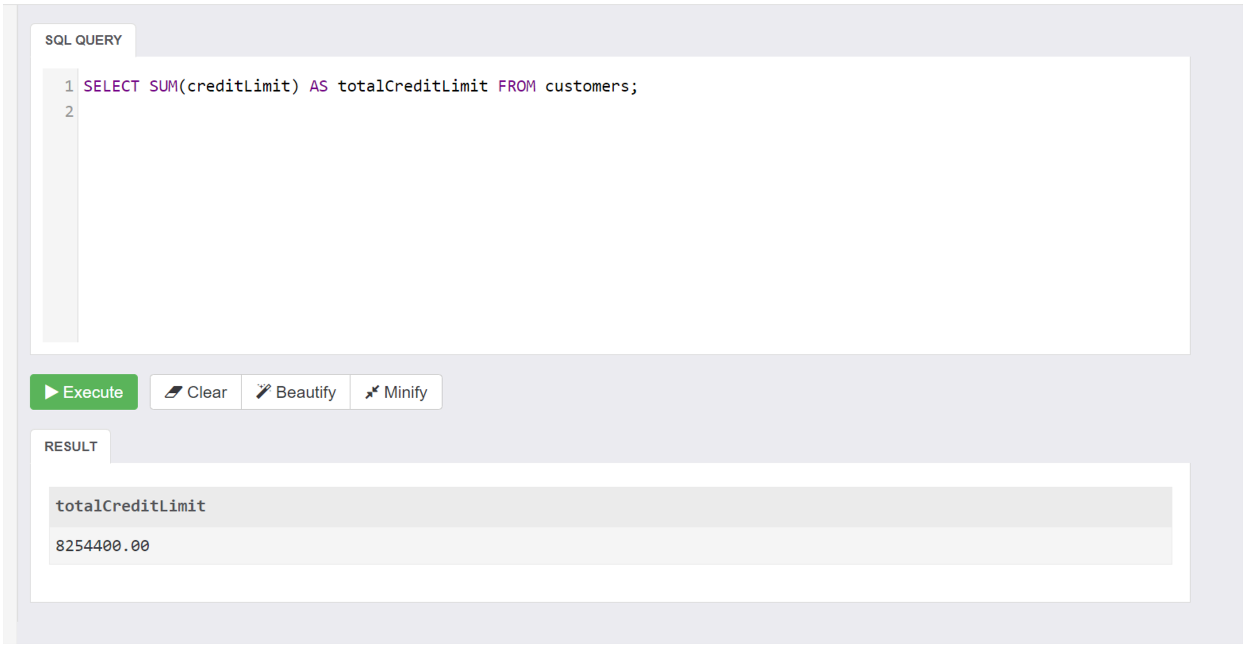
4.3.3 Assignment 1: Aggregate Functions

Scenario 1: Using the customers table from mysqltutorial.org, write an SQL statement to find the total credit limit of all customers.

Analysis:This scenario involves calculating the total credit limit across all customers.
The SUM aggregate function is used to add up the credit limits of all customers.

Table: customers
Column: creditLimit
Aggregate Function: SUM
Query:
SELECT SUM(creditLimit) AS totalCreditLimit FROM customers;



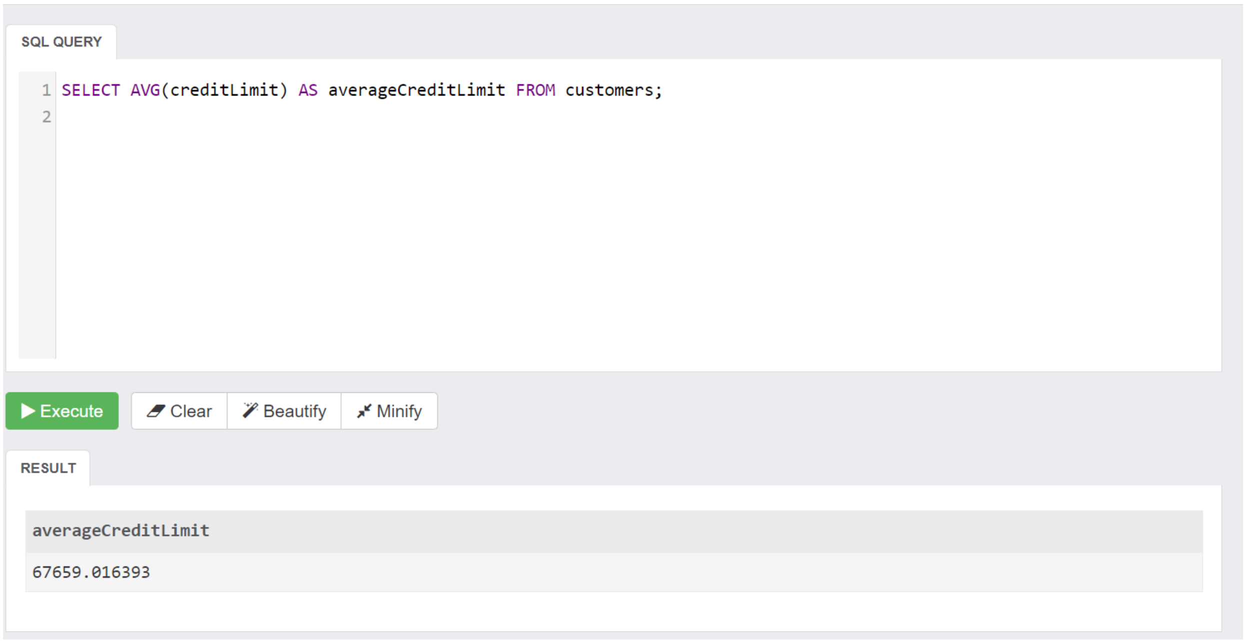
```
SELECT SUM(creditLimit) AS totalCreditLimit FROM customers;
```

```
totalCreditLimit
8254400.00
```

Scenario 2: Using the customers table from mysqltutorial.org, write an SQL statement to find the average credit limit of all customers.

Analysis:This scenario calculates the average credit limit of all customers.
The AVG aggregate function is used to find the average credit limit.

Table: customers
Column: creditLimit
Aggregate Function: AVG
Query:
SELECT AVG(creditLimit) AS averageCreditLimit FROM customers;

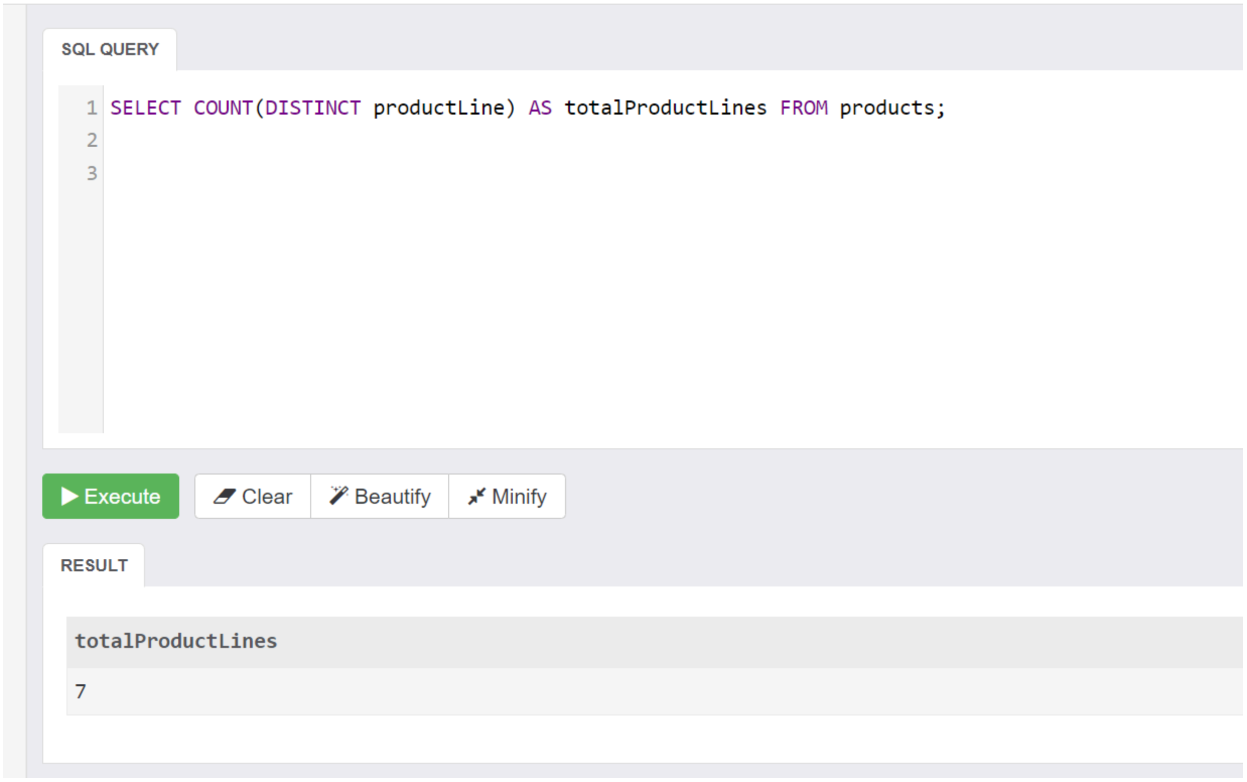


```
SELECT AVG(creditLimit) AS averageCreditLimit FROM customers;
```

```
averageCreditLimit
67659.016393
```

Scenario 3: Using the products table from mysqltutorial.org, write an SQL statement to find how many product lines are being offered by the company.

Analysis:This scenario counts the number of unique product lines offered by the company.
The COUNT(DISTINCT ...) is used to count the distinct product lines.
Table: products
Column: productLine
Aggregate Function: COUNT(DISTINCT productLine)
Query:
SELECT COUNT(DISTINCT productLine) AS totalProductLines FROM products;



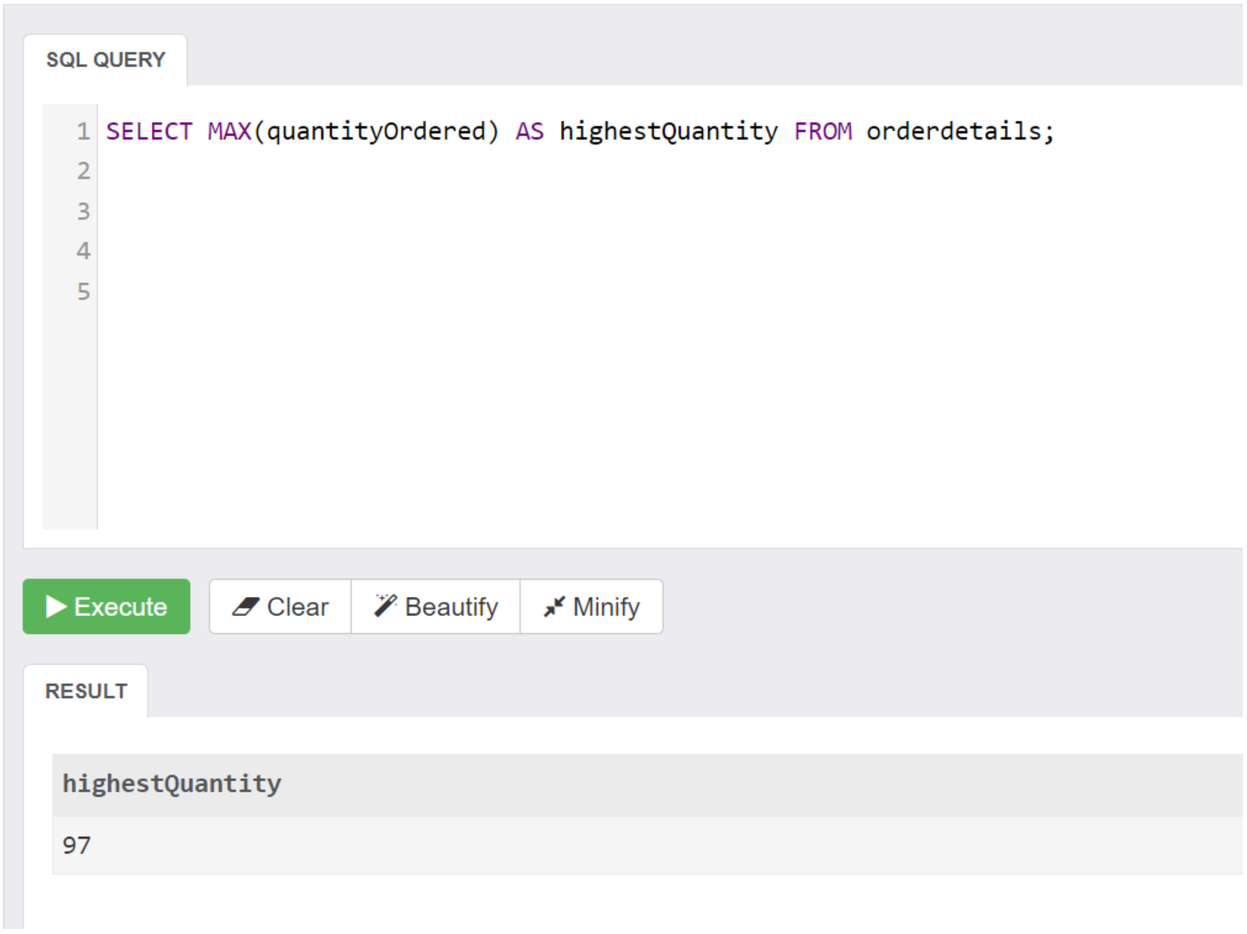
```
SELECT COUNT(DISTINCT productLine) AS totalProductLines FROM products;
```

```
totalProductLines
7
```

Scenario 4: Using the order_details table from mysqltutorial.org, write an SQL statement to get the highest quantity of orders.

AnalysisThis scenario aims to find the highest quantity of orders.
The MAX aggregate function is used to determine the maximum quantity ordered.

Table: order_details
Column: quantityOrdered
Aggregate Function: MAX
Query:
SELECT MAX(quantityOrdered) AS highestQuantity FROM order_details;



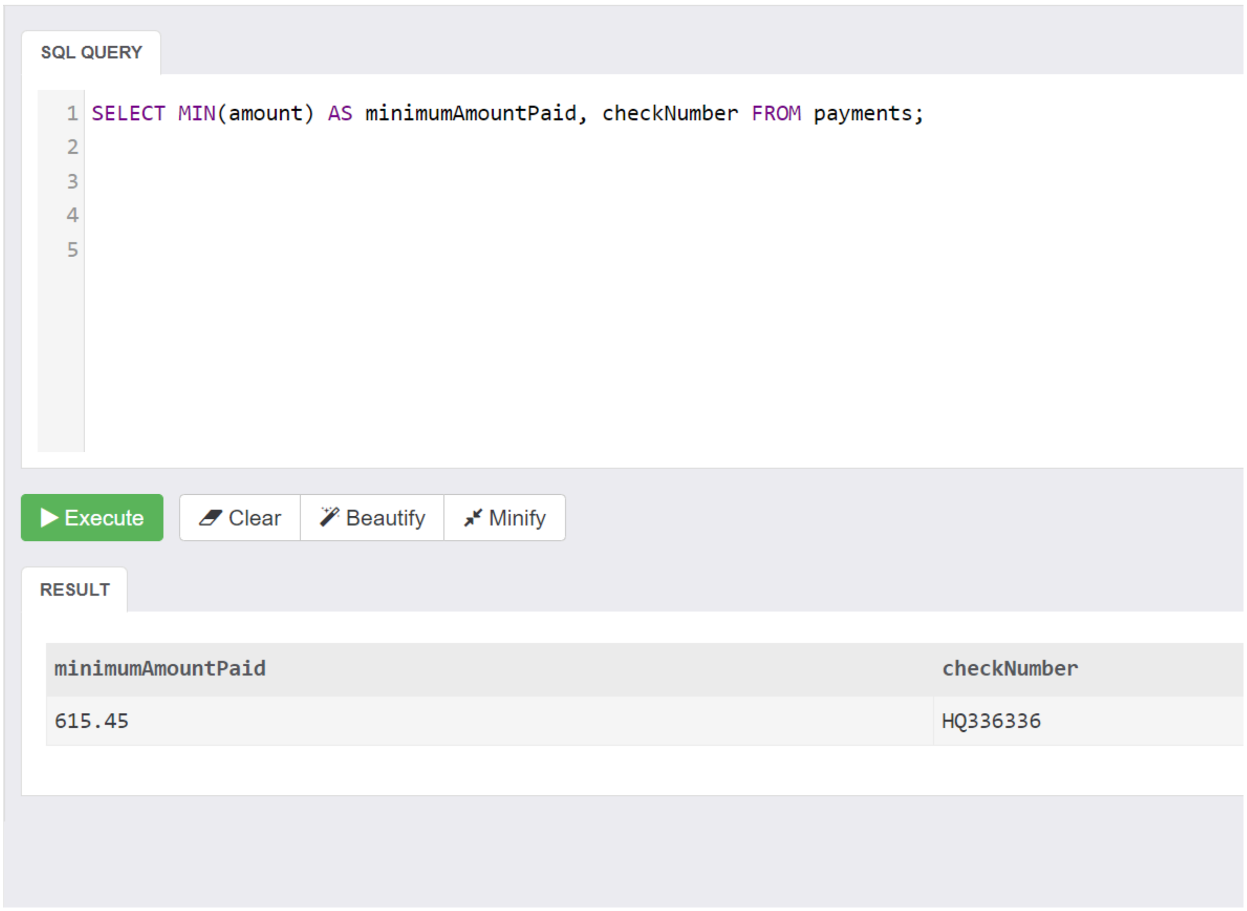
```
SELECT MAX(quantityOrdered) AS highestQuantity FROM orderdetails;
```

```
highestQuantity
97
```

Scenario 5: Using the payments table from mysqltutorial.org, write an SQL statement to get the minimum amount paid by the customer and the check number.

Analysis:This scenario retrieves the minimum amount paid by the customer along with the corresponding check number.
The MIN aggregate function is used to find the minimum payment amount.

Table: payments
Column: amount, checkNumber
Aggregate Function: MIN
Query:
SELECT MIN(amount) AS minimumAmountPaid, checkNumber FROM payments;



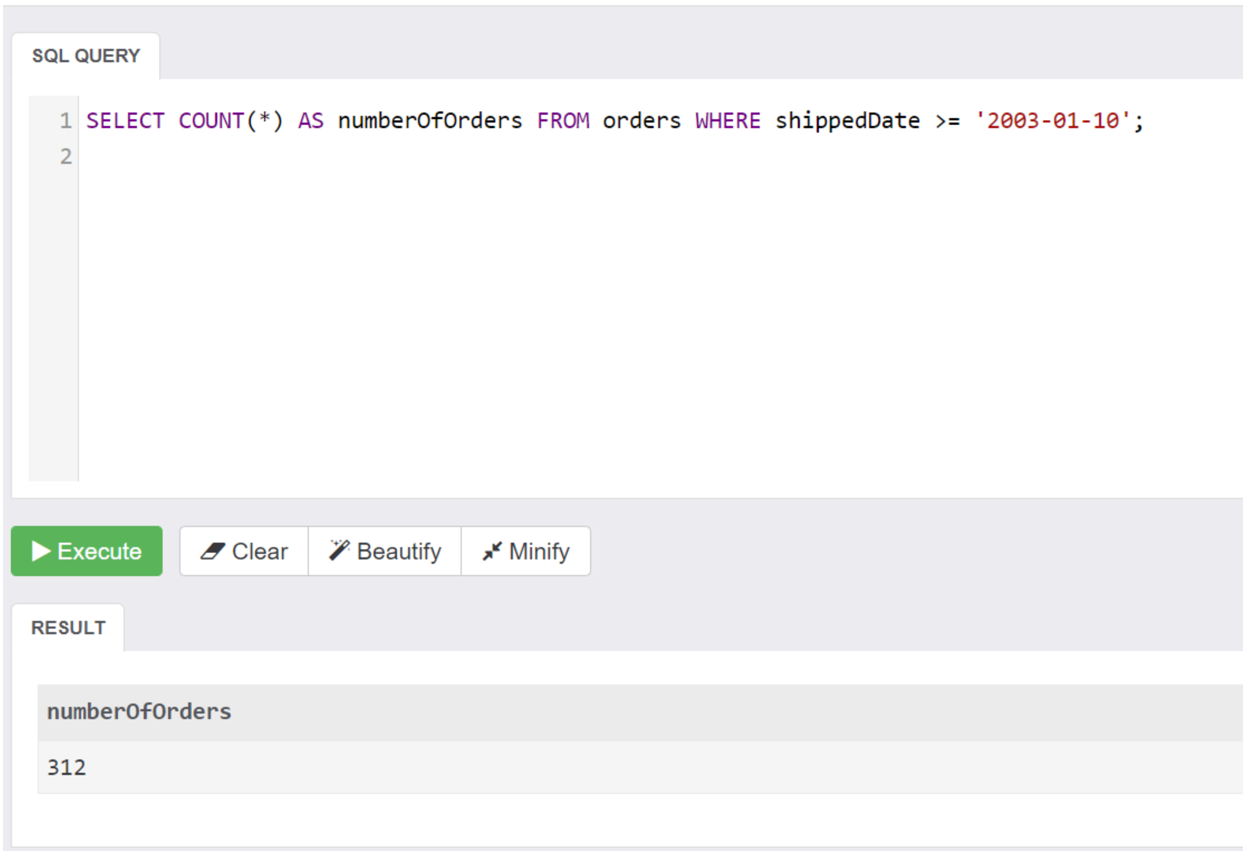
```sql
SELECT MIN(amount) AS minimumAmountPaid, checkNumber FROM payments;
```

minimumAmountPaid	checkNumber
615.45	HQ336336

Scenario 6: Using the orders table from mysqltutorial.org, write an SQL statement to get the number of orders shipped beginning 2003-01-10 and onwards.

Analysis:This scenario counts the number of orders shipped beginning from January 10, 2003. The COUNT aggregate function is used to calculate the total count.

Table: orders
Column: shippedDate
Condition: ShippedDate >= '2003-01-10'
Aggregate Function: COUNT
Query:
SELECT COUNT(*) AS numberOfOrders FROM orders WHERE shippedDate >= '2003-01-10';



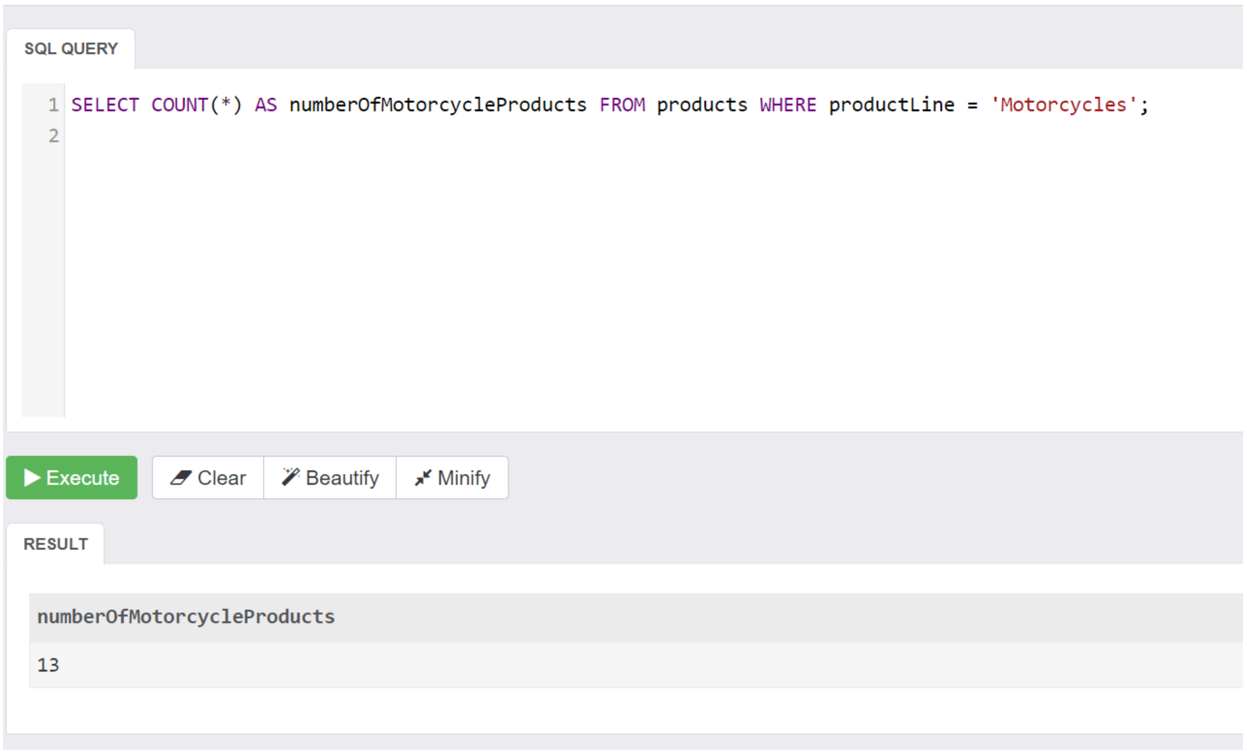
```
SELECT COUNT(*) AS numberOfOrders FROM orders WHERE shippedDate >= '2003-01-10';
```

```
numberOfOrders
312
```

Scenario 7: Using the products table from mysqltutorial.org, write an SQL statement to get the number of Motorcycle product codes.

Analysis:This scenario counts the number of product codes belonging to the 'Motorcycles' product line.
The COUNT aggregate function is used to calculate the total count.

Table: products
Column: productLine
Condition: productLine = 'Motorcycles'
Aggregate Function: COUNT
Query:
SELECT COUNT(*) AS numberOfMotorcycleProducts FROM products WHERE productLine = 'Motorcycles';



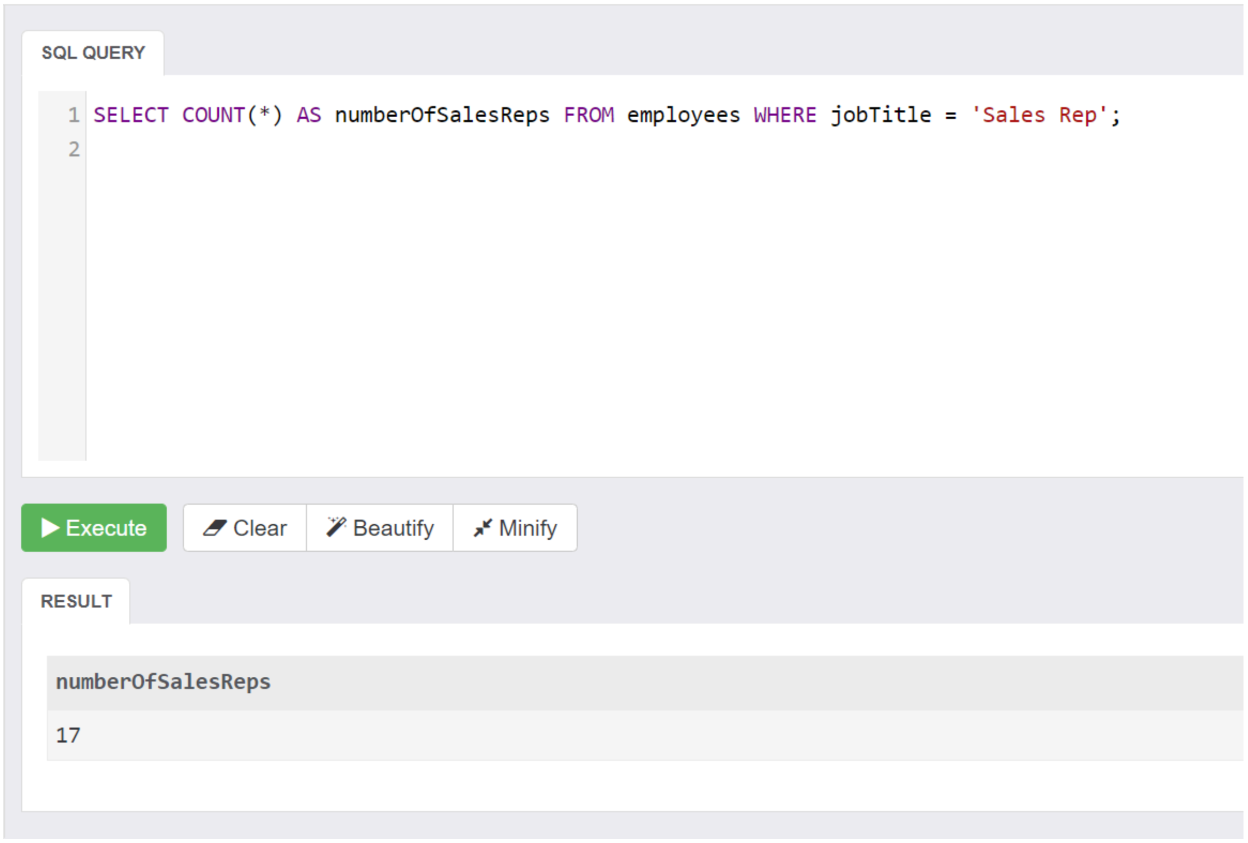
```sql
SELECT COUNT(*) AS numberOfMotorcycleProducts FROM products WHERE productLine = 'Motorcycles';
```

```
numberOfMotorcycleProducts
13
```

Scenario 8: This scenario counts the number of employees with the job title 'Sales Rep'. The COUNT aggregate function is used to calculate the total count.

Analysis:

Table: employees
Column: jobTitle
Condition: jobTitle = 'Sales Rep'
Aggregate Function: COUNT
Query:
SELECT COUNT(*) AS numberOfSalesReps FROM employees WHERE jobTitle = 'Sales Rep';



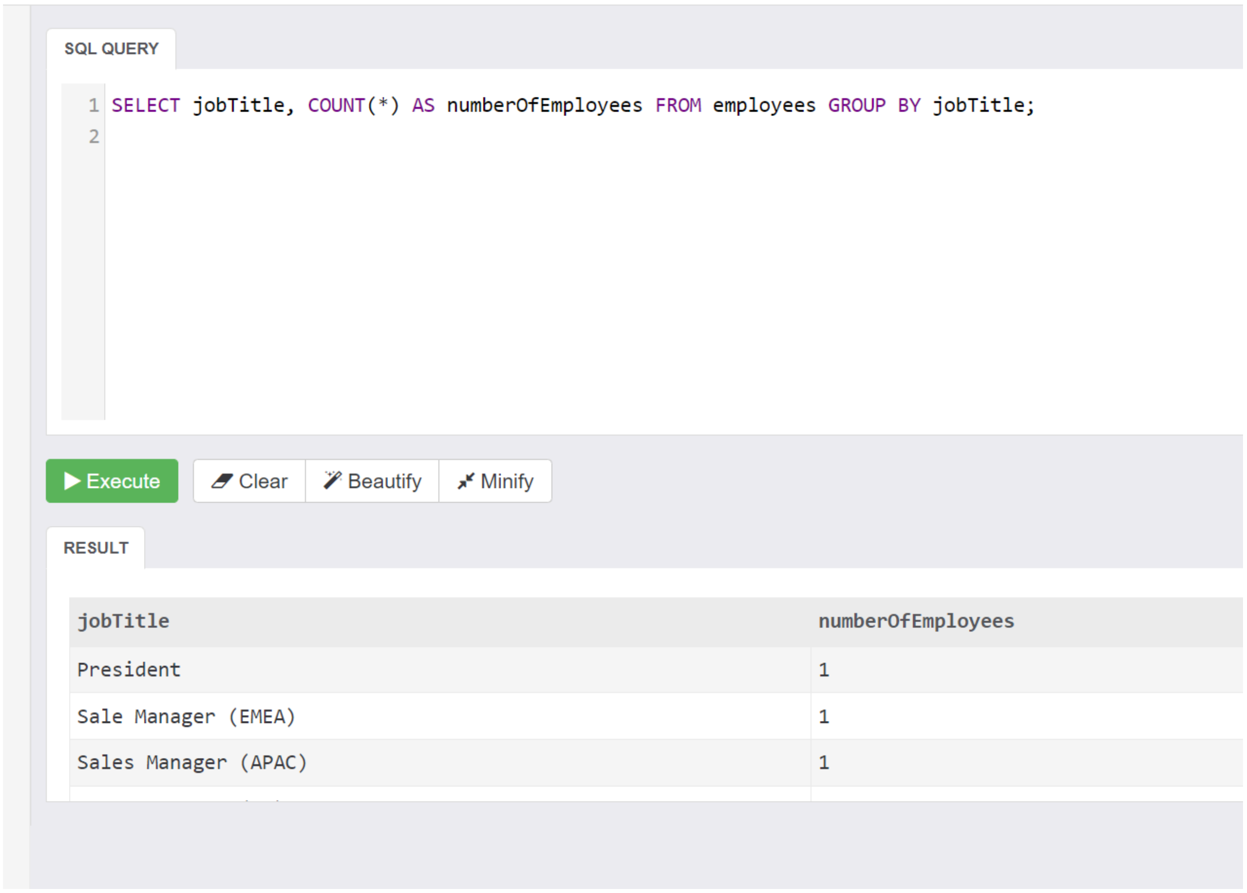
```
SELECT COUNT(*) AS numberOfSalesReps FROM employees WHERE jobTitle = 'Sales Rep';
```

```
numberOfSalesReps
17
```

Scenario 9: Using the employees table from mysqltutorial.org, write an SQL statement to get the number of employees for each position.

Analysis:This scenario counts the number of employees for each job position.
The COUNT aggregate function is used, and the results are grouped by the jobTitle column.

Table: employees
Column: jobTitle
Aggregate Function: COUNT
Group By: jobTitle
Query:
SELECT jobTitle, COUNT(*) AS numberOfEmployees FROM employees GROUP BY jobTitle;



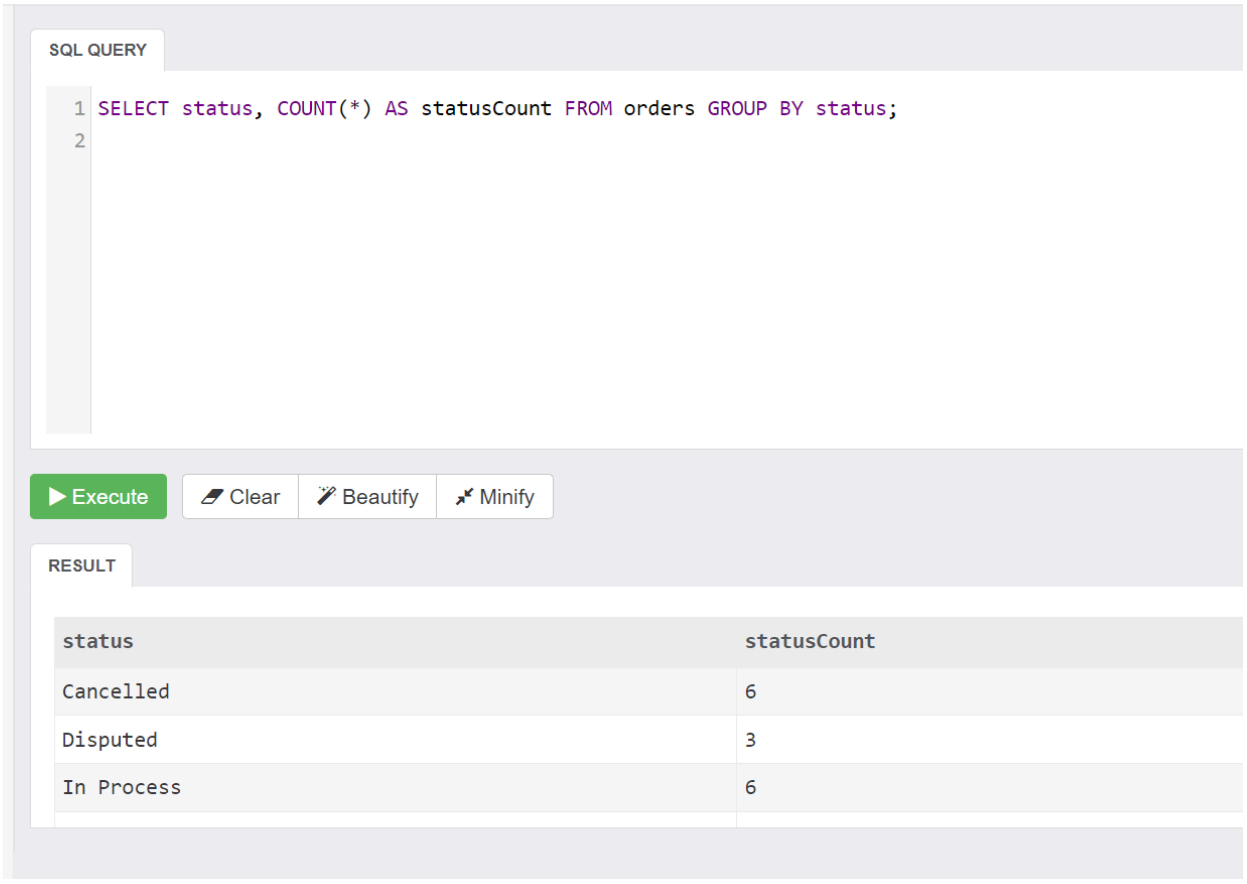
```
SELECT jobTitle, COUNT(*) AS numberOfEmployees FROM employees GROUP BY jobTitle;
```

jobTitle	numberOfEmployees
President	1
Sale Manager (EMEA)	1
Sales Manager (APAC)	1

Scenario 10: Using the orders table from mysqltutorial.org, write an SQL statement to show the various statuses in the orders table and how many occurrences for each status.

Analysis:This scenario retrieves various order statuses and counts the occurrences of each status.
The COUNT aggregate function is used, and the results are grouped by the status column.

Table: orders
Column: status
Aggregate Function: COUNT
Group By: status
Query:
SELECT status, COUNT(*) AS statusCount FROM orders GROUP BY status;



```sql
SELECT status, COUNT(*) AS statusCount FROM orders GROUP BY status;
```

status	statusCount
Cancelled	6
Disputed	3
In Process	6