LAbSQL4_Anp_c7281_Where_distinct

By

Pedda Jagadeesh

AF0366969

LAbSQL4_Anp_c7281_Where_distinct

Lab

(PN: ChatGPT exercise is mandatory)

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.

Submission:

Create an SQL script file containing your solutions for all tasks (queries). Name the file

"lab_assignment1.sql" Provide comments above each query to indicate the task

number and the query's purpose.

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

Sol:-

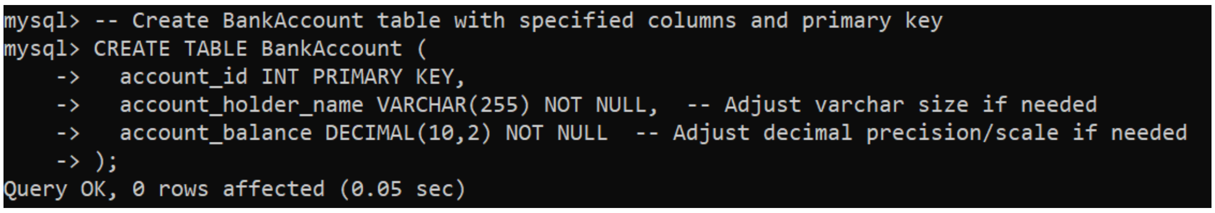
## lab_assignment1.sql

Table Creation

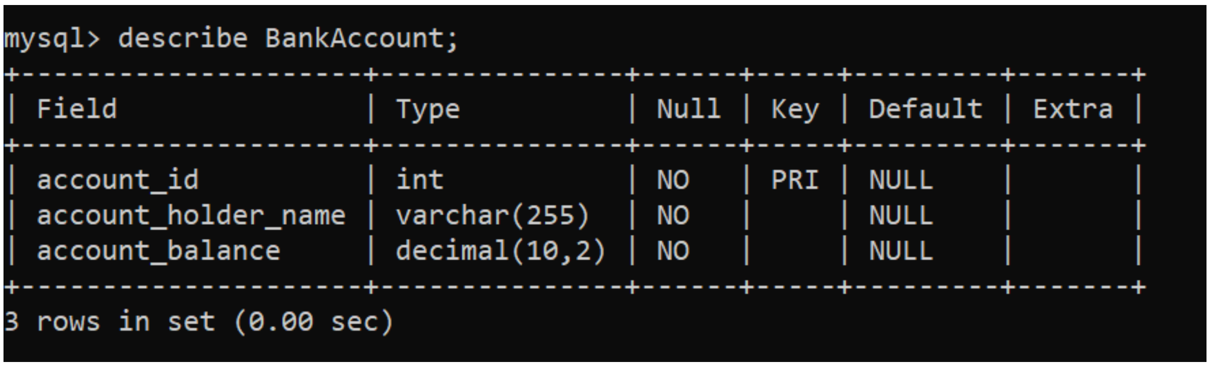
```SQL
-- Create BankAccount table with specified columns and primary key
CREATE TABLE BankAccount (
  account_id INT PRIMARY KEY,
  account_holder_name VARCHAR(255) NOT NULL,  -- Adjust varchar size if needed
```

 account_balance DECIMAL(10,2) NOT NULL -- Adjust decimal precision/scale if needed

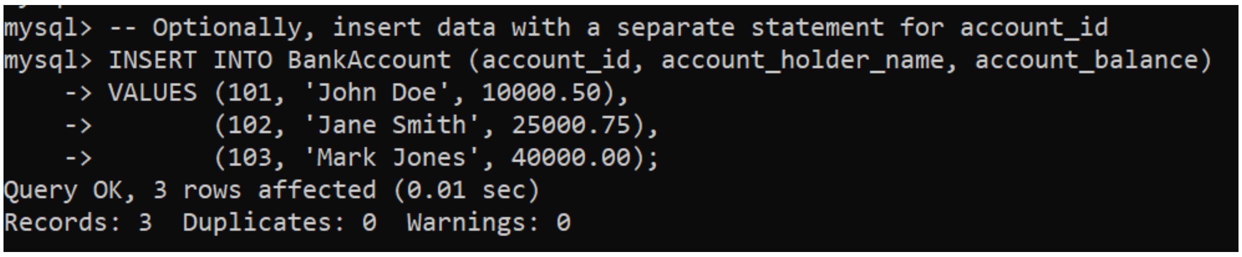
);

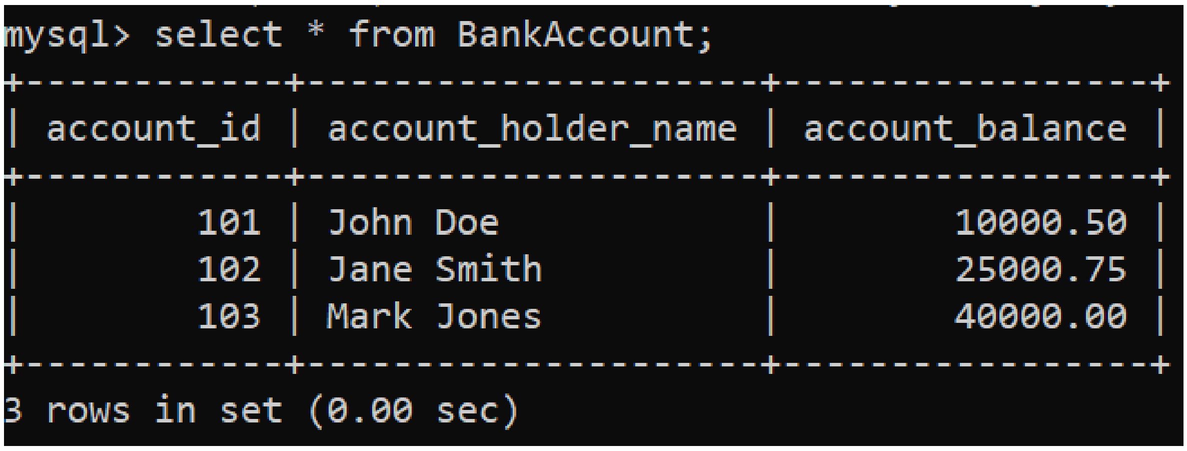


```
mysql> -- Create BankAccount table with specified columns and primary key
mysql> CREATE TABLE BankAccount (
    ->   account_id INT PRIMARY KEY,
    ->   account_holder_name VARCHAR(255) NOT NULL,   -- Adjust varchar size if needed
    ->   account_balance DECIMAL(10,2) NOT NULL  -- Adjust decimal precision/scale if needed
    -> );
Query OK, 0 rows affected (0.05 sec)
```

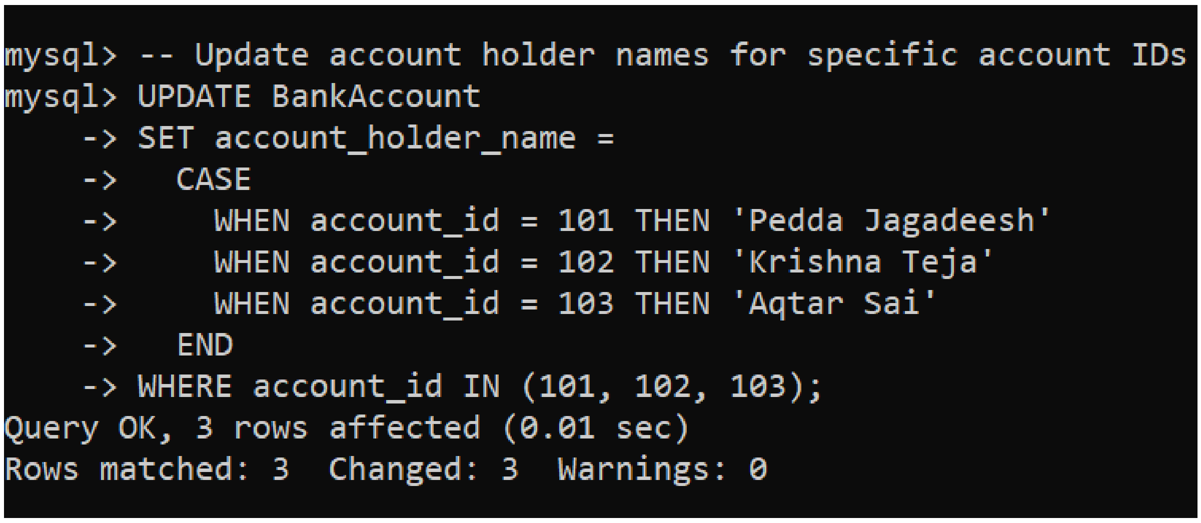


```
mysql> describe BankAccount;
+---------------------+---------------+------+-----+---------+-------+
| Field               | Type          | Null | Key | Default | Extra |
+---------------------+---------------+------+-----+---------+-------+
| account_id          | int           | NO   | PRI | NULL    |       |
| account_holder_name | varchar(255)  | NO   |     | NULL    |       |
| account_balance     | decimal(10,2) | NO   |     | NULL    |       |
+---------------------+---------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

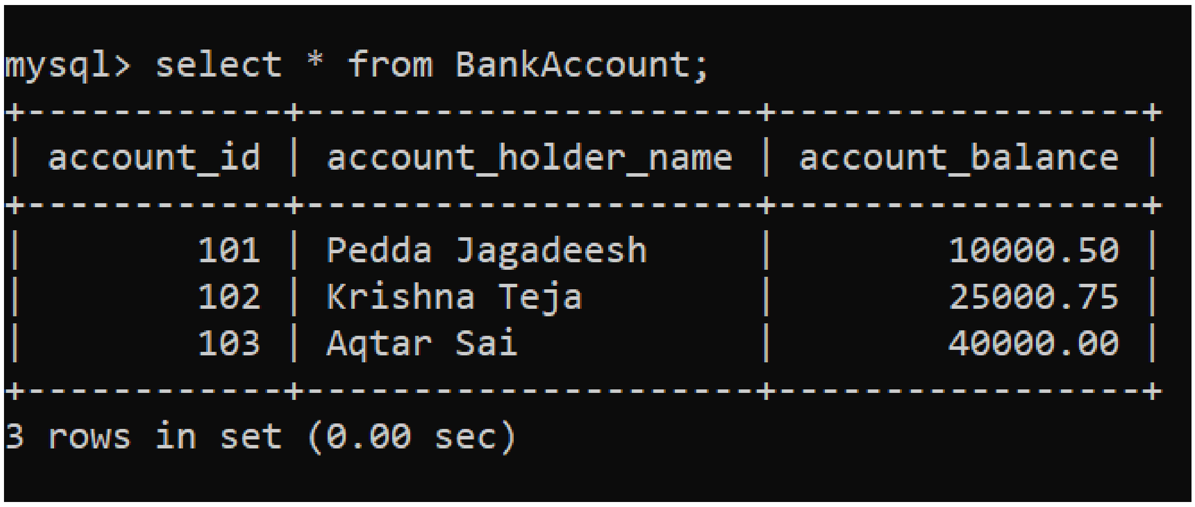
~~~

Task 1: Inserting Data

```SQL

Code:

-- Insert sample data into BankAccount table (same as before)

INSERT INTO BankAccount (account_holder_name, account_balance)

VALUES ('John Doe', 10000.50),

       ('Jane Smith', 25000.75),

       ('Mark Jones', 40000.00);



~~~
mysql> -- Optionally, insert data with a separate statement for account_id
mysql> INSERT INTO BankAccount (account_id, account_holder_name, account_balance)
    -> VALUES (101, 'John Doe', 10000.50),
    ->        (102, 'Jane Smith', 25000.75),
    ->        (103, 'Mark Jones', 40000.00);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```
```



```
mysql> select * from BankAccount;
+------------+---------------------+-----------------+
| account_id | account_holder_name | account_balance |
+------------+---------------------+-----------------+
|        101 | John Doe            |        10000.50 |
|        102 | Jane Smith          |        25000.75 |
|        103 | Mark Jones          |        40000.00 |
+------------+---------------------+-----------------+
3 rows in set (0.00 sec)
```



```
mysql> -- Update account holder names for specific account IDs
mysql> UPDATE BankAccount
    -> SET account_holder_name =
    ->   CASE
    ->     WHEN account_id = 101 THEN 'Pedda Jagadeesh'
    ->     WHEN account_id = 102 THEN 'Krishna Teja'
    ->     WHEN account_id = 103 THEN 'Aqtar Sai'
    ->   END
    -> WHERE account_id IN (101, 102, 103);
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```



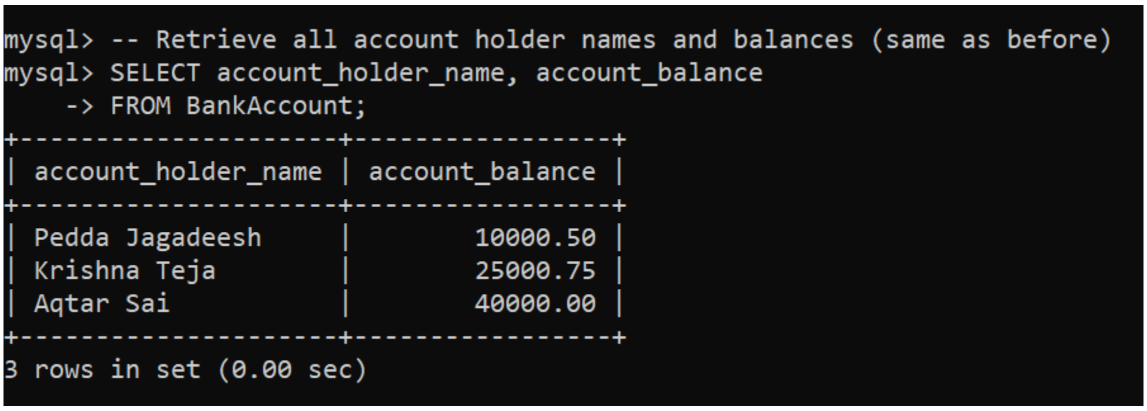
```
mysql> select * from BankAccount;
+------------+---------------------+-----------------+
| account_id | account_holder_name | account_balance |
+------------+---------------------+-----------------+
|        101 | Pedda Jagadeesh     |        10000.50 |
|        102 | Krishna Teja        |        25000.75 |
|        103 | Aqtar Sai           |        40000.00 |
+------------+---------------------+-----------------+
3 rows in set (0.00 sec)
```

~~~

Task 2: Retrieving All Data

```SQL
~~~

-- Retrieve all account holder names and balances (same as before)

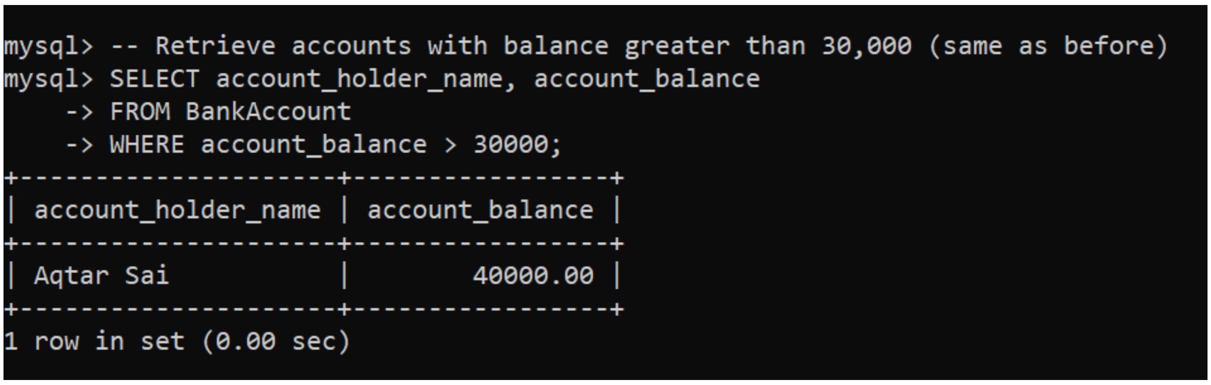
SELECT account_holder_name, account_balance

FROM BankAccount;

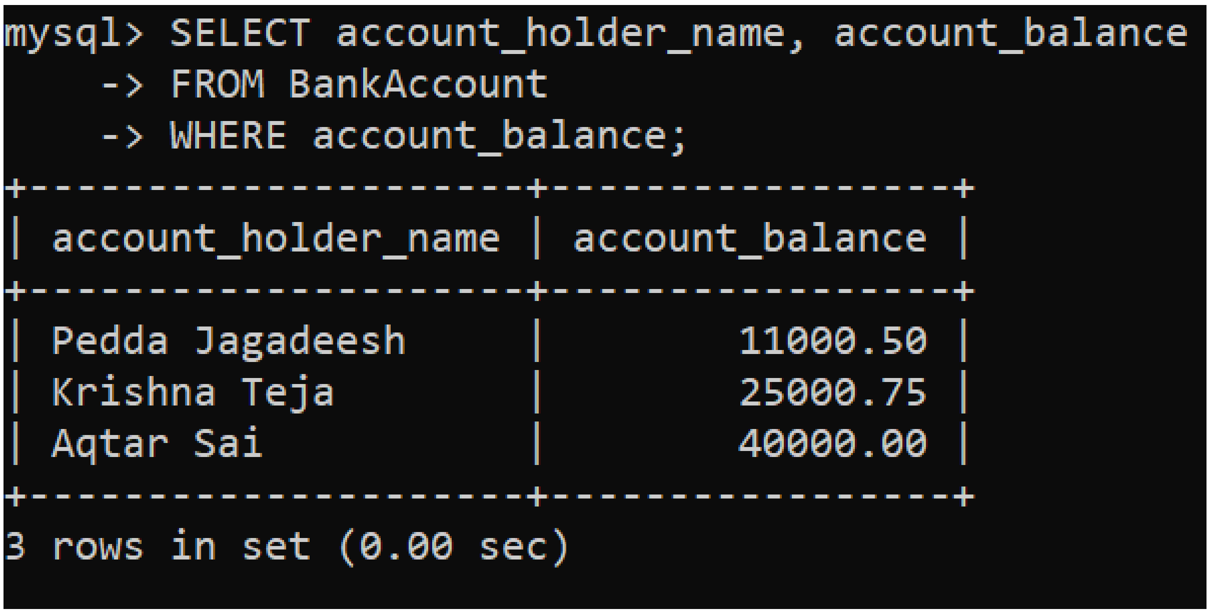


```
mysql> -- Retrieve all account holder names and balances (same as before)
mysql> SELECT account_holder_name, account_balance
    -> FROM BankAccount;
+---------------------+-----------------+
| account_holder_name | account_balance |
+---------------------+-----------------+
| Pedda Jagadeesh     |        10000.50 |
| Krishna Teja        |        25000.75 |
| Aqtar Sai           |        40000.00 |
+---------------------+-----------------+
3 rows in set (0.00 sec)
```

Task 3: Filtering Data

```SQL

-- Retrieve accounts with balance greater than 30,000 (same as before)

SELECT account_holder_name, account_balance

FROM BankAccount

WHERE account_balance > 30000;



```
mysql> -- Retrieve accounts with balance greater than 30,000 (same as before)
mysql> SELECT account_holder_name, account_balance
    -> FROM BankAccount
    -> WHERE account_balance > 30000;
+---------------------+-----------------+
| account_holder_name | account_balance |
+---------------------+-----------------+
| Aqtar Sai | 40000.00 |
+---------------------+-----------------+
1 row in set (0.00 sec)
```

```

Task 4: Updating Data

```SQL

-- Update account balance for account ID 101 (same as before)

UPDATE BankAccount

SET account_balance = account_balance + 1000.00

WHERE account_id = 101;



```
mysql> SELECT account_holder_name, account_balance
    -> FROM BankAccount
    -> WHERE account_balance;
+---------------------+-----------------+
| account_holder_name | account_balance |
+---------------------+-----------------+
Pedda Jagadeesh	11000.50
Krishna Teja	25000.75
Aqtar Sai	40000.00
+---------------------+-----------------+
3 rows in set (0.00 sec)
```

```

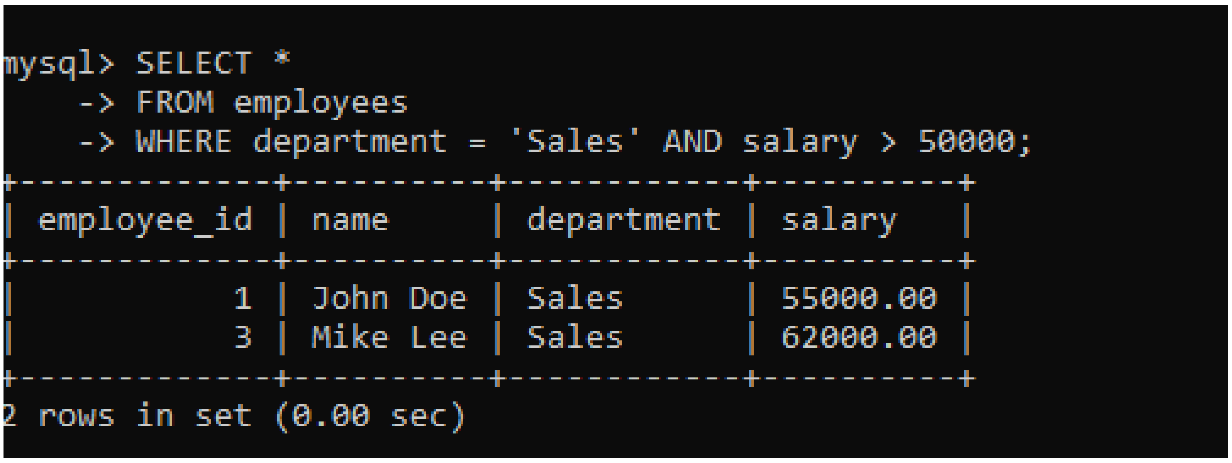
This script creates the `BankAccount` table with the specified columns and defines `account_id` as the primary key. It then proceeds with the tasks you requested from

Lab 1. Remember to adjust the data types (e.g., `VARCHAR` size, decimal precision/scale) if needed based on your specific requirements.

Scenario 1: In an employee database, you want to retrieve information about

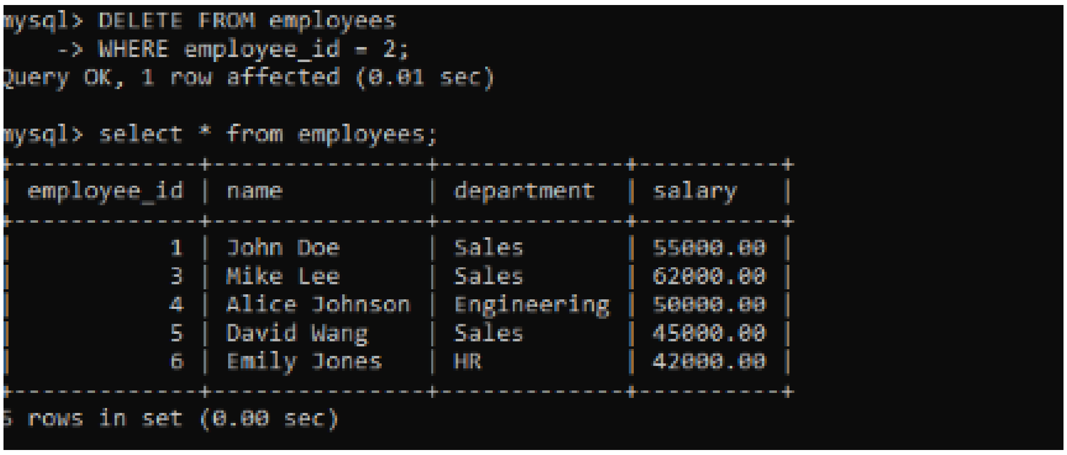
employees who belong to the "Sales" department and have a salary greater than

50,000.



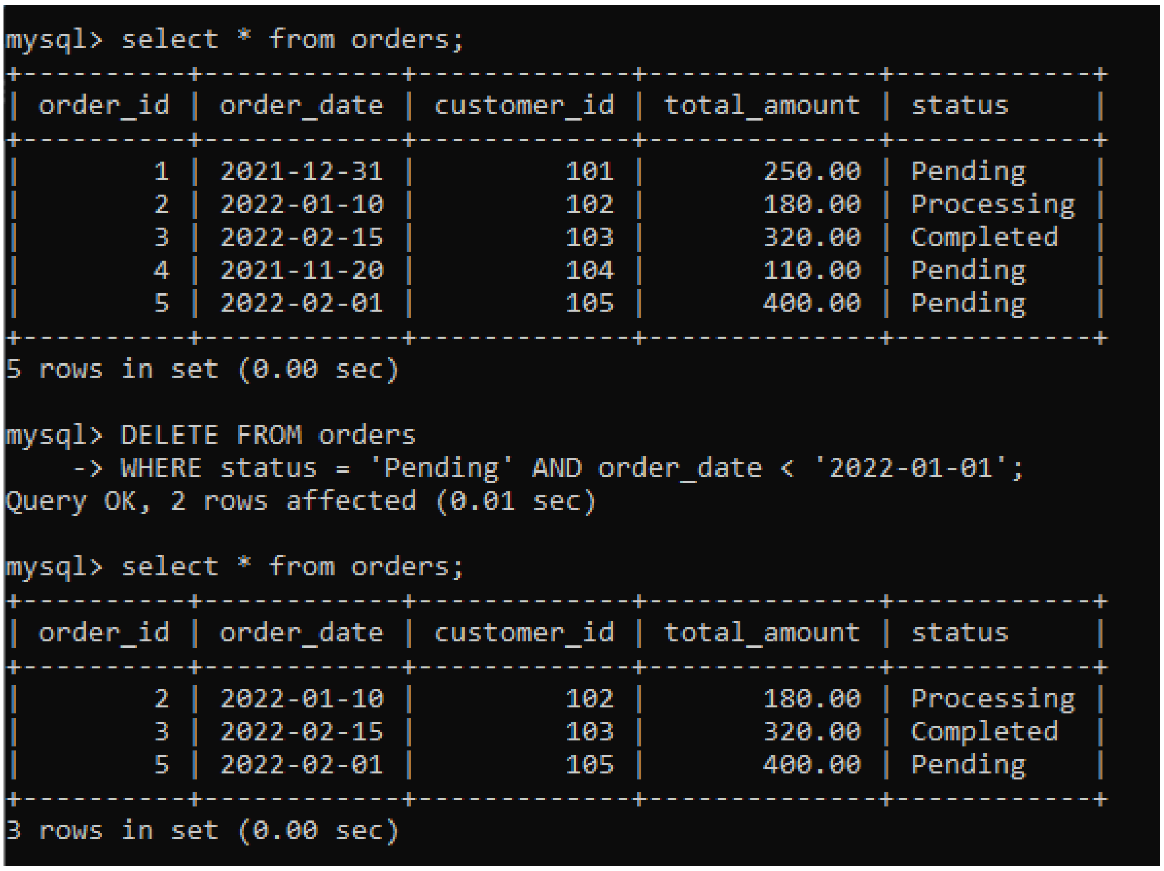
```
mysql> SELECT *
    -> FROM employees
    -> WHERE department = 'Sales' AND salary > 50000;
+-------------+----------+------------+----------+
| employee_id | name     | department | salary   |
+-------------+----------+------------+----------+
|           1 | John Doe | Sales      | 55000.00 |
|           3 | Mike Lee | Sales      | 62000.00 |
+-------------+----------+------------+----------+
2 rows in set (0.00 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.



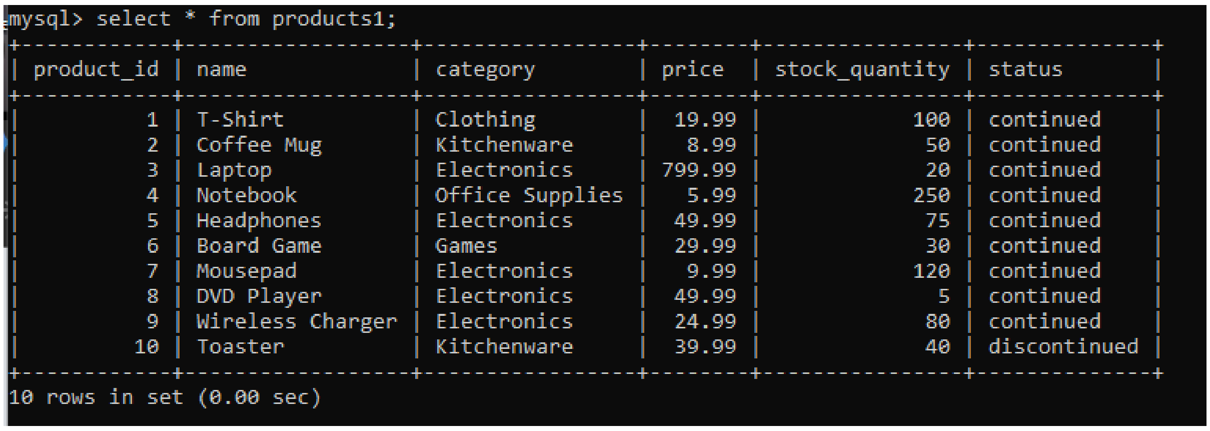
```
mysql> DELETE FROM employees
    -> WHERE employee_id = 2;
Query OK, 1 row affected (0.01 sec)

mysql> select * from employees;
+-------------+---------------+-------------+----------+
| employee_id | name          | department  | salary   |
+-------------+---------------+-------------+----------+
|           1 | John Doe      | Sales       | 55000.00 |
|           3 | Mike Lee      | Sales       | 62000.00 |
|           4 | Alice Johnson | Engineering | 50000.00 |
|           5 | David Wang    | Sales       | 45000.00 |
|           6 | Emily Jones   | HR          | 42000.00 |
+-------------+---------------+-------------+----------+
5 rows in set (0.00 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.

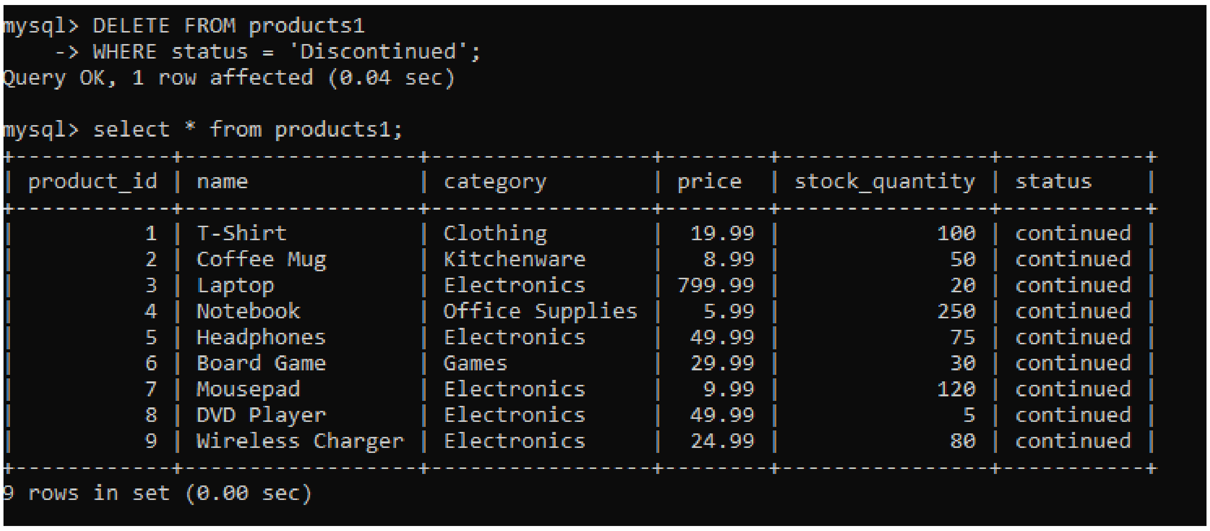


```
mysql> select * from orders;
+----------+------------+-------------+--------------+------------+
| order_id | order_date | customer_id | total_amount | status     |
+----------+------------+-------------+--------------+------------+
|        1 | 2021-12-31 |         101 |       250.00 | Pending    |
|        2 | 2022-01-10 |         102 |       180.00 | Processing |
|        3 | 2022-02-15 |         103 |       320.00 | Completed  |
|        4 | 2021-11-20 |         104 |       110.00 | Pending    |
|        5 | 2022-02-01 |         105 |       400.00 | Pending    |
+----------+------------+-------------+--------------+------------+
5 rows in set (0.00 sec)

mysql> DELETE FROM orders
    -> WHERE status = 'Pending' AND order_date < '2022-01-01';
Query OK, 2 rows affected (0.01 sec)

mysql> select * from orders;
+----------+------------+-------------+--------------+------------+
| order_id | order_date | customer_id | total_amount | status     |
+----------+------------+-------------+--------------+------------+
|        2 | 2022-01-10 |         102 |       180.00 | Processing |
|        3 | 2022-02-15 |         103 |       320.00 | Completed  |
|        5 | 2022-02-01 |         105 |       400.00 | Pending    |
+----------+------------+-------------+--------------+------------+
3 rows 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.



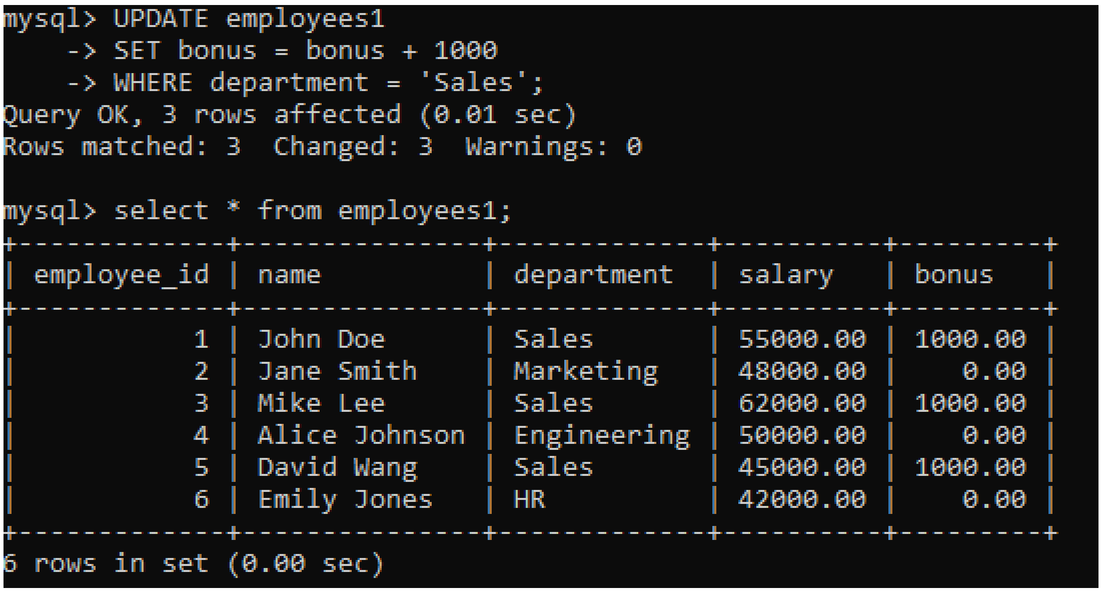
```
mysql> select * from products1;
+------------+------------------+-----------------+--------+----------------+--------------+
| product_id | name             | category        | price  | stock_quantity | status       |
+------------+------------------+-----------------+--------+----------------+--------------+
|          1 | T-Shirt          | Clothing        |  19.99 |            100 | continued    |
|          2 | Coffee Mug       | Kitchenware     |   8.99 |             50 | continued    |
|          3 | Laptop           | Electronics     | 799.99 |             20 | continued    |
|          4 | Notebook         | Office Supplies |   5.99 |            250 | continued    |
|          5 | Headphones       | Electronics     |  49.99 |             75 | continued    |
|          6 | Board Game       | Games           |  29.99 |             30 | continued    |
|          7 | Mousepad         | Electronics     |   9.99 |            120 | continued    |
|          8 | DVD Player       | Electronics     |  49.99 |              5 | continued    |
|          9 | Wireless Charger | Electronics     |  24.99 |             80 | continued    |
|         10 | Toaster          | Kitchenware     |  39.99 |             40 | discontinued |
+------------+------------------+-----------------+--------+----------------+--------------+
10 rows in set (0.00 sec)
```



```
mysql> DELETE FROM products1
    -> WHERE status = 'Discontinued';
Query OK, 1 row affected (0.04 sec)

mysql> select * from products1;
+------------+------------------+-----------------+--------+----------------+-----------+
| product_id | name             | category        | price  | stock_quantity | status    |
+------------+------------------+-----------------+--------+----------------+-----------+
|          1 | T-Shirt          | Clothing        |  19.99 |            100 | continued |
|          2 | Coffee Mug       | Kitchenware     |   8.99 |             50 | continued |
|          3 | Laptop           | Electronics     | 799.99 |             20 | continued |
|          4 | Notebook         | Office Supplies |   5.99 |            250 | continued |
|          5 | Headphones       | Electronics     |  49.99 |             75 | continued |
|          6 | Board Game       | Games           |  29.99 |             30 | continued |
|          7 | Mousepad         | Electronics     |   9.99 |            120 | continued |
|          8 | DVD Player       | Electronics     |  49.99 |              5 | continued |
|          9 | Wireless Charger | Electronics     |  24.99 |             80 | continued |
+------------+------------------+-----------------+--------+----------------+-----------+
9 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



```
mysql> UPDATE employees1
    -> SET bonus = bonus + 1000
    -> WHERE department = 'Sales';
Query OK, 3 rows affected (0.01 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> select * from employees1;
+-------------+---------------+-------------+----------+---------+
| employee_id | name          | department  | salary   | bonus   |
+-------------+---------------+-------------+----------+---------+
|           1 | John Doe      | Sales       | 55000.00 | 1000.00 |
|           2 | Jane Smith    | Marketing   | 48000.00 |    0.00 |
|           3 | Mike Lee      | Sales       | 62000.00 | 1000.00 |
|           4 | Alice Johnson | Engineering | 50000.00 |    0.00 |
|           5 | David Wang    | Sales       | 45000.00 | 1000.00 |
|           6 | Emily Jones   | HR          | 42000.00 |    0.00 |
+-------------+---------------+-------------+----------+---------+
6 rows in set (0.00 sec)
```

UPDATE query for this