SQL QUERY

```sql
SELECT *
FROM customers
WHERE creditLimit >1000000 AND creditLimit <200
OR country ='USA'
```

Execute | Clear | Beautify | Minify

RESULT

customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1
112	Signal Gift Stores	King	Jean	7025551838	8489 Strong St.
124	Mini Gifts Distributors Ltd.	Nelson	Susan	4155551450	5677 Strong St.
129	Mini Wheels Co.	Murphy	Julie	6505555787	5557 North Pendale

SQL QUERY

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

Execute | Clear | Beautify | Minify

RESULT

new_credit
23000.00
73800.00
119300.00

SQL QUERY

```sql
SELECT creditLimit
FROM customers
LIMIT 3;
```

Execute | Clear | Beautify | Minify

RESULT

creditLimit
21000.00
71800.00
117300.00

SQL QUERY

```sql
SELECT contactFirstName
FROM customers
WHERE contactFirstName LIKE "je%"
OR contactFirstName LIKE "le%"

```

Execute | Clear | Beautify | Minify

RESULT

contactFirstName
Jean
Jeff
Jerry

SQL QUERY

```sql
SELECT contactFirstName
FROM customers
WHERE contactFirstName LIKE "%el"
OR contactFirstName LIKE "%ie"

```

Execute | Clear | Beautify | Minify

RESULT

contactFirstName
Julie
Veysel
Marie

SQL QUERY

```sql
SELECT contactFirstName
FROM customers
WHERE contactFirstName REGEXP "b[g]"
OR contactFirstName REGEXP "b[a]"

```

Execute | Clear | Beautify | Minify

RESULT

contactFirstName

You have seen MySQL pattern matching with **LIKE ...%**. MySQL supports another type of pattern matching operation based on the regular expressions and the **REGEXP** operator. If you are aware of PHP or PERL, then it is very simple for you to understand because this matching is same like those scripting the regular expressions.

Following is the table of pattern, which can be used along with the **REGEXP** operator.

Pattern	What the pattern matches
^	Beginning of string
$	End of string
.	Any single character
[...]	Any character listed between the square brackets
[^...]	Any character not listed between the square brackets
p1|p2|p3	Alternation; matches any of the patterns p1, p2, or p3
*	Zero or more instances of preceding element
+	One or more instances of preceding element
{n}	n instances of preceding element
{m,n}	m through n instances of preceding element