

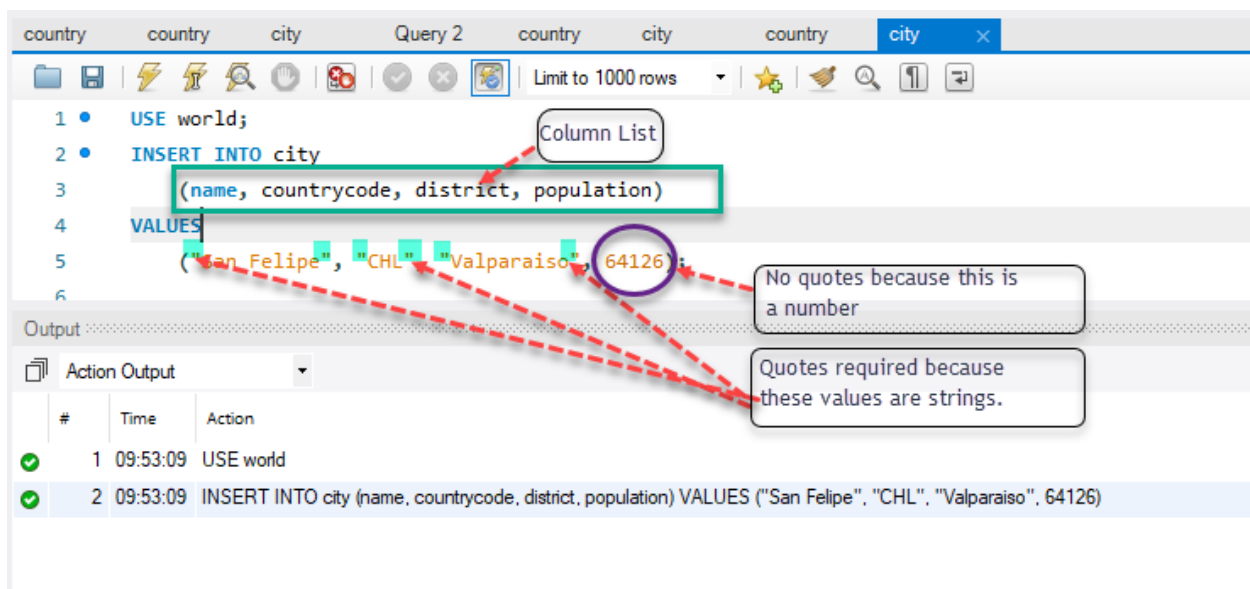
How to Insert, Update, Delete Data in Tables

The INSERT Clause with a Column List

- You can INSERT single or multiple rows at a time.
- An INSERT with a column list DOES NOT requires you to provide a value for each column. If you do not want to provide a value for a specific column, you do not have to include it in the column list. For columns that allows null values, the system will automatically provide a null value for you.
- If you want a column that provides a default value such as an auto-increment column to be populated with the default value, you do not need to list the column in the column list. The system will automatically provide the default value.
- When coding with a column list, the columns may appear in any order as long as the VALUES list matches the order of the column list.

Below is a basic example of an INSERT statement with a column list.

```
1  USE world;
2  INSERT INTO city
3      (name, countryCode, district, population)
4  VALUES
5      ("San Felipe", "CHL", "Valparaiso", 64126);
```



Results of the Insert

Query 2

```

1 • SELECT *
2 FROM CITY
3 WHERE name = 'san felipe'
4 AND countrycode = 'chl';
5

```

An auto-increment value was automatically provided.

ID	Name	CountryCode	District	Population
4082	San Felipe	CHL	Valparaiso	64126
NULL	NULL	NULL	NULL	NULL

INSERT INTO city

- Insert the value into the city table. The INTO keyword is not required.

(name, countryCode, district, population)

- The column list is comma separated and enclosed in parentheses.

VALUES

- The VALUES keyword is between the column list and the actual values. No commas are necessary.

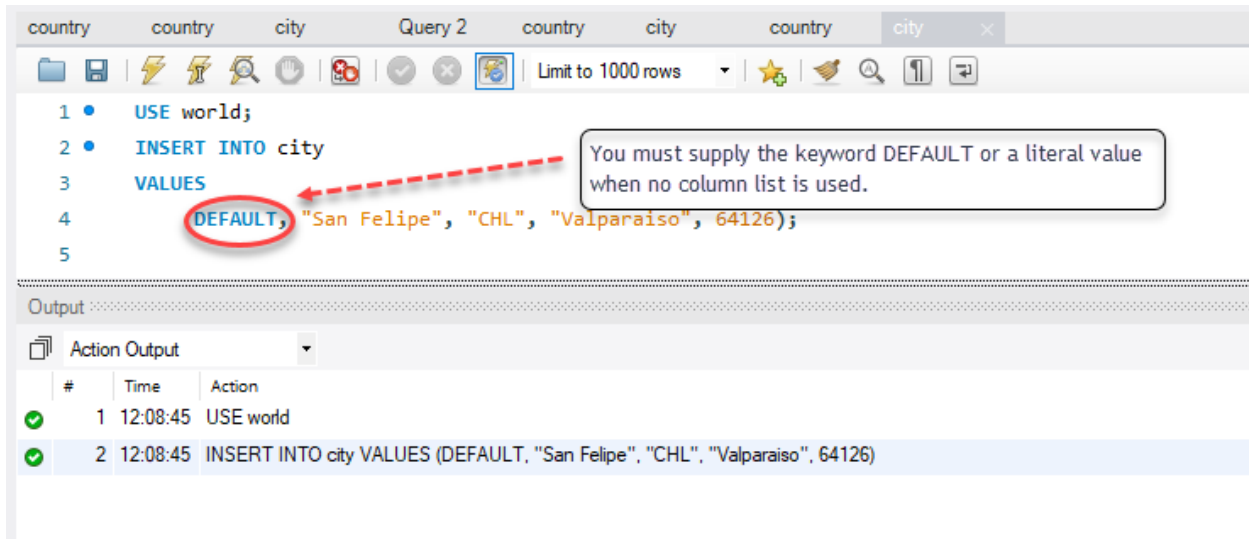
("San Felipe", "CHL", "Valparaiso", 64126);

- The values order must appear in the corresponding order of the column list.
- You must enclose strings in quotes.
- You must not enclose numbers in quotes.
- You do not have to specify columns that allow null values or default values in the column list. They will automatically get a null or default value.

The INSERT Clause without a Column List

- You can INSERT single or multiple rows at a time.
- An INSERT without a column list requires you to provide a value for every column.
- You must list values in the same order that they appear on the table.
- You must explicitly use the keyword "null" for columns that allow for nulls if you do not want to provide a value.
- You must explicitly use the keyword "DEFAULT" for columns that provide a default value if you do not want to provide one.

```
1  USE world;
2  INSERT INTO city
3  VALUES
4      (DEFAULT, "San Felipe", "CHL", "Valparaiso", 64126);
```



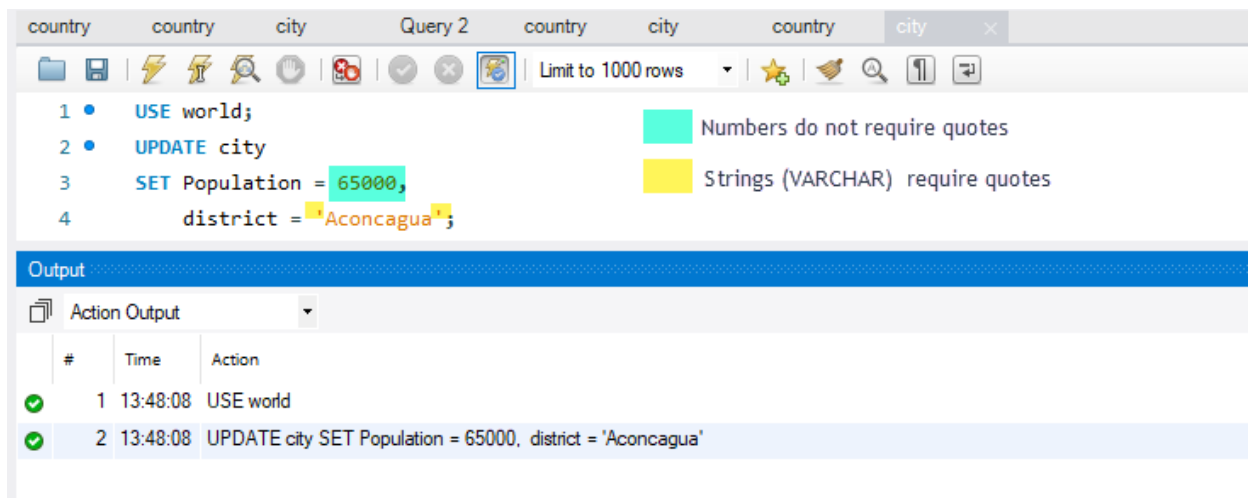
(DEFAULT "San Felipe", "CHL", "Valparaiso", 64126);

- The values order must appear in the same order they exist in the table.
- You must enclose strings in quotes.
- You must NOT enclose numbers in quotes.
- You must specify all column names and provide the keyword "DEFAULT" or a literal value for columns that provide a default option.
- If you do not want to provide a value for columns that allow null values, you must provide the keyword "null".

The UPDATE Clause with a Column List

- You can UPDATE single or multiple rows at a time.
- In a SET clause, you define the column along with its new value that may be a literal value or an expression.
- You can update one or all of the columns in a row.
- You can use a subquery or WHERE clause in an UPDATE statement.

```
1  USE world;  
2  UPDATE city  
3  SET Population = 65000, district = 'Aconcagua';
```



UPDATE city

- You indicate the table you want to UPDATE.

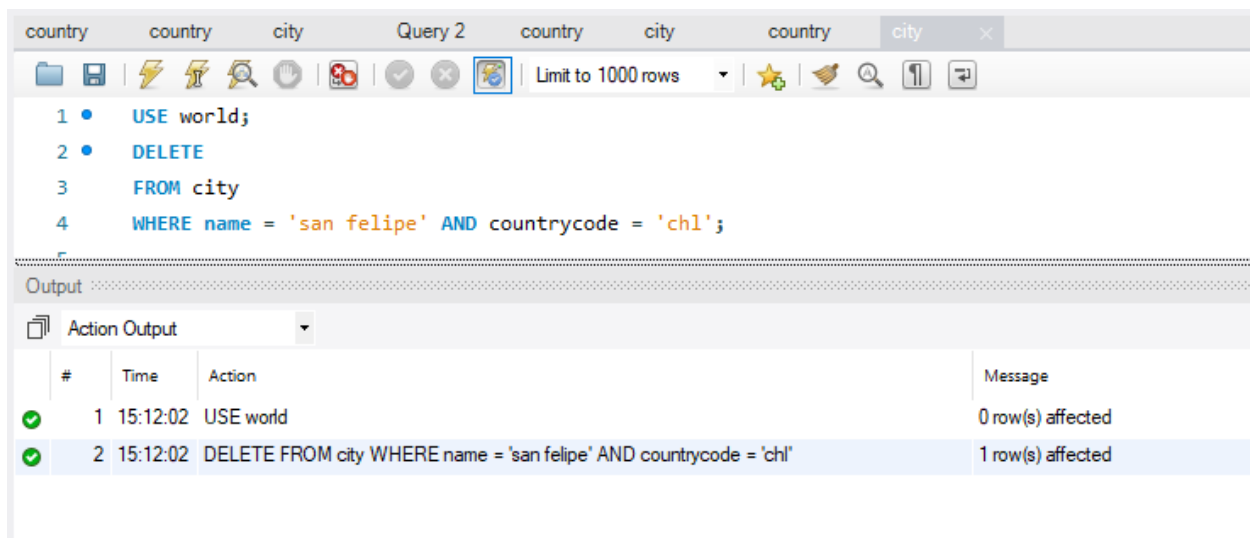
```
SET Population = 65000, district = 'Aconcagua';
```

- You indicate the table columns and associated values you want to change them to using the equals sign (=).
- You must separate each column and value with a comma.
- There is no trailing comma

The DELETE Clause with a Column

- You can delete single or multiple columns with a single statement.
- You can use a subquery or a WHERE clause with a DELETE statement.
- By default MySQL is in safe update mode which prevents coding a delete statement without a WHERE clause.

```
1  USE world;
2  DELETE
3  FROM city
4  WHERE name = 'san felipe' AND countrycode = 'chl';
```



The screenshot shows a MySQL query editor window with a toolbar and a query list. The query list contains four queries: 1. USE world; 2. DELETE 3. FROM city 4. WHERE name = 'san felipe' AND countrycode = 'chl';. Below the query list is an 'Output' section with a table showing the execution results.

#	Time	Action	Message
1	15:12:02	USE world	0 row(s) affected
2	15:12:02	DELETE FROM city WHERE name = 'san felipe' AND countrycode = 'chl'	1 row(s) affected

DELETE

- You begin a delete statement with the DELETE clause.

FROM city

- You must specify the table from which you are deleting rows.

```
WHERE name = 'san felipe' AND countrycode = 'chl';
```

- You should use a WHERE clause with a DELETE statement to avoid deleting every row in a table.