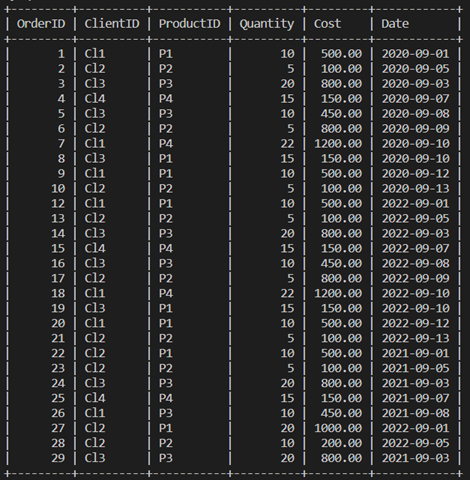
In the ungraded lab, you completed a series of tasks in which you helped Lucky Shrub improve the performance of their SELECT queries to make it easier and faster for them to retrieve information on their employees and client orders.

Now it’s time to review your understanding of the tasks you completed by answering the questions that follow. A screenshot of the Orders table is included for reference.



### 1.

Question 1

In the first task, you optimized the following SELECT query to extract data from the Orders table:

SELECT \* FROM Orders;

What action did you take to optimize this query?

1 / 1 point

You added an index to the ProductID column.

You reduced the number of rows by adding filter criteria.

You rewrote the query to avoid the use of \*.

You added an index to the OrderID column.

Correct

Correct! You avoided bringing unnecessary columns and specified only the required columns in the SELECT clause.

### 2.

Question 2

In the second task, you helped Lucky Shrub retrieve the order placed by the client with the ID of Cl1. You performed this task by creating an index on the Orders table to help optimize the following query:

SELECT \* FROM Orders WHERE ClientID='Cl1';

On which column in the Orders table did you create the index?

1 / 1 point

Quantity column

ClientID column

ProductID column

OrderID column

Correct

Correct! You created the index on the ClientID column of the Orders table.

3.

Question 3

In the second task, you reviewed a query EXPLAIN plan before optimizing the query. The plan indicated NULL values in the possible\_keys and keys columns. This suggests that there is no index in the targeted table which can be used to perform the search.

1 / 1 point

True

False

Correct

Correct! NULL values in these columns suggests that there is no index in the targeted table which can be used to perform the search.

4.

Question 4

In the third task you performed three steps on the Orders table and then rewrote the SELECT query to use a trailing wildcard.

In the third task, Lucky Shrub used the following SELECT query, which contains a leading wildcard, to find the details of the employee whose last name is 'Tolo':

SELECT \* FROM Employees WHERE FullName LIKE '%Tolo';

You helped to optimize this query by replacing the leading wildcard with a trailing wildcard. Why is the use of a trailing wildcard a more optimal approach?

1 / 1 point

With a trailing wild card, MySQL can make use of an index created on the column to which the wildcard is assigned.

A trailing wild card provides more accurate results when the SQL query is executed.

A trailing wildcard needs to match a smaller number of rows when compared to a leading wildcard.

A trailing wild card takes less time to identify matches when compared to a leading wildcard.

Correct

Correct! If a leading wildcard is used in the WHERE clause condition, then MySQL cannot utilize an index. Therefore, the use of a trailing wildcard a more optimal approach.