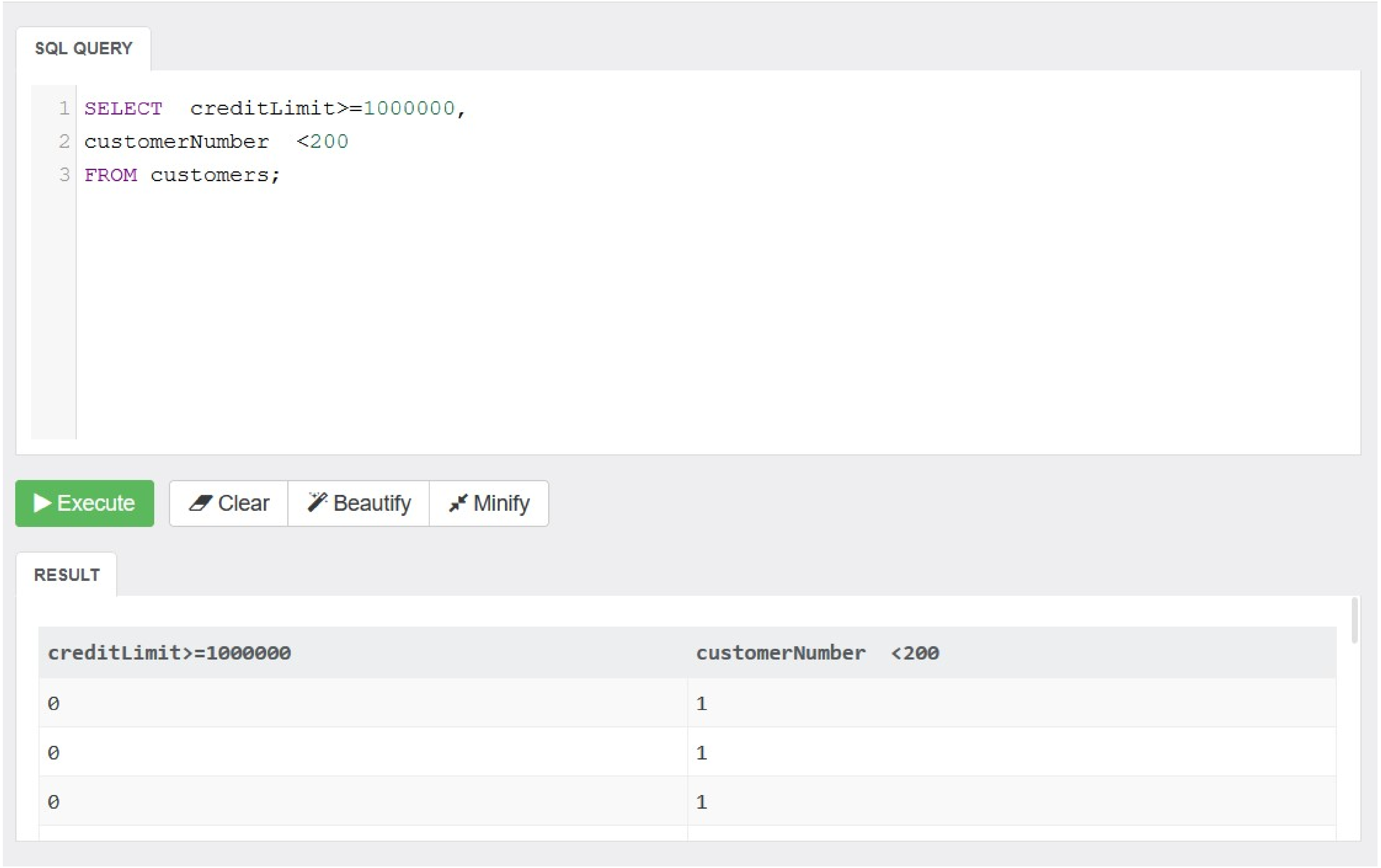
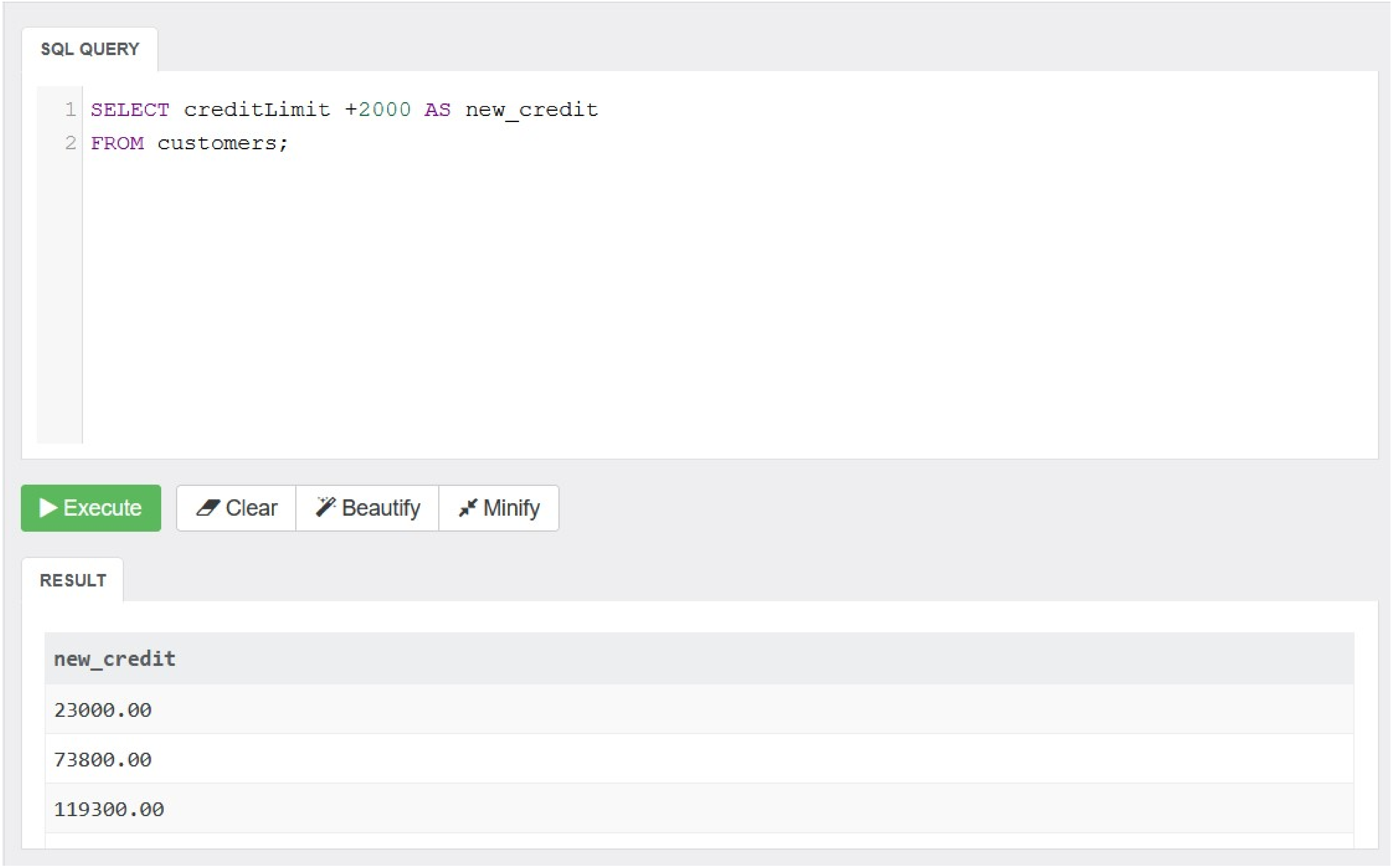


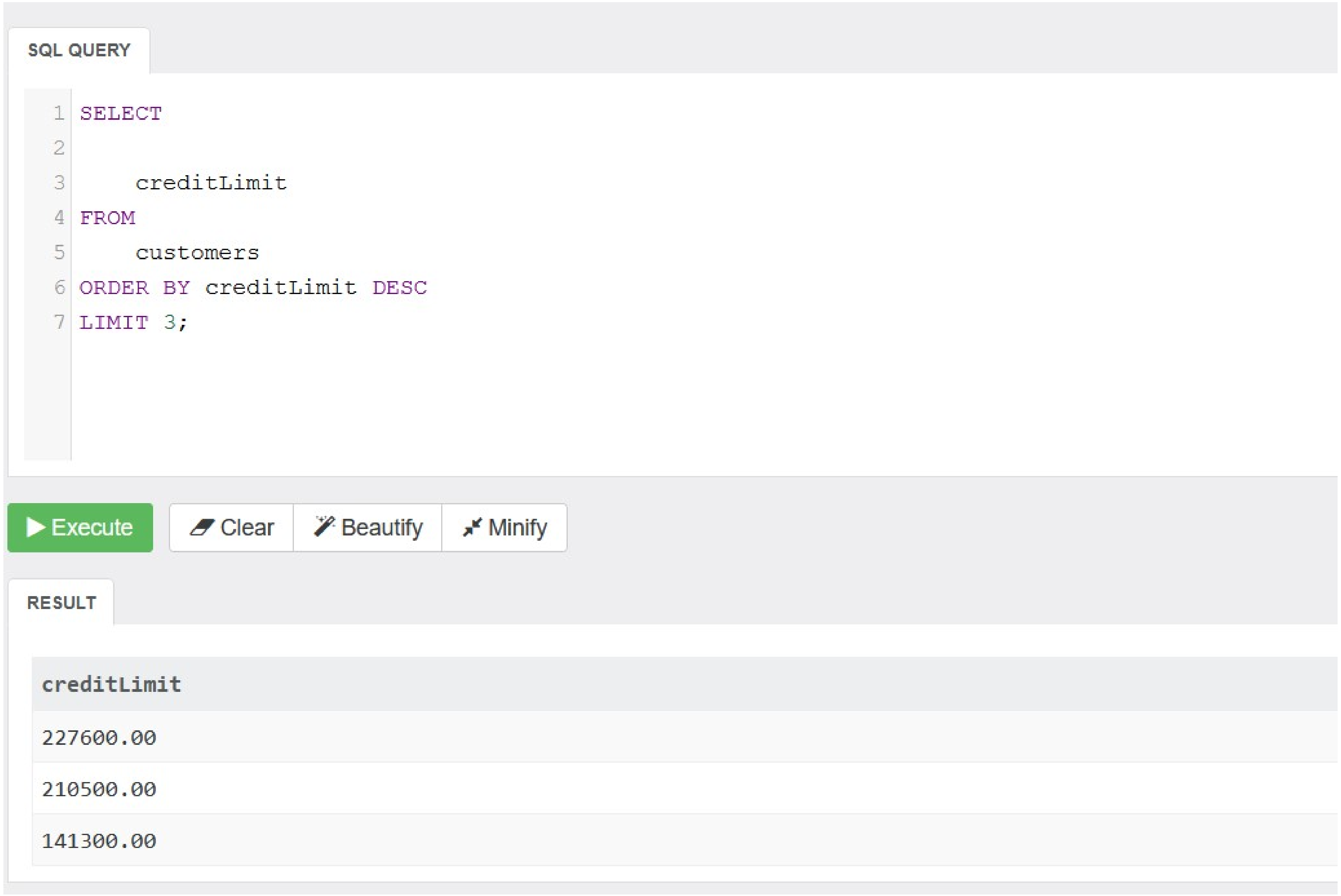
```sql
SELECT  creditLimit>=1000000,
customerNumber  <200
FROM customers;
```

creditLimit>=1000000	customerNumber <200
0	1
0	1
0	1



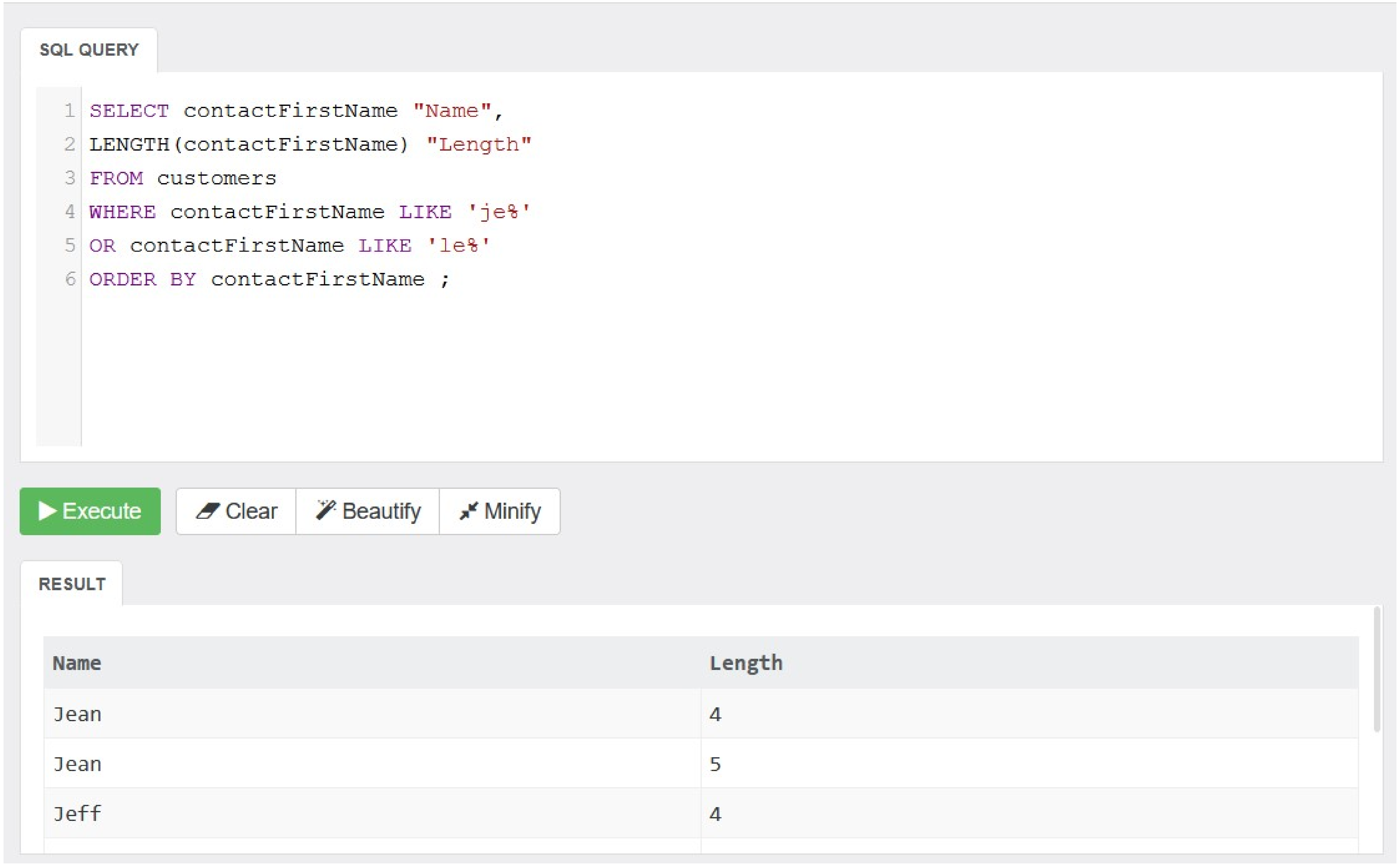
```sql
SELECT creditLimit +2000 AS new_credit
FROM customers;
```

new_credit
23000.00
73800.00
119300.00



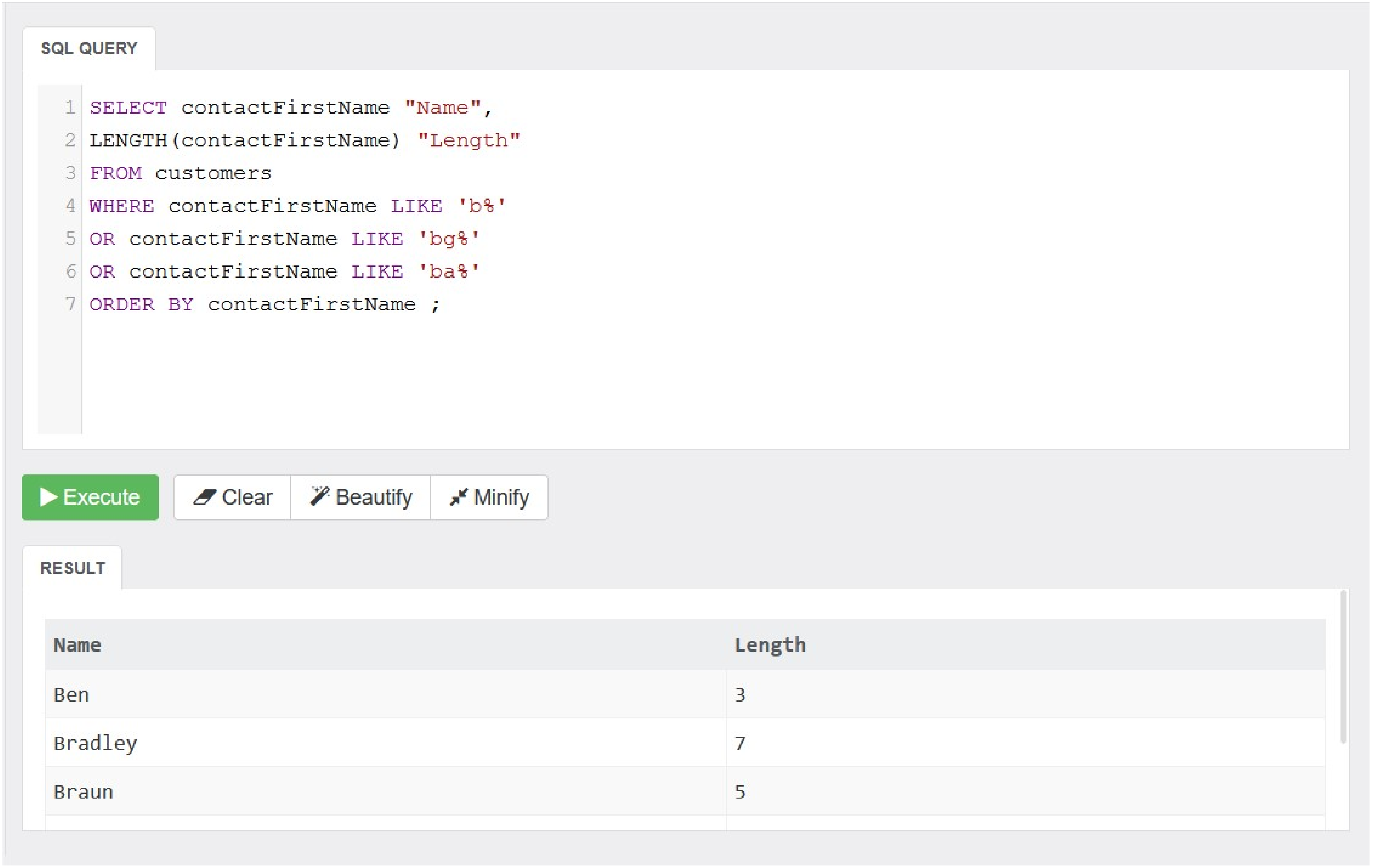
```sql
SELECT

    creditLimit
FROM
    customers
ORDER BY creditLimit DESC
LIMIT 3;
```

creditLimit
227600.00
210500.00
141300.00



```sql
SELECT contactFirstName "Name",
LENGTH(contactFirstName) "Length"
FROM customers
WHERE contactFirstName LIKE 'je%'
OR contactFirstName LIKE 'le%'
ORDER BY contactFirstName ;
```

Name	Length
Jean	4
Jean	5
Jeff	4



```
SELECT contactFirstName "Name",
LENGTH(contactFirstName) "Length"
FROM customers
WHERE contactFirstName LIKE 'b%'
OR contactFirstName LIKE 'bg%'
OR contactFirstName LIKE 'ba%'
ORDER BY contactFirstName ;
```

Name	Length
Ben	3
Bradley	7
Braun	5

MySQL supports another type of pattern matching operation based on the regular expressions and the REGEXP operator.

1. It provide a powerful and flexible pattern match that can help us implement power search utilities for our database systems.
2. REGEXP is the operator used when performing regular expression pattern matches. RLIKE is the synonym.
3. It also supports a number of metacharacters which allow more flexibility and control when performing pattern matching.
4. The backslash is used as an escape character. It's only considered in the pattern match if double backslashes have used.
5. Not case sensitive.

Pattern	**What the Pattern matches**
*	Zero or more instances of string preceding it
+	One or more instances of strings preceding it
.	Any single character
?	Match zero or one instances of the strings preceding it.
^	caret(^) matches Beginning of string
$	End of string
[abc]	Any character listed between the square brackets
[^abc]	Any character not listed between the square brackets

Pattern	What the Pattern matches
*	Zero or more instances of string preceding it
+	One or more instances of strings preceding it
.	Any single character
?	Match zero or one instances of the strings preceding it.
^	caret(^) matches Beginning of string
$	End of string
[abc]	Any character listed between the square brackets
[^abc]	Any character not listed between the square brackets
[A-Z]	match any upper case letter.
[a-z]	match any lower case letter
[0-9]	match any digit from 0 through to 9.
[[:<:]]	matches the beginning of words.
[[:>:]]	matches the end of words.
[:class:]	matches a character class i.e. [:alpha:] to match letters, [:space:] to match white space, [:punct:] is match punctuations and [:upper:] for upper class letters.
p1|p2|p3	Alternation; matches any of the patterns p1, p2, or p3
{n}	n instances of preceding element
{m,n}	m through n instances of preceding element