Data Selection and Manipulation

SQL

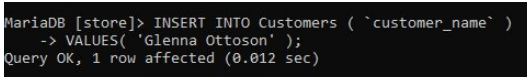
Creating Rows (Records)

Insert a Record

Now that we have defined a table complete with columns expecting data, we can look into the “C” of our CRUD! In SQL, we use “INSERT” to add new records to our table.

Give it a try:

```
INSERT INTO Customers ( `customer_name` )
  VALUES ( ‘Glenna Ottoson ’ );
```



```
MariaDB [store]> INSERT INTO Customers ( `customer_name` )
    -> VALUES( 'Glenna Ottoson' );
Query OK, 1 row affected (0.012 sec)
```

Note we are entering the table name after “INSERT INTO,” and the column name(s) we want to enter are enclosed afterward in parentheses. Column names we can enclose in backticks.

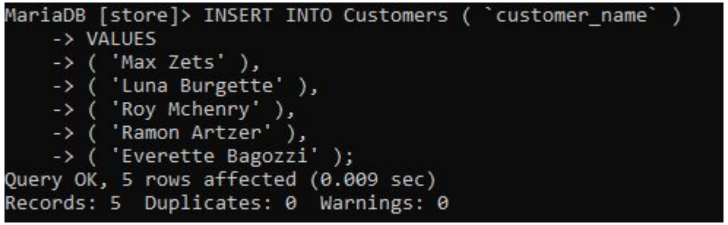
For the value(s) we want to assign to the columns of this record, we must use the VALUES keyword followed by parentheses and text enclosed in single or double quotation marks.

Inserting Multiple Records

We can insert multiple records, or rows, at once by adding more parenthesis-enclosed values separated by commas like so:

```
INSERT INTO Customers ( `customer_name` )
 VALUES
  ( ‘Max Zets’ ),
  ( ‘Luna Burgette’ ),
  ( ‘Roy Mchenry’ ),
  ( ‘Ramon Artzer’ ),
  ( ‘Everette Bagozzi’ );
```

Try it out! Pay attention to the names you enter, so that we can check if they’re all there when we’re done.



```
MariaDB [store]> INSERT INTO Customers ( `customer_name` )
    -> VALUES
    -> ( 'Max Zets' ),
    -> ( 'Luna Burgette' ),
    -> ( 'Roy Mchenry' ),
    -> ( 'Ramon Artzer' ),
    -> ( 'Everette Bagozzi' );
Query OK, 5 rows affected (0.009 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

At this point, if you followed both this and the previous slide, there should now be six entries in this table!

Reading Rows (Records)

Select Records

Let's read from the database and see if everything made it into our table intact. For this, we use the SELECT keyword.

Try the following statement—pay close attention, again, to the syntax we use:

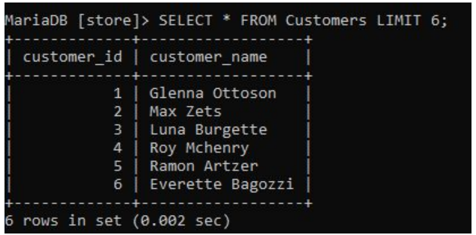
SELECT * FROM Customers LIMIT 6;

Try naming a column you want to read and note the difference:

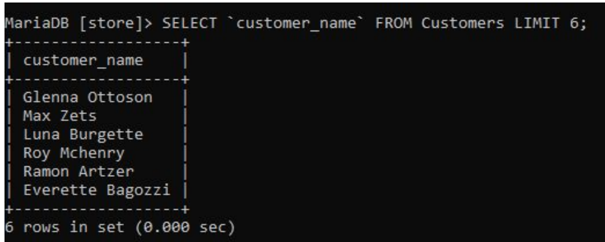
SELECT `customer_name` FROM Customers LIMIT 6;

The format here is SELECT, followed by column name(s) (or an asterisk, to show all column data), then FROM with the table name.

Optionally we can include a LIMIT to the number of results. This is especially useful in large databases, as queries like this can otherwise take a long time or many computer resources.



```
MariaDB [store]> SELECT * FROM Customers LIMIT 6;
+-------------+------------------+
| customer_id | customer_name    |
+-------------+------------------+
|           1 | Glenna Ottoson   |
|           2 | Max Zets         |
|           3 | Luna Burgette    |
|           4 | Roy Mchenry      |
|           5 | Ramon Artzer     |
|           6 | Everette Bagozzi |
+-------------+------------------+
6 rows in set (0.002 sec)
```



```
MariaDB [store]> SELECT `customer_name` FROM Customers LIMIT 6;
+------------------+
| customer_name    |
+------------------+
| Glenna Ottoson   |
| Max Zets         |
| Luna Burgette    |
| Roy Mchenry      |
| Ramon Artzer     |
| Everette Bagozzi |
+------------------+
6 rows in set (0.000 sec)
```

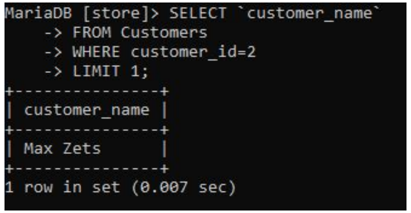
Updating Rows (Records)

Update Records

If we'd like to update a row in our table, we'll have to write out which table we're updating, what value we'd like to change, and which row it is that we want to target!

Firstly, let's decide which record we want to change. Let's say… the **second** one we entered. Let's refresh ourselves on whom that was so we can easily check for a change:

```
SELECT `customer_name`
  FROM Customers
  WHERE customer_id=2
  LIMIT 1;
```

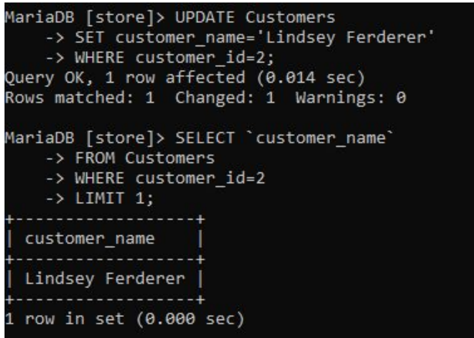


```
MariaDB [store]> SELECT `customer_name`
    -> FROM Customers
    -> WHERE customer_id=2
    -> LIMIT 1;
+---------------+
| customer_name |
+---------------+
| Max Zets      |
+---------------+
1 row in set (0.007 sec)
```

Alright! We'll change Max's name. In SQL, that that update might look like so:

```
UPDATE Customers
  SET customer_name="Lindsey Ferderer"
  WHERE customer_id=2;
```

Note we're using WHERE to let our database know which row, precisely, we want to update.



```
MariaDB [store]> UPDATE Customers
    -> SET customer_name='Lindsey Ferderer'
    -> WHERE customer_id=2;
Query OK, 1 row affected (0.014 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [store]> SELECT `customer_name`
    -> FROM Customers
    -> WHERE customer_id=2
    -> LIMIT 1;
+------------------+
| customer_name    |
+------------------+
| Lindsey Ferderer |
+------------------+
1 row in set (0.000 sec)
```

Deleting Rows (Records)

Delete Records

Much like with SELECT and UPDATE, we want to be mindful of which record(s) we are targeting. Especially when it comes to deletions, we need to be very, ***very*** careful. In most real databases, there are "soft deletes" or archives instead of an actual removal from a table.

A soft delete is typically a column that stores whether or not a record is "deleted" and should be hidden or not. An archive is a separate table that a record would be moved to for storage, instead of being permanently deleted. Both of these options allow us to restore or at least read "discarded" rows.

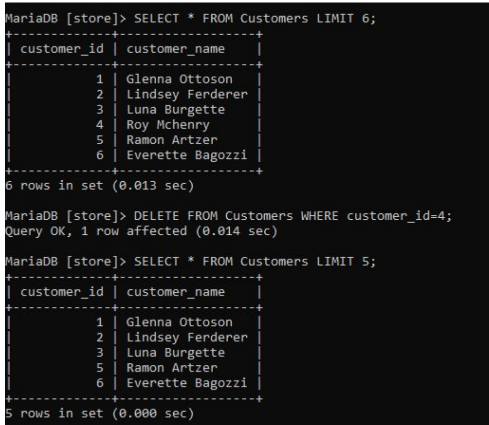
*IT IS VERY IMPORTANT TO NOTE THAT EXCLUDING THE "WHERE" CLAUSE WILL DELETE **ALL** RECORDS.

Let's delete the fourth customer in our table. First, we'll have a look at our customers before, delete our target, and check the listing again:

SELECT * FROM Customers LIMIT 6;

DELETE FROM Customers WHERE customer_id=4;

SELECT * FROM Customers LIMIT 5;



```
MariaDB [store]> SELECT * FROM Customers LIMIT 6;
+-------------+-----------------+
| customer_id | customer_name   |
+-------------+-----------------+
|           1 | Glenna Ottoson  |
|           2 | Lindsey Ferderer |
|           3 | Luna Burgette   |
|           4 | Roy Mchenry     |
|           5 | Ramon Artzer    |
|           6 | Everette Bagozzi |
+-------------+-----------------+
6 rows in set (0.013 sec)

MariaDB [store]> DELETE FROM Customers WHERE customer_id=4;
Query OK, 1 row affected (0.014 sec)

MariaDB [store]> SELECT * FROM Customers LIMIT 5;
+-------------+-----------------+
| customer_id | customer_name   |
+-------------+-----------------+
|           1 | Glenna Ottoson  |
|           2 | Lindsey Ferderer |
|           3 | Luna Burgette   |
|           5 | Ramon Artzer    |
|           6 | Everette Bagozzi |
+-------------+-----------------+
5 rows in set (0.000 sec)
```

SELECT

More Complex Selects

With the SELECT keyword, we can further customize our record queries with more than a simple “WHERE...=” and “LIMIT.” We can also adjust the ORDER of our result, select data from multiple tables at once, and more.

Make note of the following words that can help you make the most of your SELECTs:

DISTINCT	Removes any duplicates in the result rows.
COUNT	Returns the number (count) of results, instead of individual rows.
AS	Set up an alias (stand-in name) for a result column.
ORDER BY	Display results ordered via a specific column and ordering thereof.

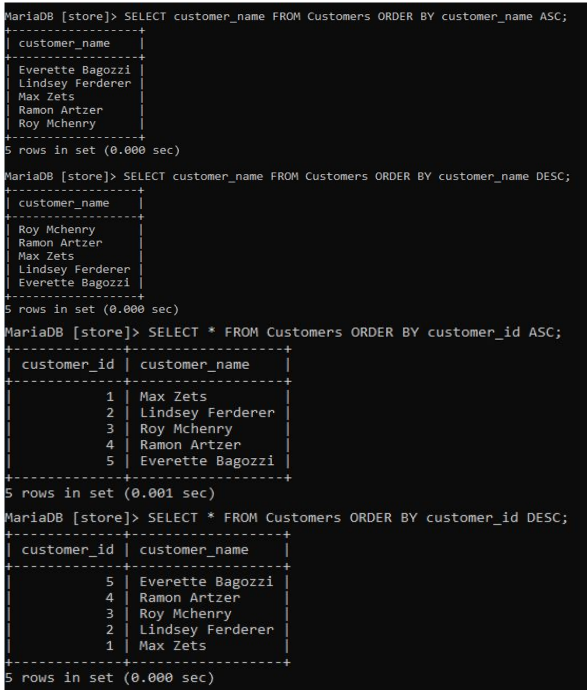
ORDER BY

The ORDER BY clause is used to sort and correctly order rows or records in your query result. Try the following examples and observe their effects:

```
SELECT customer_name FROM Customers ORDER BY customer_name ASC;

SELECT customer_name FROM Customers ORDER BY customer_name DESC;

SELECT * FROM Customers ORDER BY customer_id ASC;

SELECT * FROM Customers ORDER BY customer_id DESC;
```

You can even sort by multiple expressions, if necessary. It will be matched in the order you write them. If we had duplicate names, for instance, we could have the fallback ordering in such cases be by ID number:

```
SELECT customer_name FROM Customers
  ORDER BY
    customer_name ASC,
    customer_id ASC;
```



```
MariaDB [store]> SELECT customer_name FROM Customers ORDER BY customer_name ASC;
+------------------+
| customer_name    |
+------------------+
| Everette Bagozzi |
| Lindsey Ferderer |
| Max Zets         |
| Ramon Artzer     |
| Roy Mchenry      |
+------------------+
5 rows in set (0.000 sec)

MariaDB [store]> SELECT customer_name FROM Customers ORDER BY customer_name DESC;
+------------------+
| customer_name    |
+------------------+
| Roy Mchenry      |
| Ramon Artzer     |
| Max Zets         |
| Lindsey Ferderer |
| Everette Bagozzi |
+------------------+
5 rows in set (0.000 sec)

MariaDB [store]> SELECT * FROM Customers ORDER BY customer_id ASC;
+-------------+------------------+
| customer_id | customer_name    |
+-------------+------------------+
|           1 | Max Zets         |
|           2 | Lindsey Ferderer |
|           3 | Roy Mchenry      |
|           4 | Ramon Artzer     |
|           5 | Everette Bagozzi |
+-------------+------------------+
5 rows in set (0.001 sec)

MariaDB [store]> SELECT * FROM Customers ORDER BY customer_id DESC;
+-------------+------------------+
| customer_id | customer_name    |
+-------------+------------------+
|           5 | Everette Bagozzi |
|           4 | Ramon Artzer     |
|           3 | Roy Mchenry      |
|           2 | Lindsey Ferderer |
|           1 | Max Zets         |
+-------------+------------------+
5 rows in set (0.000 sec)
```

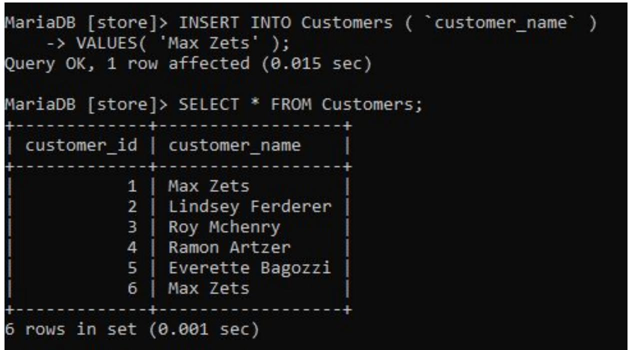
DISTINCT

Let’s add a duplicate entry to test the DISTINCT feature:

```
INSERT INTO Customers ( `customer_name` )
  VALUES ( ‘Max Zets’ );
```

Let’s confirm we have two identical entries:

```
SELECT * FROM Customers;
```



```
MariaDB [store]> INSERT INTO Customers ( `customer_name` )
    -> VALUES( 'Max Zets' );
Query OK, 1 row affected (0.015 sec)

MariaDB [store]> SELECT * FROM Customers;
+-------------+------------------+
| customer_id | customer_name    |
+-------------+------------------+
|           1 | Max Zets         |
|           2 | Lindsey Ferderer |
|           3 | Roy Mchenry      |
|           4 | Ramon Artzer     |
|           5 | Everette Bagozzi |
|           6 | Max Zets         |
+-------------+------------------+
6 rows in set (0.001 sec)
```

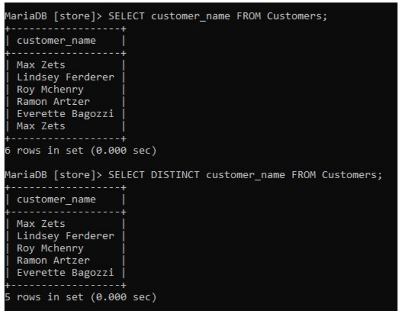
Alright. Let’s grab the list of names now:

```
SELECT customer_name FROM Customers;
```

Now let’s try the same request with the DISTINCT keyword:

```
SELECT DISTINCT customer_name FROM Customers;
```

Note the difference between these two results!



```
MariaDB [store]> SELECT customer_name FROM Customers;
+------------------+
| customer_name    |
+------------------+
| Max Zets         |
| Lindsey Ferderer |
| Roy Mchenry      |
| Ramon Artzer     |
| Everette Bagozzi |
| Max Zets         |
+------------------+
6 rows in set (0.000 sec)

MariaDB [store]> SELECT DISTINCT customer_name FROM Customers;
+------------------+
| customer_name    |
+------------------+
| Max Zets         |
| Lindsey Ferderer |
| Roy Mchenry      |
| Ramon Artzer     |
| Everette Bagozzi |
+------------------+
5 rows in set (0.000 sec)
```

Recommended Reading

It’s a good idea to read up more on what we’ve covered today. Have a look at the following:

- Beaulieu, A. (March 2020). Learning SQL, 3rd Edition. O’Reilly Media, Inc.
  - Chapter 2. Creating and Populating a Database
    - Populating and Modifying Tables