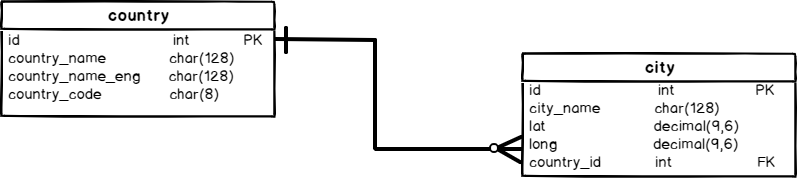
 CREATE DATABASE & CREATE TABLE Operations

Welcome to the first article in the [Learn SQL](https://www.sqlshack.com/learn-sql-create-database-create-table/#ToC) series. In this part, we’ll start with two essential commands in SQL: Create Database and Create Table. While both are pretty simple, they should be used first before you start working on anything with data (unless you use some template database).

Later in this series, I’ll try to cover everything essential for the complete beginner to jump into the magical world of SQL and databases. So, let’s start:



The goal of this article is to create a database (using the SQL Create Database command) and two tables (using the SQL Create Table command) as shown in the picture above. In the upcoming articles, we’ll insert data into these tables, update and delete data, but also add new tables and create queries.

What is a database?

Before we create a database using the SQL Create database command, I want to define what a database is. I’ll use the definition provided by Oracle:

*A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).*  
(source: <https://www.oracle.com/database/what-is-database.html>)

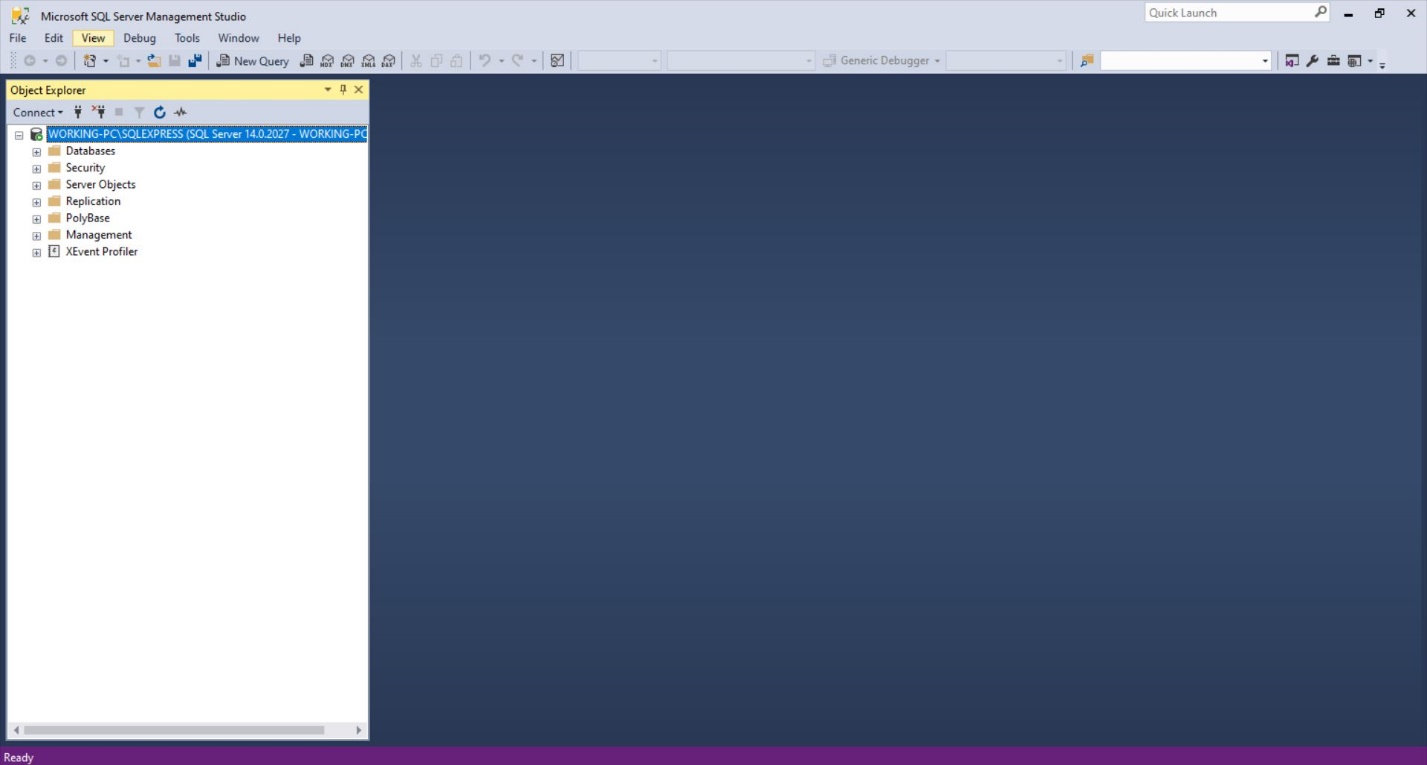
In this article, I’ll use the Microsoft SQL Server Express edition. So, DBMS is SQL Server, and the language we’ll use is T-SQL. Once again I’ll use a quote:

*T-SQL (Transact-SQL) is a set of programming extensions from Sybase and Microsoft that add several features to the Structured Query Language (SQL), including transaction control, exception and error handling, row processing and declared variables.*  
(source: <https://searchsqlserver.techtarget.com/definition/T-SQL>)

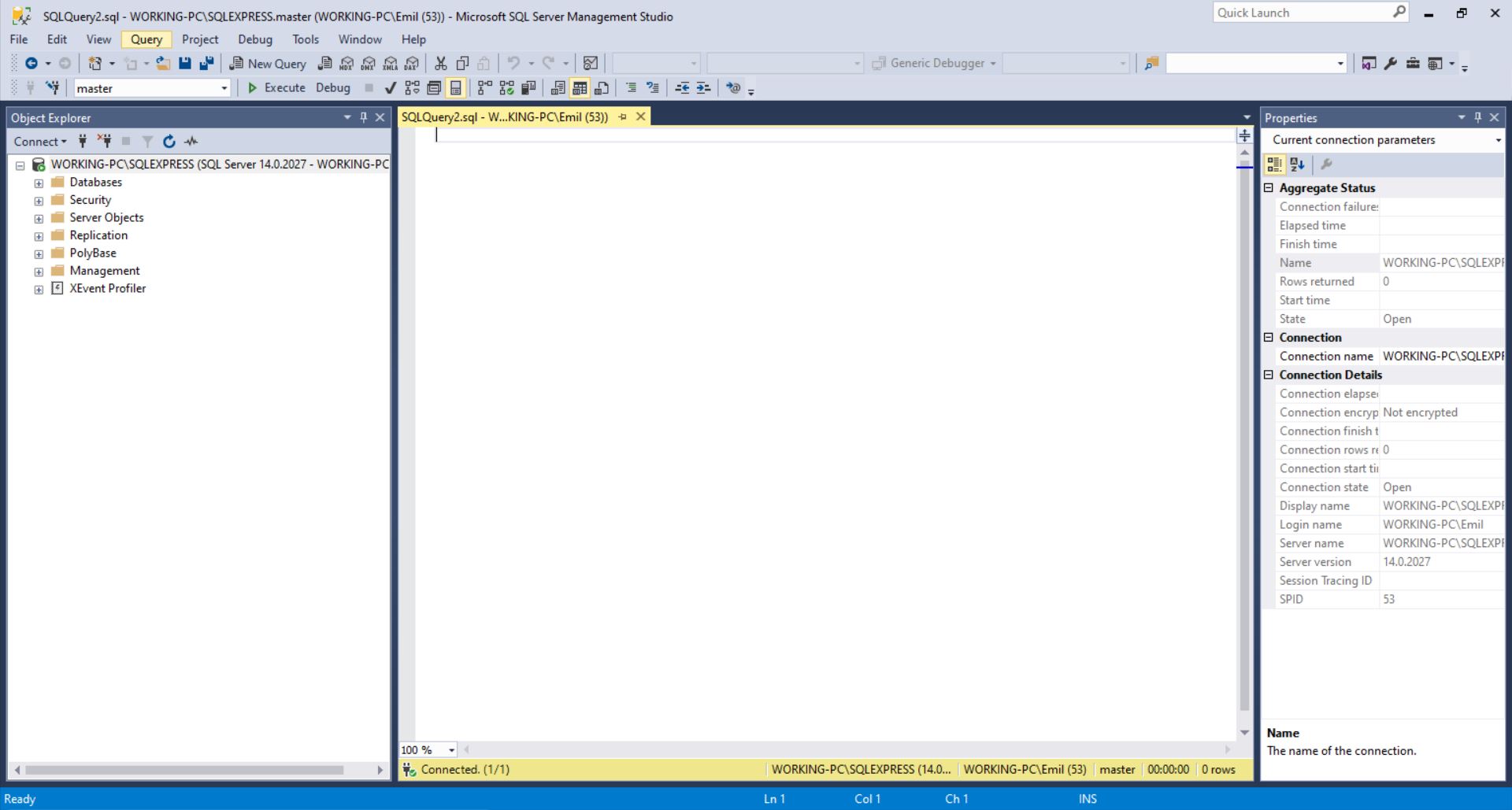
I won’t go in-depth in this article, but we can conclude this part with a statement that a database is an organized set of tables that contain data from the real-world and some additional columns needed for the system to work properly. We’ll discuss these in upcoming articles.

SQL Create Database statement

After installing and opening Microsoft [SQL Server Management Studio](https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15), our screen looks something like this:

[](https://www.sqlshack.com/wp-content/uploads/2019/12/sql-server.jpeg)

It doesn’t look fun at all. We’ll make it more fun by creating a new database. After clicking on the **New Query**, the new window opens and we’re able to type something in. It looks like on the picture below:

[](https://www.sqlshack.com/wp-content/uploads/2019/12/sql-server-new-query.jpeg)

Before typing anything, we should be sure we’re typing it in the right way. T-SQL is a language and as such it has its’ words – set of rules on how to write different commands.

Luckily, one of these commands is the SQL Create Database command. You can see the full T-SQL Create Database syntax on [Microsoft pages](https://docs.microsoft.com/en-us/sql/t-sql/statements/create-database-transact-sql?view=sql-server-ver15).

I’ll simplify it a lot and go only with the most basic form. In order to create a new database on our server, we need to use the following command:

|  |  |
| --- | --- |
| 1 | CREATE DATABASE database\_name; |

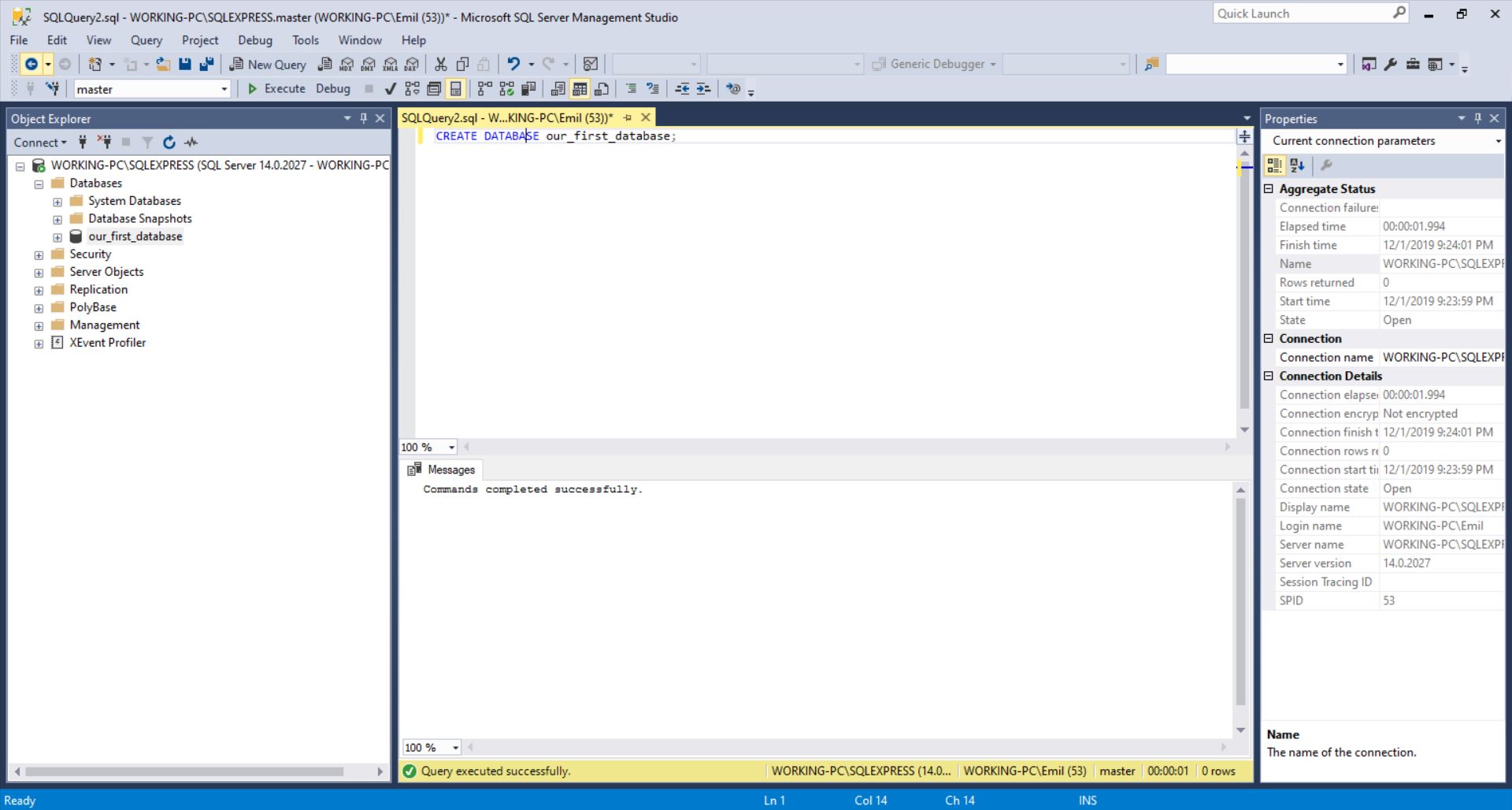
Where we’ll use the desired name instead of the *database\_name*.

SQL Create Database example

OK, let’s try it. We’ll run a command:

|  |  |
| --- | --- |
| 1 | CREATE DATABASE our\_first\_database; |

After running this command, our database is created, and you can see it in the databases list:

[](https://www.sqlshack.com/wp-content/uploads/2019/12/sql-server-sql-create-database.jpeg)

Click on the + next to the folder **Databases**, and besides two folders, you’ll also see that **our\_first\_database** had been created.

This is cool and you’ve just successfully created your first database. The problem is that we don’t have anything stored inside the database. Let’s change that.

SQL Create Table statement

In database theory, a table is a structure (“basic unit”) used to store data in the database.

I love to use analogies a lot, so I’ll do it here too. If you think of a library, a database is one shelf with books, and each book is a table. Each book has its own contents but is somehow related to other books on the same shelf – either by sharing some properties, either by just being close.

There is a lot of theory behind database tables, and how to decide what goes where, but the simplest you can do is following. When we look at our data and we need to decide what goes where we should group data in tables in such a manner that everything that belongs to the same real-life entity goes to the same table.

E.g. if we want to store data describing cities and countries, we’ll have two separate tables in our database – one for cities and another one for countries. We won’t mix their data but rather relate them. This goes out of the scope of this article and shall be covered in the upcoming parts of this series.

To define a table, we’ll follow the syntax. You can see full T-SQL [Create Table syntax here](https://docs.microsoft.com/en-us/sql/t-sql/statements/create-table-transact-sql?view=sql-server-ver15), but I’ll once more simplify the statement:

|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE table\_name (      column\_name column\_type,      column\_name column\_type,      ...  ); |

We’ll simply choose the name for our table and list all the columns we want to have in this table. Columns are also called attributes and each column describes a property of one record in the table. The column has its type and we should choose the type based on values we expect in that column (number, text, etc.).

SQL Create Table example

Let’s take a look at the definition of our two tables:

First, we’ll define the **city** table.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | -- Table: city  CREATE TABLE city (      id int  NOT NULL IDENTITY(1, 1),      city\_name char(128)  NOT NULL,      lat decimal(9,6)  NOT NULL,      long decimal(9,6)  NOT NULL,      country\_id int  NOT NULL,      CONSTRAINT city\_pk PRIMARY KEY  (id)  ); |

Please notice a few things:

* NOT NULL -> This is a property telling us that this column can’t be empty (must be defined)
* IDENTITY(1, 1) -> is also a property of the column telling us that this value shall be generated automatically, starting from 1 and increasing by 1 for each new record
* CONSTRAINT city\_pk PRIMARY KEY (id) -> This is not a column, but the rule, telling us that column id shall contain only UNIQUE values. So only 1 city can have id =5

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | -- Table: country  CREATE TABLE country (      id int  NOT NULL IDENTITY(1, 1),      country\_name char(128)  NOT NULL,      country\_name\_eng char(128)  NOT NULL,      country\_code char(8)  NOT NULL,      CONSTRAINT country\_ak\_1 UNIQUE (country\_name),      CONSTRAINT country\_ak\_2 UNIQUE (country\_name\_eng),      CONSTRAINT country\_ak\_3 UNIQUE (country\_code),      CONSTRAINT country\_pk PRIMARY KEY  (id)  ); |

Here we have 1 new CONSTRAINT and that is the UNIQUE constraining. This one tells us that this value must be UNIQUE within this table. E.g. *CONSTRAINT country\_ak\_1 UNIQUE (country\_name)* defines that we can’t store 2 countries with the same name.

The last part of the script is the definition of foreign keys. We have only 1 such key and it relates city and country table (*city.county\_id = country.id*).

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | -- foreign keys  -- Reference: city\_country (table: city)  ALTER TABLE city ADD CONSTRAINT city\_country      FOREIGN KEY (country\_id)      REFERENCES country (id); |

Keys (primary and foreign) are too complex and shall be covered in a separate article. After executing these commands, the status of our database is as in the picture below: