write an SQL SELECT statement to retrieve the account_holder_name and account_balance where the account_balance is more than 30,000.



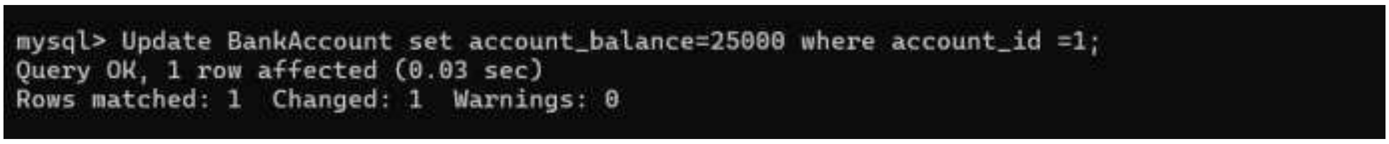
```
mysql> -- Retrieving account_holder_name and account_balance for account holders with a balance greater than 30000
mysql> SELECT account_holder_name, account_balance
    -> FROM BankAccount
    -> WHERE account_balance > 30000;
+---------------------+-----------------+
| account_holder_name | account_balance |
+---------------------+-----------------+
| Alice Johnson       |        45000.00 |
+---------------------+-----------------+
1 row in set (0.00 sec)
```

Task 4: Updating Data

Write an SQL UPDATE statement to change the account_balance of the account holder whose ID is 1.

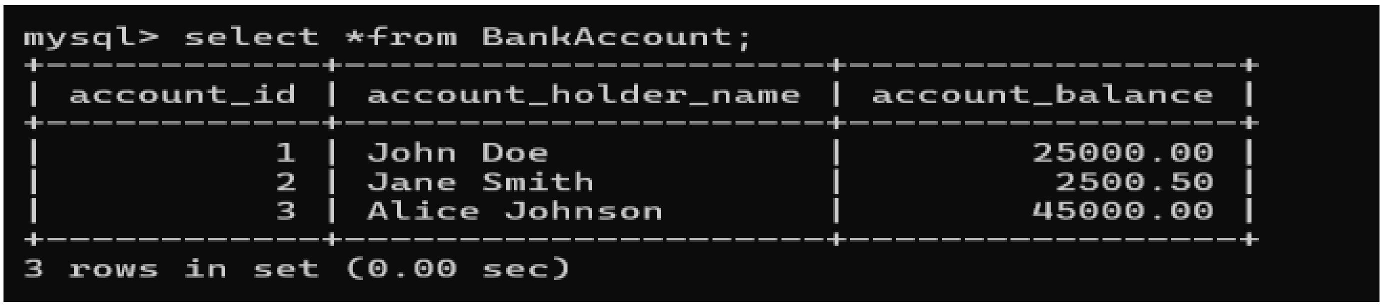
Code:-

-- Updating the account_balance to 25000 for the bank account with account_id = 1



```
mysql> Update BankAccount set account_balance=25000 where account_id =1;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Output:-

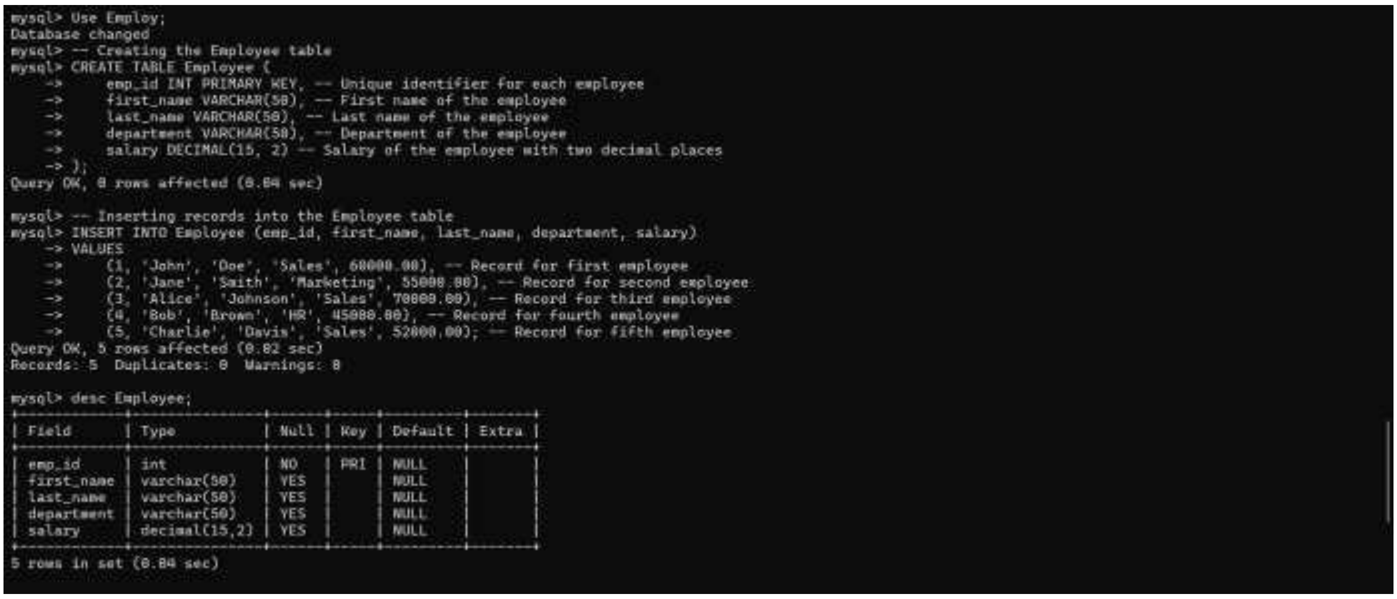


```
mysql> select *from BankAccount;
+------------+---------------------+-----------------+
| account_id | account_holder_name | account_balance |
+------------+---------------------+-----------------+
|          1 | John Doe            |        25000.00 |
|          2 | Jane Smith          |         2500.50 |
|          3 | Alice Johnson       |        45000.00 |
+------------+---------------------+-----------------+
3 rows in set (0.00 sec)
```

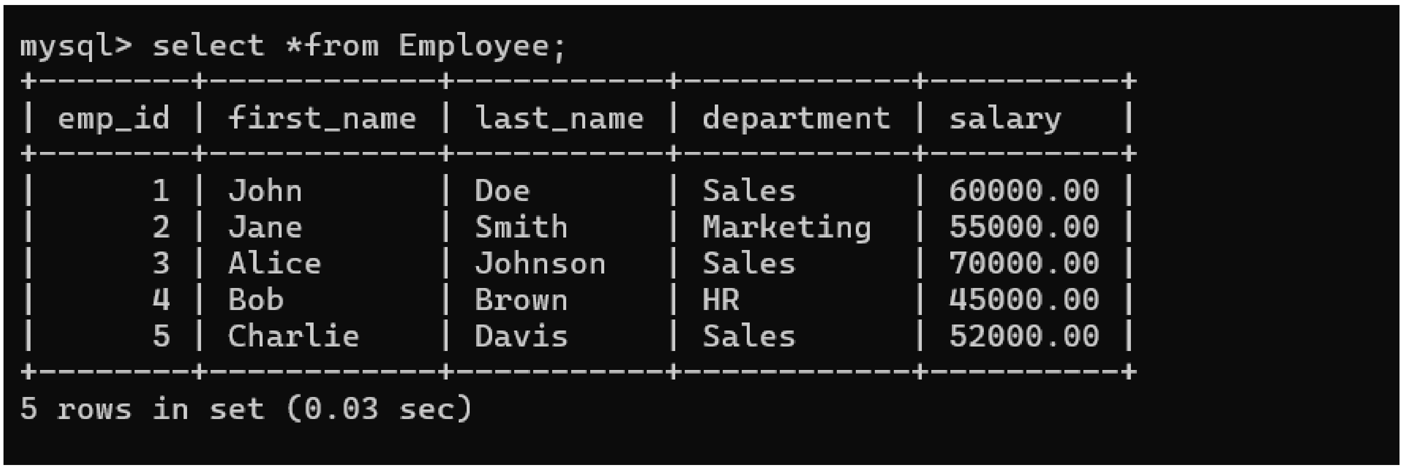
ChatGPT Exercise

Using ChatGPT generates SQL queries of the below problem .

Scenario 1: In an employee database, you want to retrieve information about employees who belong to the "Sales" department and have a salary greater than 50,000.

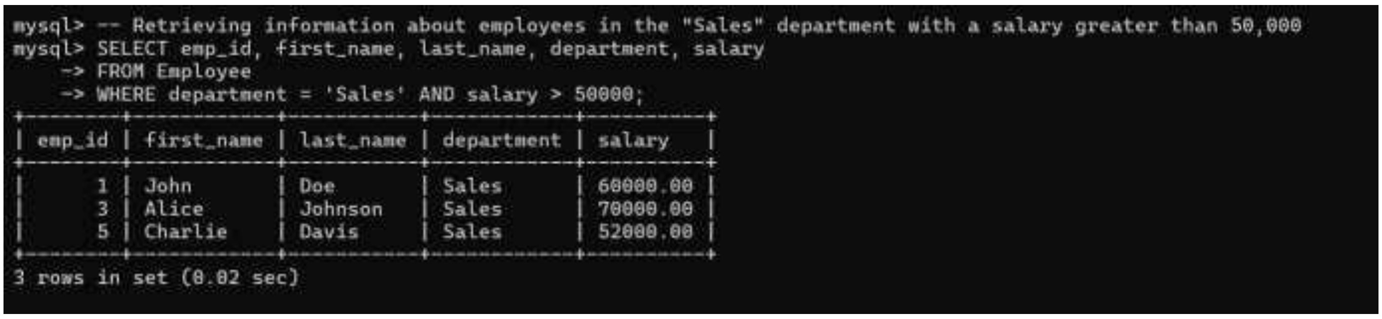


```
mysql> Use Employ;
Database changed
mysql> -- Creating the Employee table
mysql> CREATE TABLE Employee (
    ->     emp_id INT PRIMARY KEY, -- Unique identifier for each employee
    ->     first_name VARCHAR(50), -- First name of the employee
    ->     last_name VARCHAR(50), -- Last name of the employee
    ->     department VARCHAR(50), -- Department of the employee
    ->     salary DECIMAL(15, 2) -- Salary of the employee with two decimal places
    -> );
Query OK, 0 rows affected (0.04 sec)

mysql> -- Inserting records into the Employee table
mysql> INSERT INTO Employee (emp_id, first_name, last_name, department, salary)
    -> VALUES
    ->     (1, 'John', 'Doe', 'Sales', 60000.00), -- Record for first employee
    ->     (2, 'Jane', 'Smith', 'Marketing', 55000.00), -- Record for second employee
    ->     (3, 'Alice', 'Johnson', 'Sales', 70000.00), -- Record for third employee
    ->     (4, 'Bob', 'Brown', 'HR', 45000.00), -- Record for fourth employee
    ->     (5, 'Charlie', 'Davis', 'Sales', 52000.00); -- Record for fifth employee
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> desc Employee;
+------------+---------------+------+-----+---------+-------+
| Field      | Type          | Null | Key | Default | Extra |
+------------+---------------+------+-----+---------+-------+
| emp_id     | int           | NO   | PRI | NULL    |       |
| first_name | varchar(50)   | YES  |     | NULL    |       |
| last_name  | varchar(50)   | YES  |     | NULL    |       |
| department | varchar(50)   | YES  |     | NULL    |       |
| salary     | decimal(15,2) | YES  |     | NULL    |       |
+------------+---------------+------+-----+---------+-------+
5 rows in set (0.04 sec)
```



```
mysql> select *from Employee;
+--------+------------+-----------+------------+----------+
| emp_id | first_name | last_name | department | salary   |
+--------+------------+-----------+------------+----------+
|      1 | John       | Doe       | Sales      | 60000.00 |
|      2 | Jane       | Smith     | Marketing  | 55000.00 |
|      3 | Alice      | Johnson   | Sales      | 70000.00 |
|      4 | Bob        | Brown     | HR         | 45000.00 |
|      5 | Charlie    | Davis     | Sales      | 52000.00 |
+--------+------------+-----------+------------+----------+
5 rows in set (0.03 sec)
```

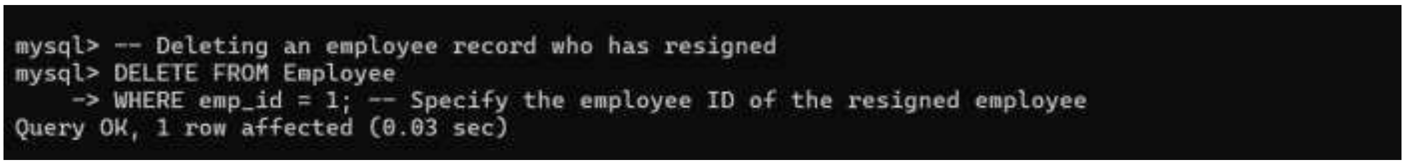
Code & output:-



```
mysql> -- Retrieving information about employees in the "Sales" department with a salary greater than 50,000
mysql> SELECT emp_id, first_name, last_name, department, salary
    -> FROM Employee
    -> WHERE department = 'Sales' AND salary > 50000;
+--------+------------+-----------+------------+----------+
| emp_id | first_name | last_name | department | salary   |
+--------+------------+-----------+------------+----------+
|      1 | John       | Doe       | Sales      | 60000.00 |
|      3 | Alice      | Johnson   | Sales      | 70000.00 |
|      5 | Charlie    | Davis     | Sales      | 52000.00 |
+--------+------------+-----------+------------+----------+
3 rows in set (0.02 sec)
```

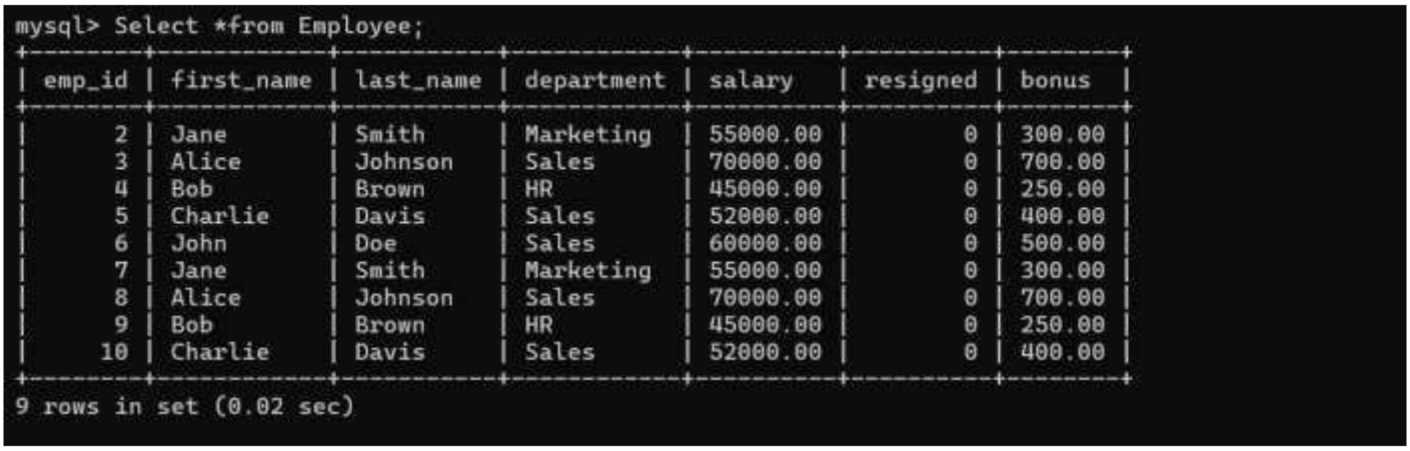
Scenario 2: An employee has resigned, and you need to remove their record from the "employees" table. Write an SQL DELETE query for this.

Code:-



```
mysql> -- Deleting an employee record who has resigned
mysql> DELETE FROM Employee
    -> WHERE emp_id = 1; -- Specify the employee ID of the resigned employee
Query OK, 1 row affected (0.03 sec)
```

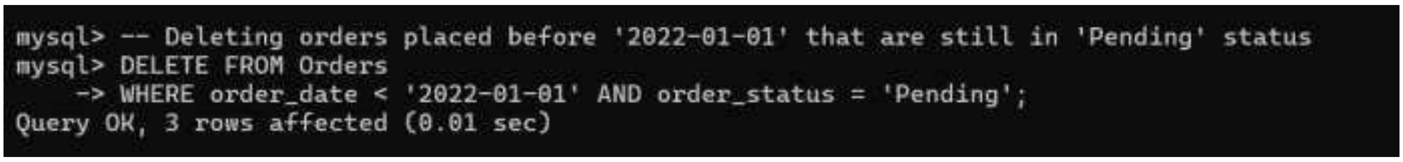
Output:-



```
mysql> Select *from Employee;
+--------+------------+-----------+------------+----------+----------+--------+
| emp_id | first_name | last_name | department | salary   | resigned | bonus  |
+--------+------------+-----------+------------+----------+----------+--------+
|      2 | Jane       | Smith     | Marketing  | 55000.00 |        0 | 300.00 |
|      3 | Alice      | Johnson   | Sales      | 70000.00 |        0 | 700.00 |
|      4 | Bob        | Brown     | HR         | 45000.00 |        0 | 250.00 |
|      5 | Charlie    | Davis     | Sales      | 52000.00 |        0 | 400.00 |
|      6 | John       | Doe       | Sales      | 60000.00 |        0 | 500.00 |
|      7 | Jane       | Smith     | Marketing  | 55000.00 |        0 | 300.00 |
|      8 | Alice      | Johnson   | Sales      | 70000.00 |        0 | 700.00 |
|      9 | Bob        | Brown     | HR         | 45000.00 |        0 | 250.00 |
|     10 | Charlie    | Davis     | Sales      | 52000.00 |        0 | 400.00 |
+--------+------------+-----------+------------+----------+----------+--------+
9 rows in set (0.02 sec)
```

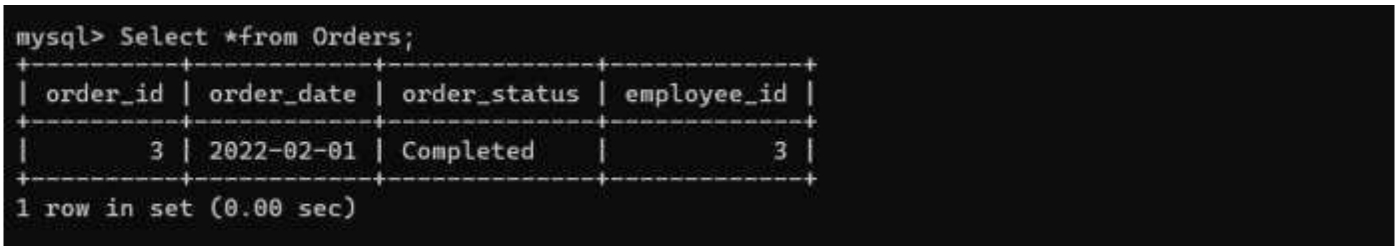
Scenario 3: You want to delete all orders placed before '2022-01-01' that are still in the 'Pending' status. Write an SQL DELETE query for this.

Code:-



```
mysql> -- Deleting orders placed before '2022-01-01' that are still in 'Pending' status
mysql> DELETE FROM Orders
    -> WHERE order_date < '2022-01-01' AND order_status = 'Pending';
Query OK, 3 rows affected (0.01 sec)
```

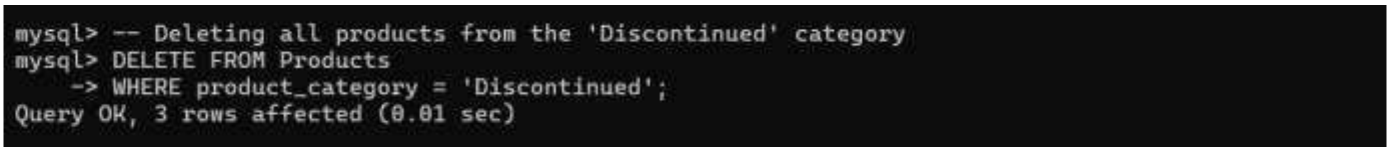
Output:-



```
mysql> Select *from Orders;
+----------+------------+--------------+-------------+
| order_id | order_date | order_status | employee_id |
+----------+------------+--------------+-------------+
|        3 | 2022-02-01 | Completed    |           3 |
+----------+------------+--------------+-------------+
1 row in set (0.00 sec)
```

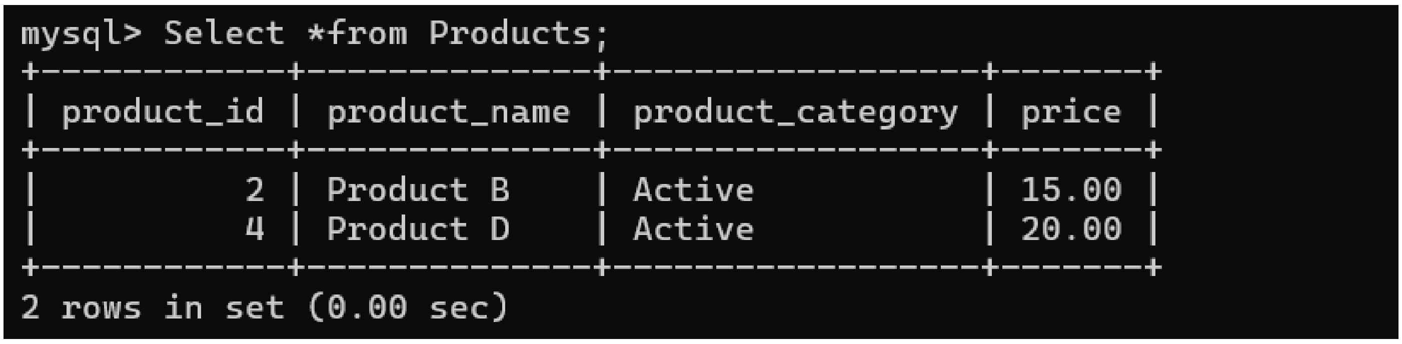
Scenario 4: You want to remove all products from the "Discontinued" category as they are no longer available. Write an SQL DELETE query for this.

Code:-



```
mysql> -- Deleting all products from the 'Discontinued' category
mysql> DELETE FROM Products
    -> WHERE product_category = 'Discontinued';
Query OK, 3 rows affected (0.01 sec)
```

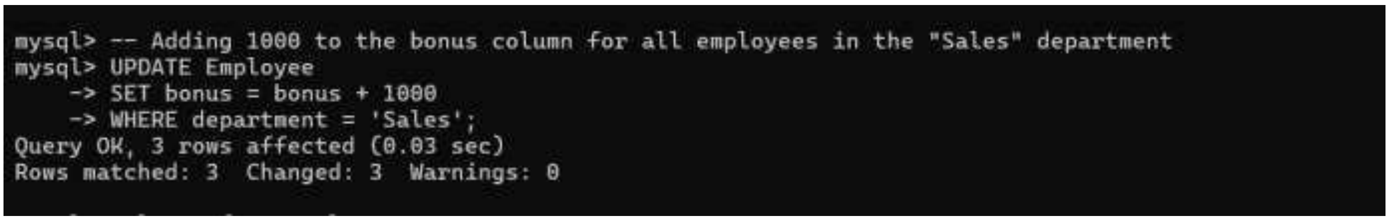
Output:-



```
mysql> Select *from Products;
+------------+--------------+------------------+-------+
| product_id | product_name | product_category | price |
+------------+--------------+------------------+-------+
|          2 | Product B    | Active           | 15.00 |
|          4 | Product D    | Active           | 20.00 |
+------------+--------------+------------------+-------+
2 rows in set (0.00 sec)
```

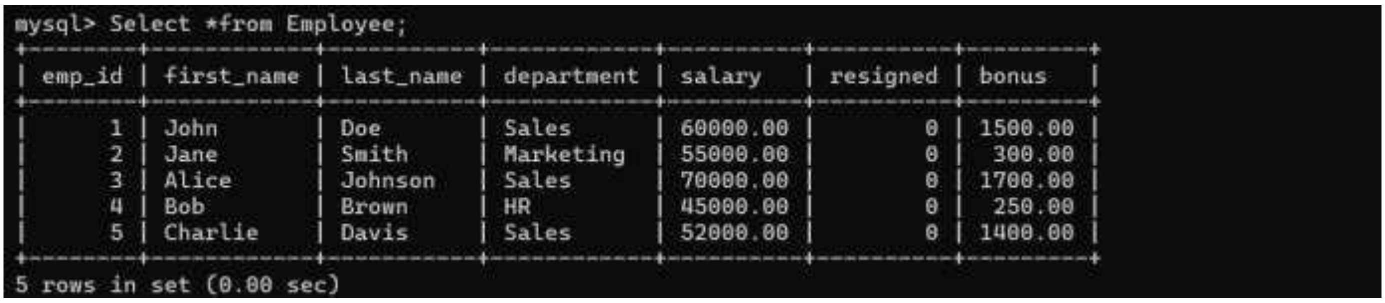
Scenario 5: Employees in the "Sales" department are getting a bonus, and you want to add 1000 to the bonus column for all employees in that department. Write an SQL UPDATE query for this

Code:-



```
mysql> -- Adding 1000 to the bonus column for all employees in the "Sales" department
mysql> UPDATE Employee
    -> SET bonus = bonus + 1000
    -> WHERE department = 'Sales';
Query OK, 3 rows affected (0.03 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

Output:-



```
mysql> Select *from Employee;
+--------+------------+-----------+------------+----------+----------+---------+
| emp_id | first_name | last_name | department | salary   | resigned | bonus   |
+--------+------------+-----------+------------+----------+----------+---------+
|      1 | John       | Doe       | Sales      | 60000.00 |        0 | 1500.00 |
|      2 | Jane       | Smith     | Marketing  | 55000.00 |        0 |  300.00 |
|      3 | Alice      | Johnson   | Sales      | 70000.00 |        0 | 1700.00 |
|      4 | Bob        | Brown     | HR         | 45000.00 |        0 |  250.00 |
|      5 | Charlie    | Davis     | Sales      | 52000.00 |        0 | 1400.00 |
+--------+------------+-----------+------------+----------+----------+---------+
5 rows in set (0.00 sec)
```