

# Databases and SQL Editor in PhpStorm



## Redirection Notice

This page will redirect to <https://www.jetbrains.com/help/phpstorm/relational-databases.html> in about 2 seconds.

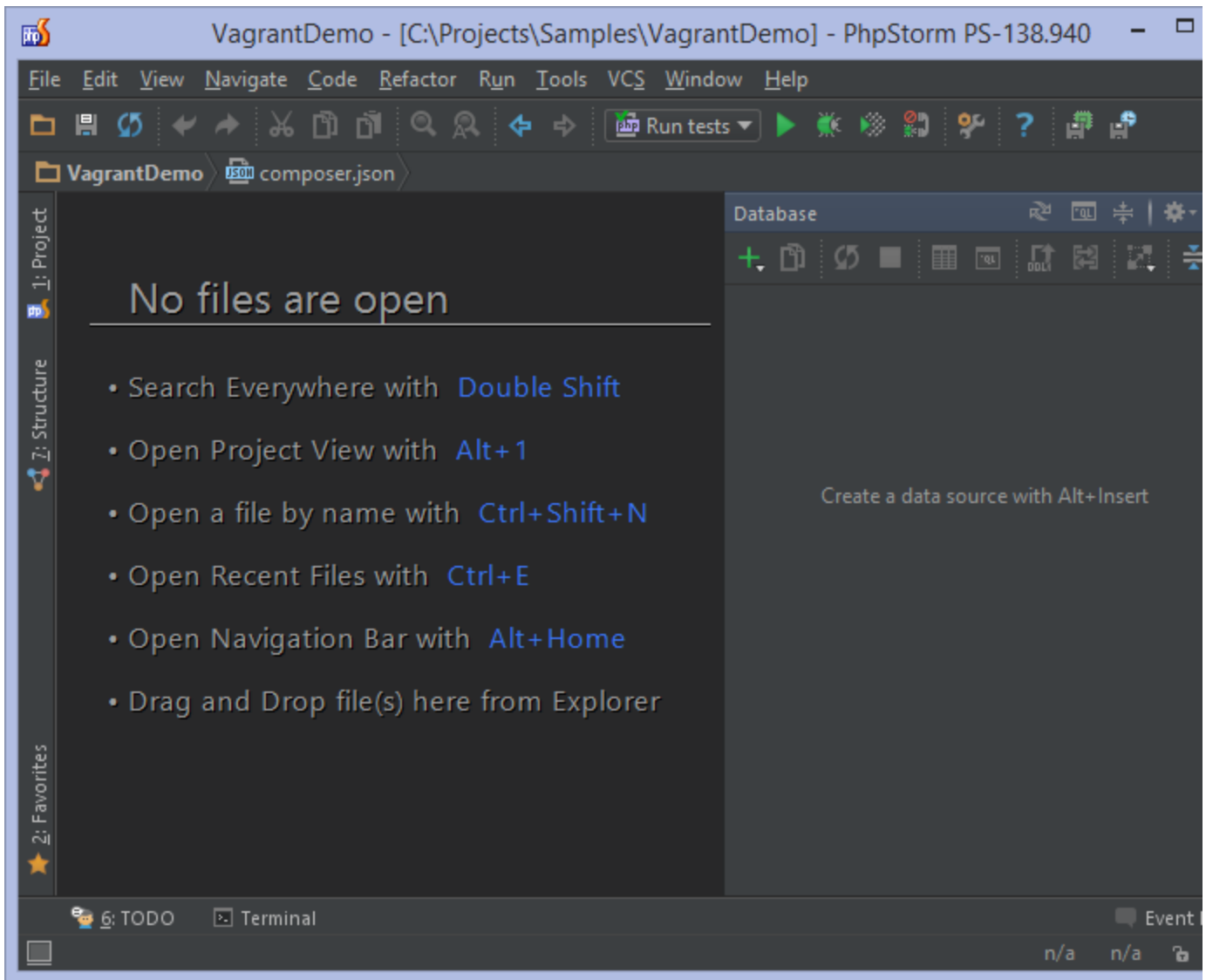
[Tweet](#)

When writing an application, chances are that we have to work with a database. PhpStorm can help us perform all types of routine database tasks, such as querying for a record; checking what that column was named again; database development where we have to create the schema structure; or just developing PHP code against a database. In this tutorial, we will see how.

- Working with the Database Tool Window
- Setting up a database connection
- Refactoring the Database
  - Creating a Table
  - Deleting a Table, Index or Column
  - Adding a Column to an Existing Table
  - Renaming a Table, Index or Column
  - Creating Indexes and Foreign Keys
  - Deleting Indexes and Foreign Keys
- Querying, Inserting and Updating data
  - Using the Table Editor
- Using the Database Console
  - Value View / Single Record Transpose View
- Generating a UML database diagram
- Database Migration
  - Exporting Database Tables as SQL Statements
  - Comparing and Migrating Database Code
- Database Code Completion while working in PHP
- Coloring Data Sources
- Read-Only Data Sources

## Working with the Database Tool Window

Database support is provided through the Database Tool Window on the right-hand side of the IDE. Use the View | Tool Windows | Database menu or Find Action, by pressing Ctrl+Alt+A (Alt-CMD-A on Mac OS X) and search for "Database" to open it.

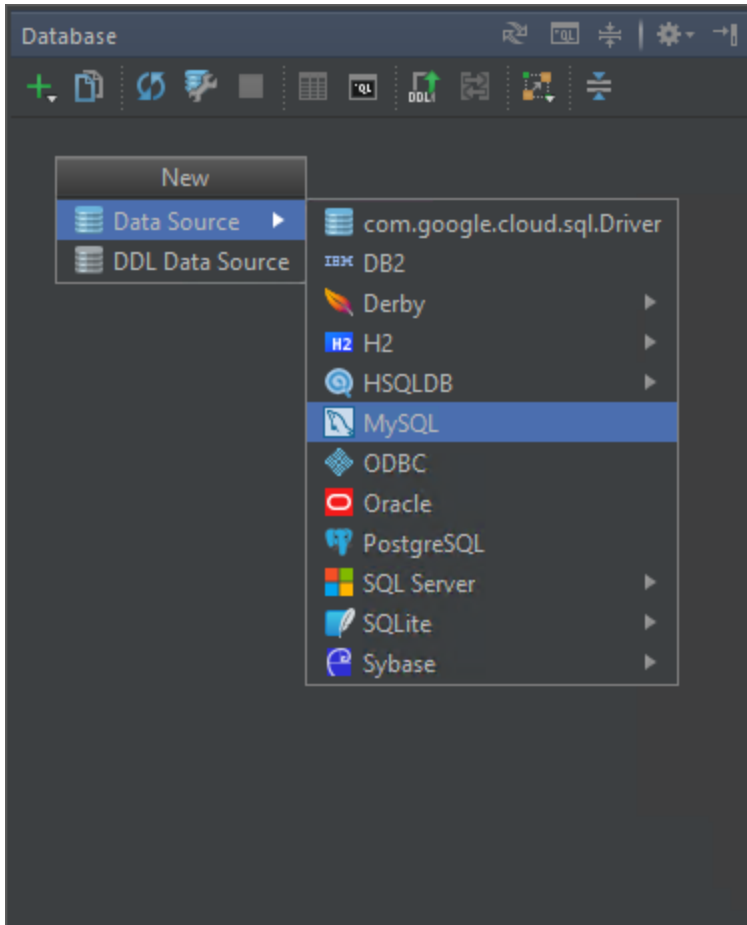


From the Database tool window, we can connect to various data sources and view database structure; modify it by refactoring tables and columns; manage data stored in tables by simply editing or adding values; write arbitrary SQL queries with code completion and syntax highlighting; and run SQL queries from code or the built-in SQL editor. We can generate UML diagrams showing relations between tables, copy records as INSERT statements, get smart completion when writing joins on tables, and so on.

But first things first: let's set up a database connection.

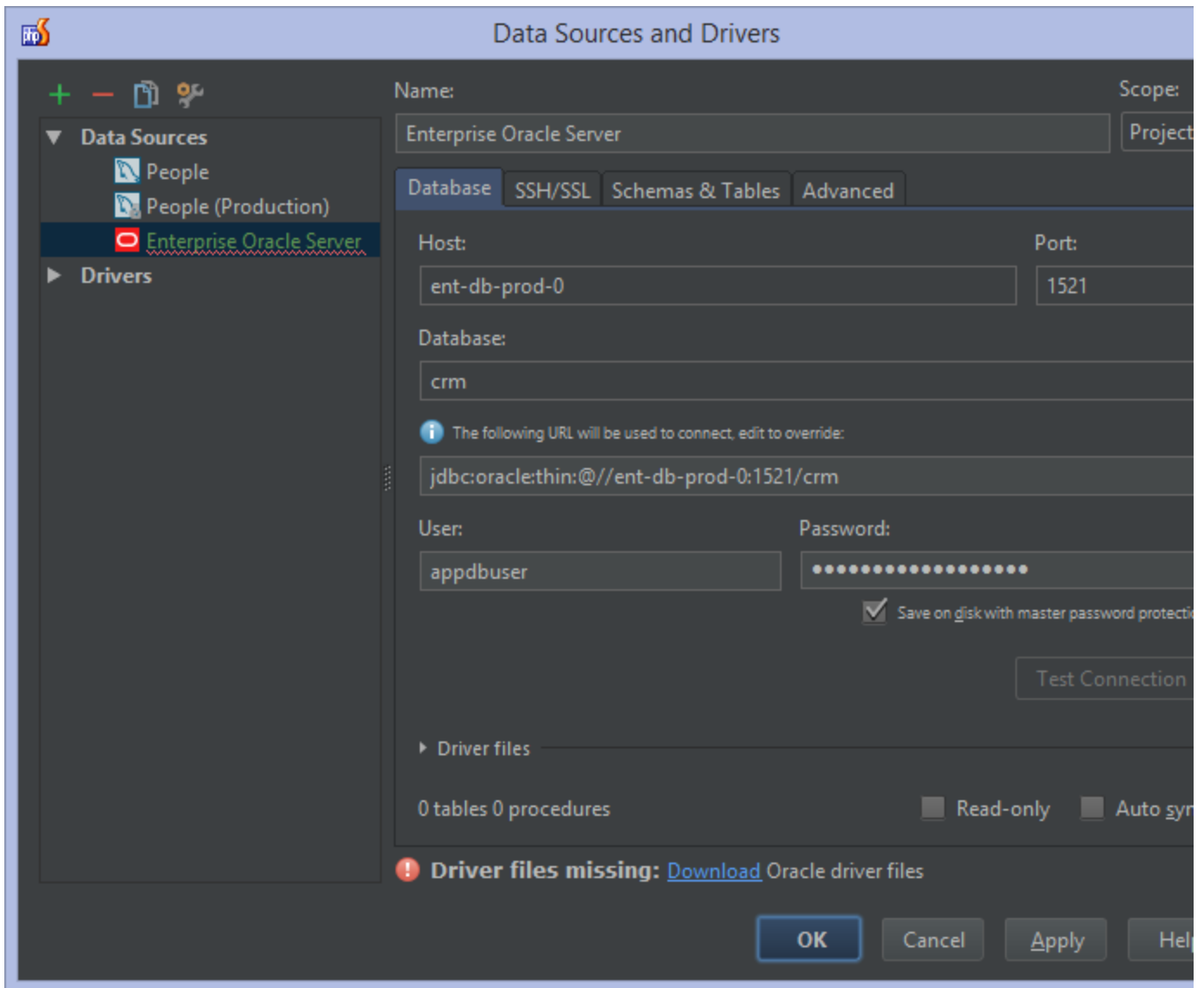
## Setting up a database connection

From the Database Tool window, we can use the green + icon in the toolbar or press Alt+Insert (CMD-N on Mac OS X) to create a new data source. A data source can be a "real" data source, connected to a database system, or a "DDL" data source which is a collection of files that describe tables, indexes and such and can be exported to a database server later on.

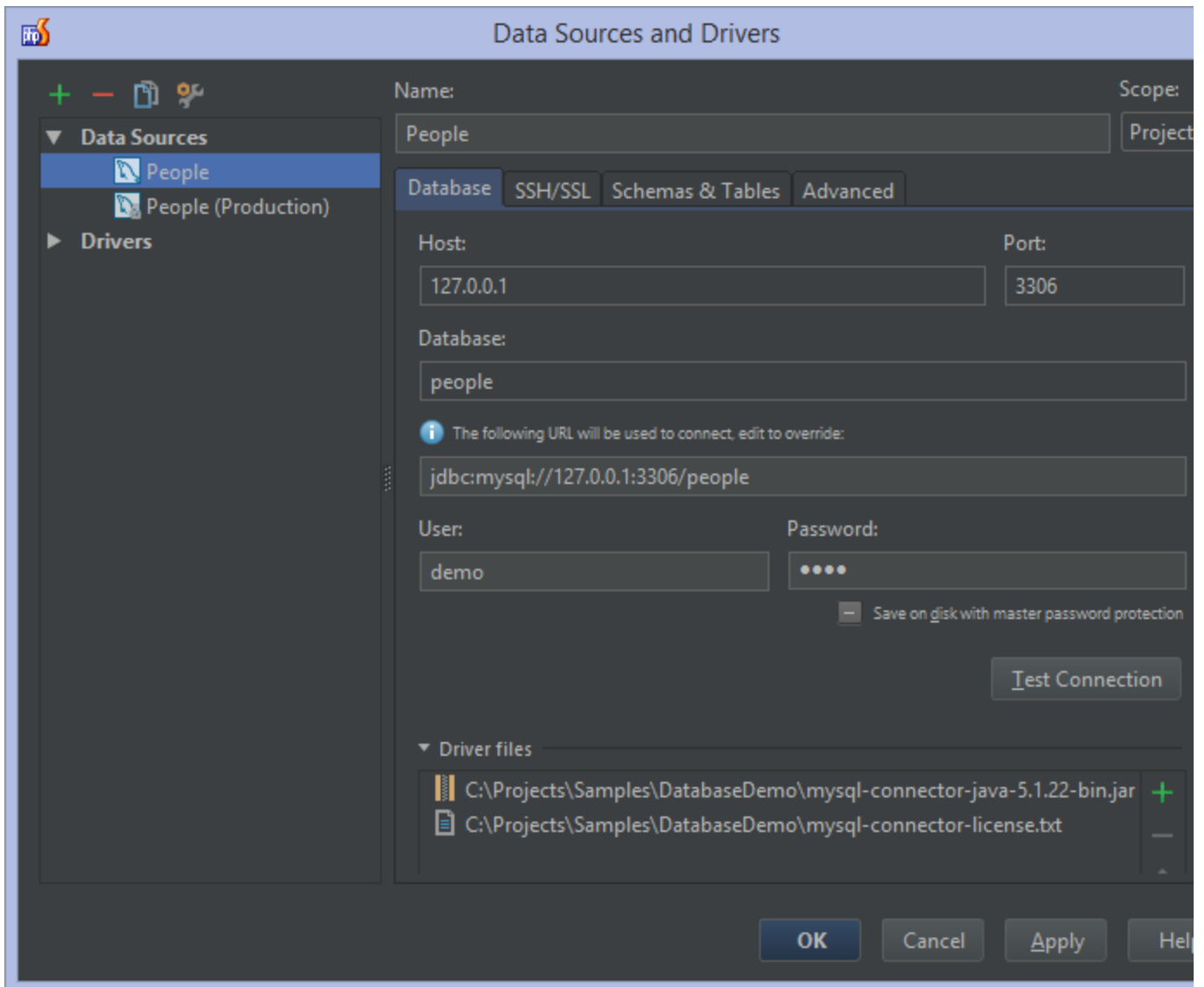


To connect to a database, we can first pick the type of database. We can connect to Google Cloud SQL, DB2, Derby, H2, HSQLDB, MySQL, an ODBC connection, Oracle, PostgreSQL, SQL Server (and Microsoft Azure Database), or SQLite an Sybase.

For every database system, different options will be available for configuring the connection. Typically these will be the server, database name, username and password, but many drivers provide additional options for configuring the connection. PhpStorm does not ship with all database drivers installed, but it does provide a handy way of downloading them when needed: click the "Download ... driver files" link next to the warning about missing drivers to download them.

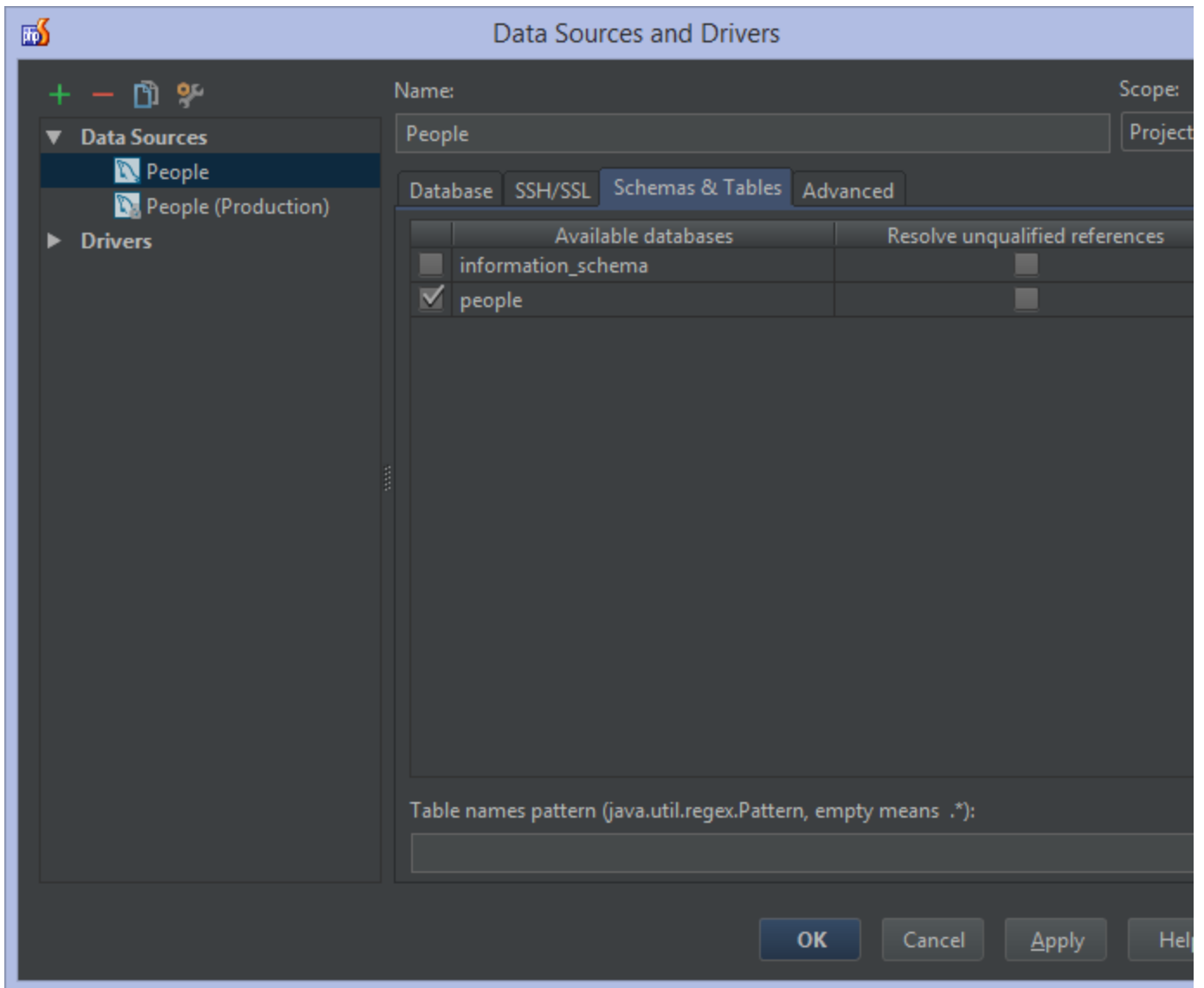


Here's another example for MySQL connecting to a local development database:

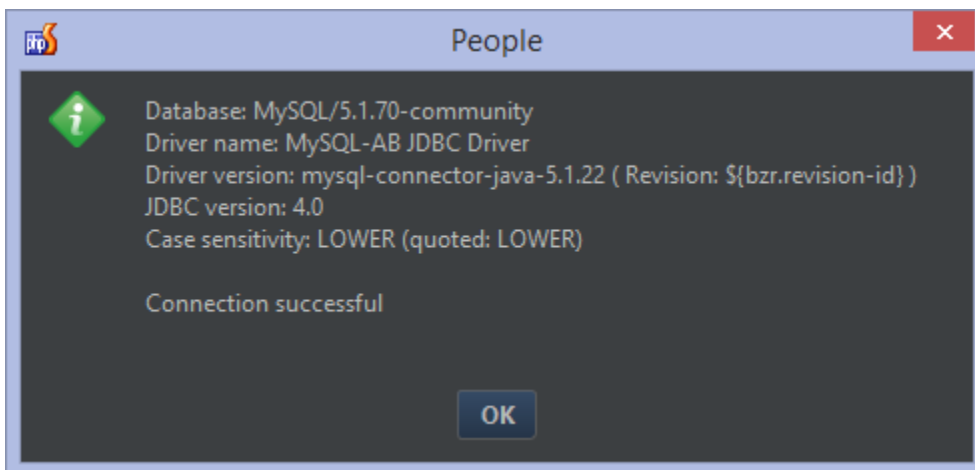


- ✔ For most drivers, we can just enter the connection details, and PhpStorm will generate the underlying database connection string for us. For some, we may have to enter a full JDBC connection string. Try searching the Internet for "<driver/database type> jdbc connection string" if you encounter such driver.

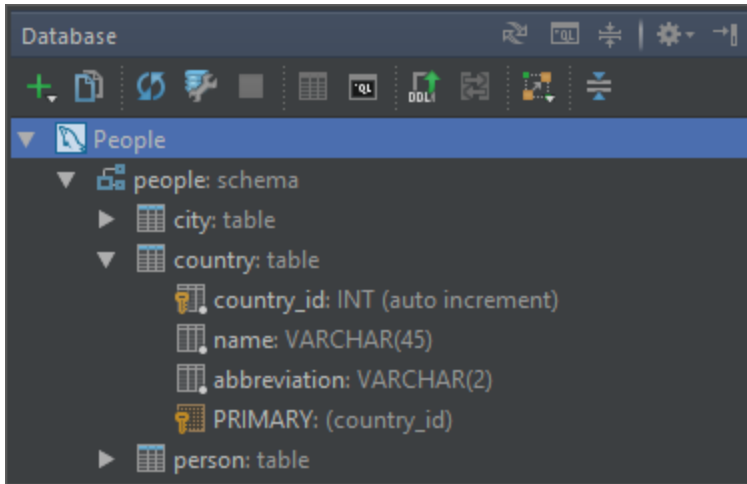
Under the Schemas & Tables tab, we will have to specify which schemas and tables we want to be able to manipulate and generate code completion information for.



To test the settings, click the Test Connection button, which will tell us if it succeeded (or not) in connecting to the database.



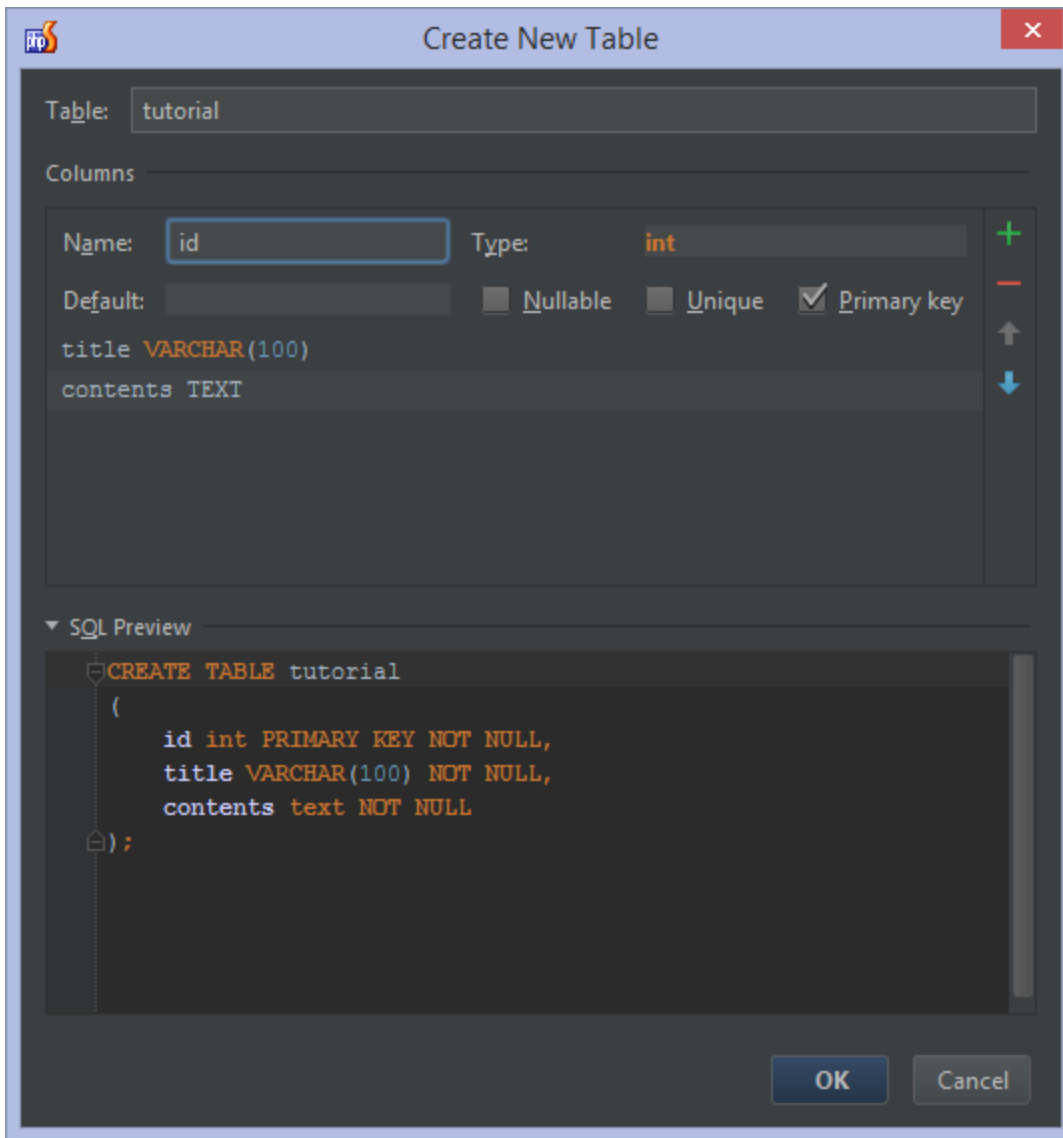
When the connection succeeds, we can save our settings. This will also trigger PhpStorm to download schema information from the database and display it in the database tool window. We can expand schemas, tables, columns and indexes and get more details on the database structure.



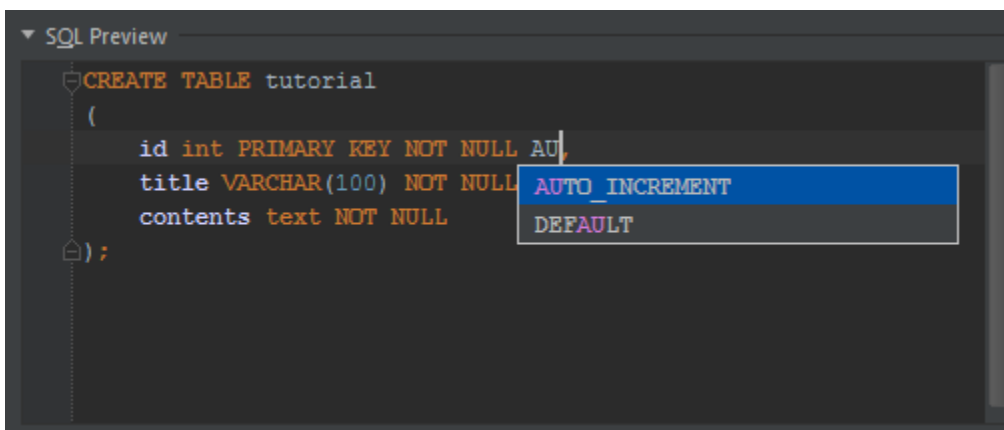
## Refactoring the Database

### Creating a Table

We can use the New | Table context menu (Alt+Insert, or CMD-N on Mac OS X) to create a new table. A dialog will open in which we can give the new table a name and specify the columns that should be created.



Columns can be edited from the UI and PhpStorm will generate the DDL code for us. However, if we wanted to add additional statements to the DDL, we can do so by editing them in the SQL preview editor. Completion is available by pressing Ctrl+Space. Let's make our id column auto increment (in the MySQL dialect).

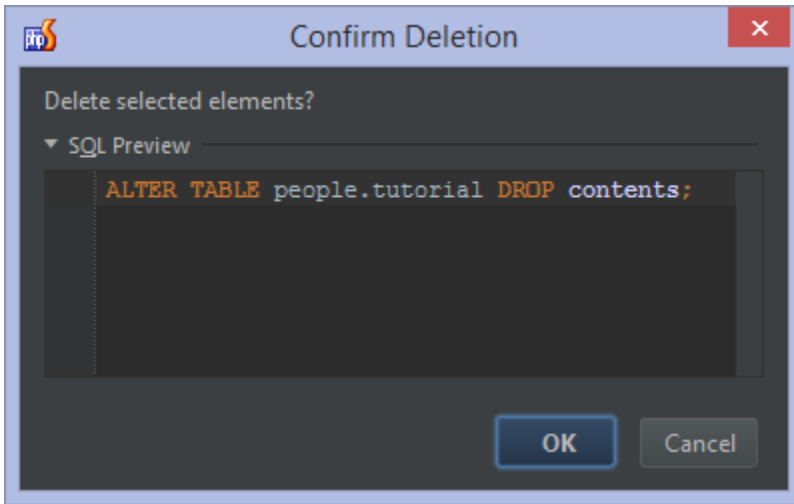


Clicking OK will create the table in our database.

## Deleting a Table, Index or Column

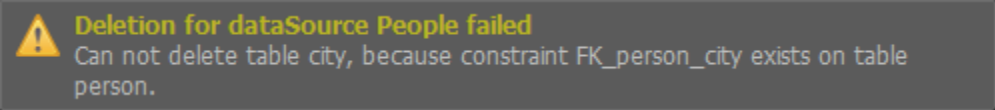


To delete a table, index or column from our database, we can navigate to it and press Delete (or use the Delete context menu). This will show us a preview of the SQL statement that is about to be executed. It could be a DROP TABLE, an ALTER TABLE, or something similar that gets generated.



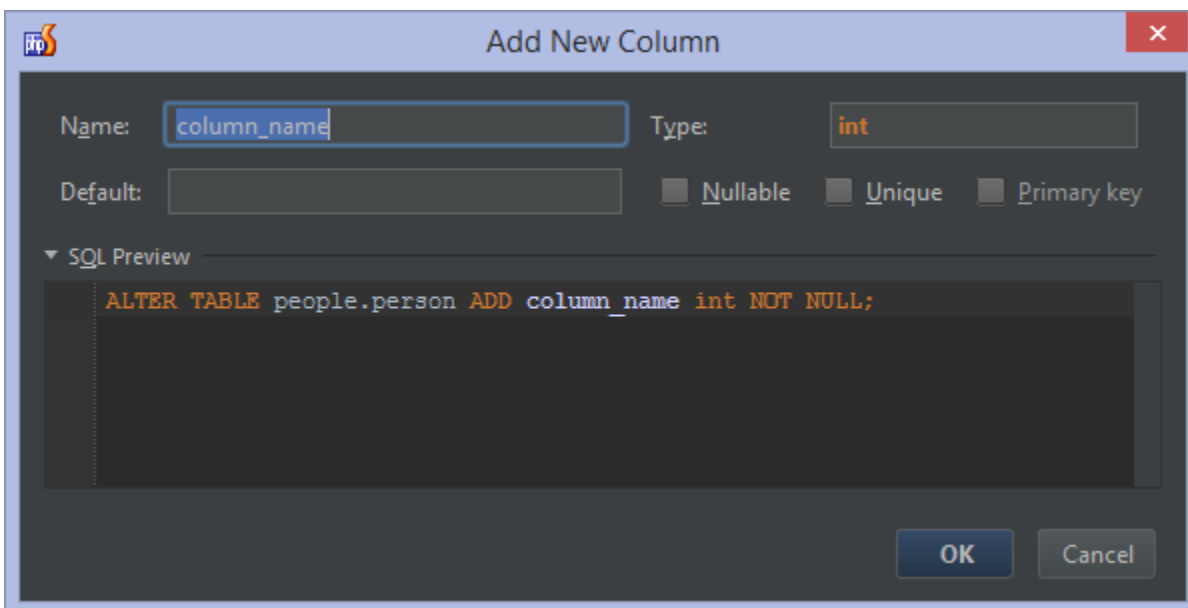
Clicking OK will perform the change.

- ✔ If we attempt to delete a table or column that is referenced in an index or foreign key constraint, the IDE will warn us about this. We will have to manually delete the reference first.



## Adding a Column to an Existing Table

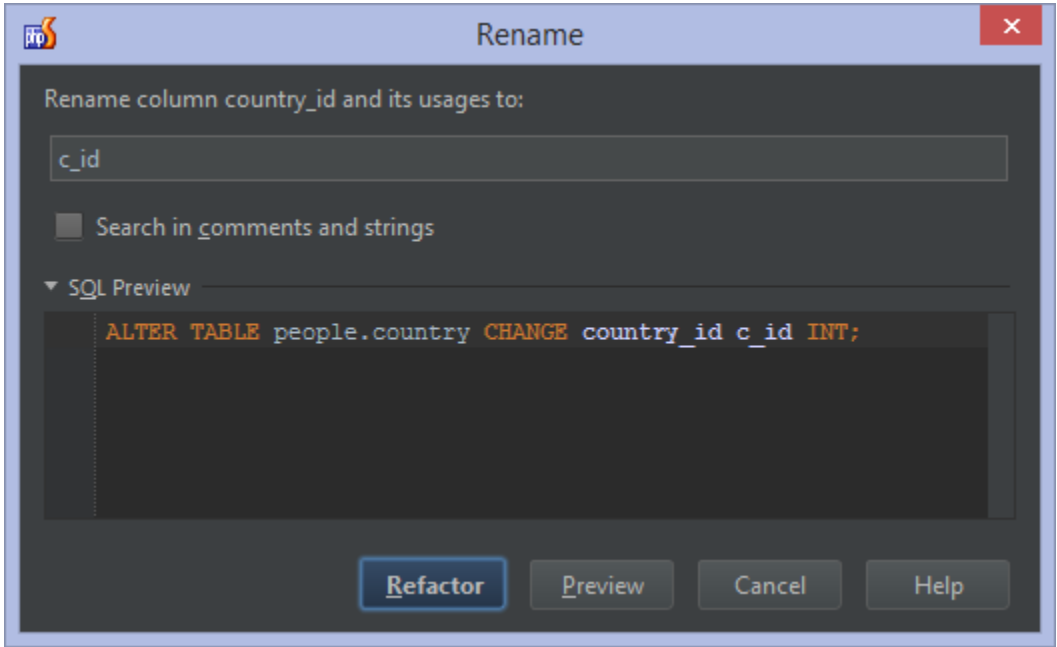
Adding a column to an existing table can be done by selecting the table and using the New | Column context menu or pressing Alt+Insert (CMD-N on Mac OS X). This will give us a UI in which we can generate the ALTER TABLE statement. Clicking OK will apply it to our database.



## Renaming a Table, Index or Column

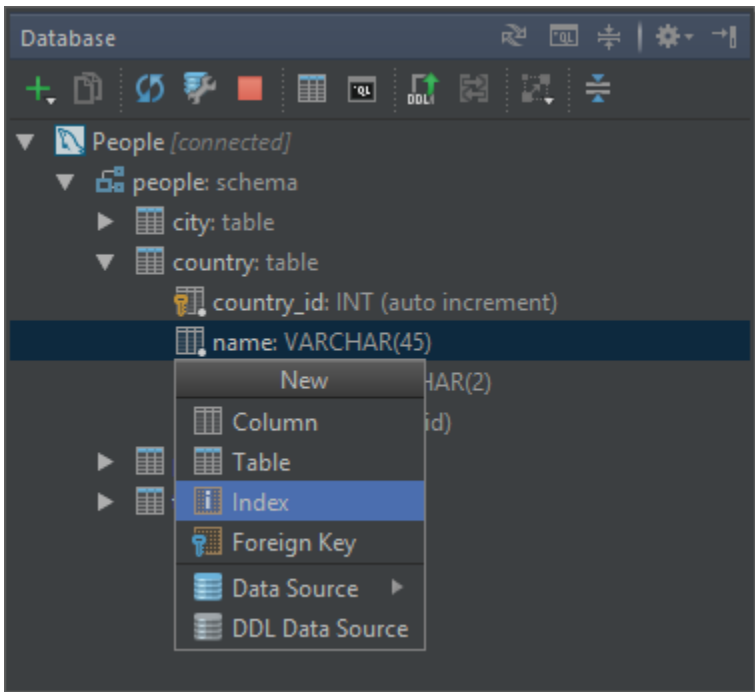
Adding a table, index or column can be done by selecting it and using the Rename refactoring through the Rename... context

menu or by pressing Shift+F6. This will let us rename the selected item. We can click Refactor to perform the rename immediately, or first click Preview to get an overview of the changes that will be carried out.



## Creating Indexes and Foreign Keys

Indexes and foreign key constraints can be added by selecting a column and using the New context menu or pressing Alt+Insert (CMD-N on Mac OS X).

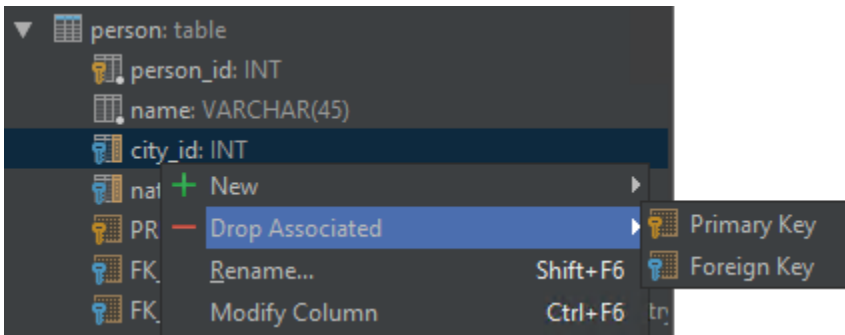


We can name our index and optionally change the generated DDL before clicking OK to add the index or foreign key to the database.



## Deleting Indexes and Foreign Keys

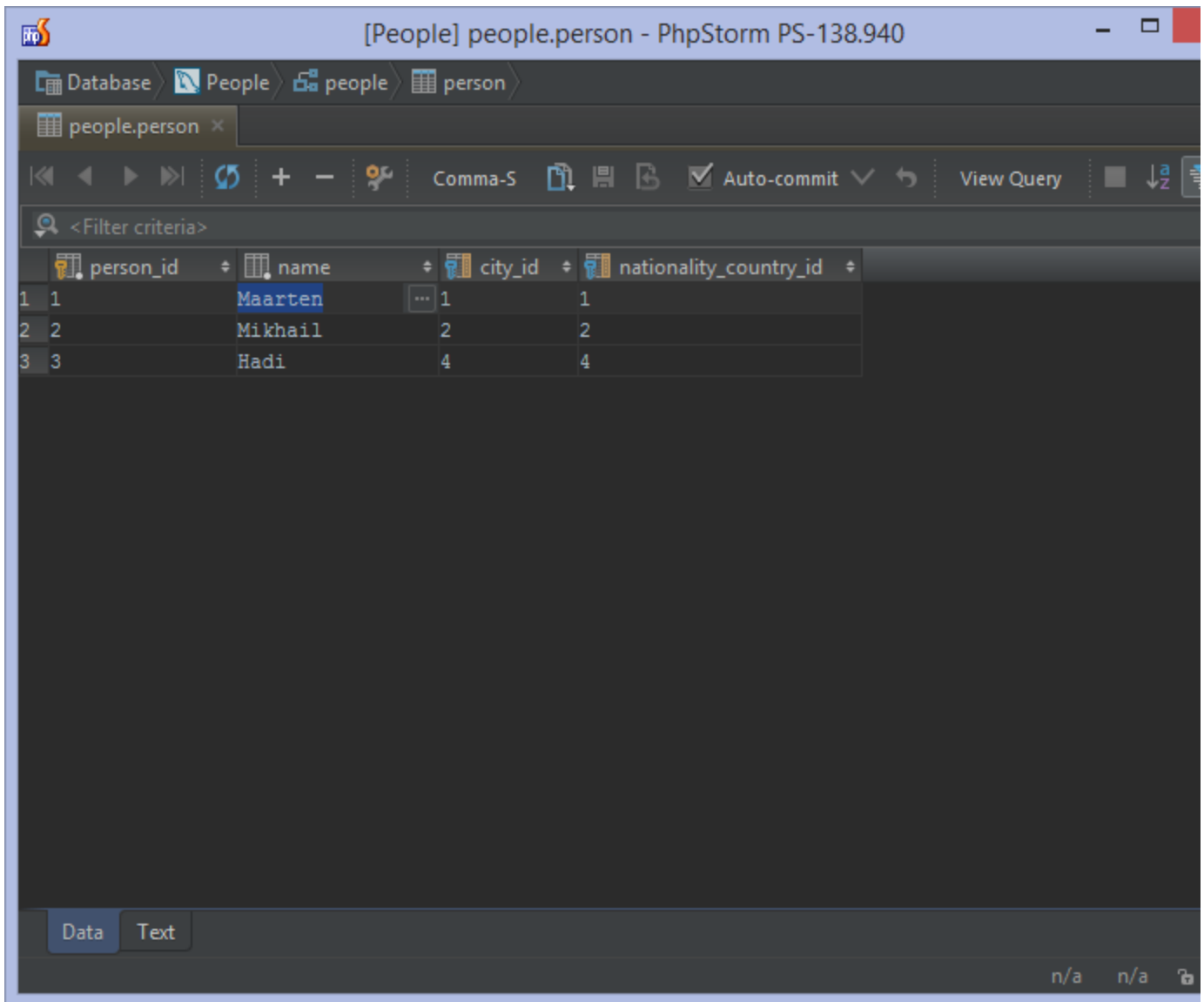
To delete an index or foreign key, we can find it in the database tool window and press Delete. Another option is to find the column for which we want to delete the index or foreign key and use the Drop Associated context menu instead.



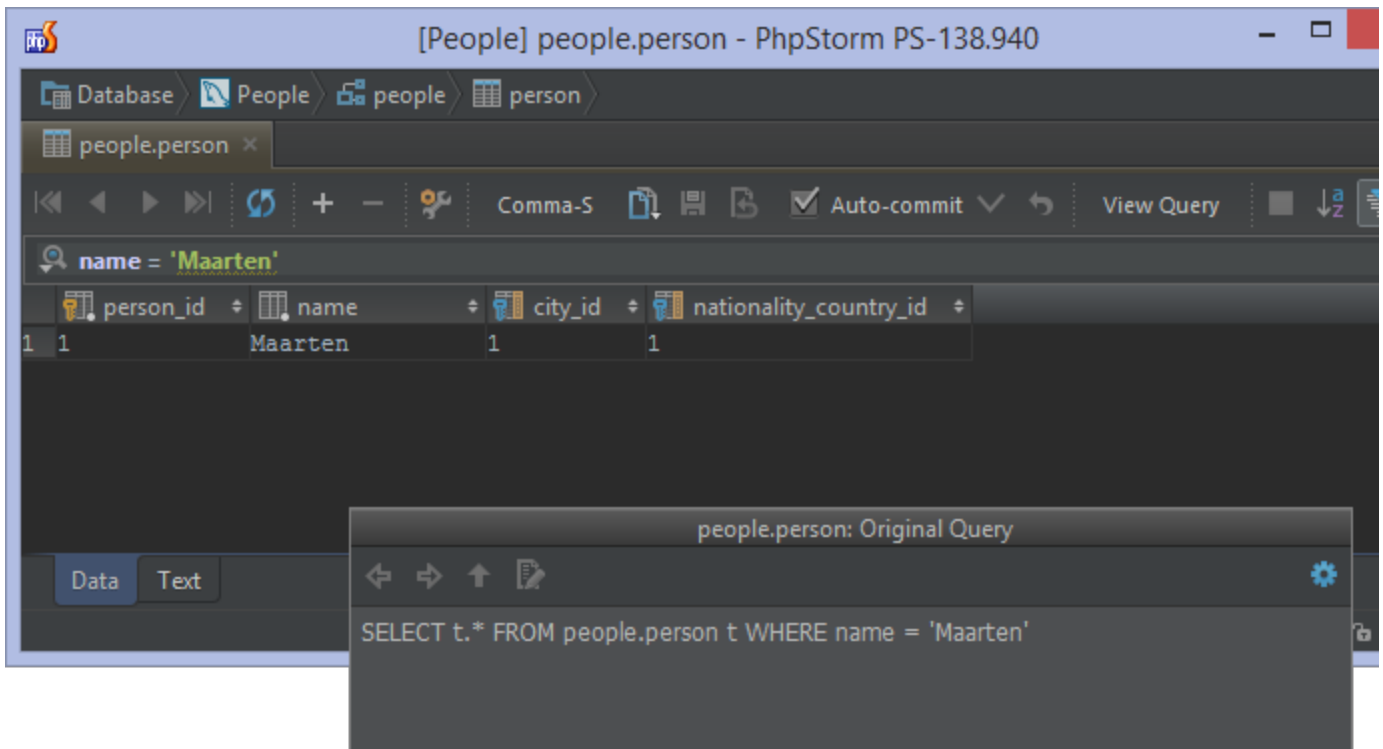
## Querying, Inserting and Updating data

### Using the Table Editor

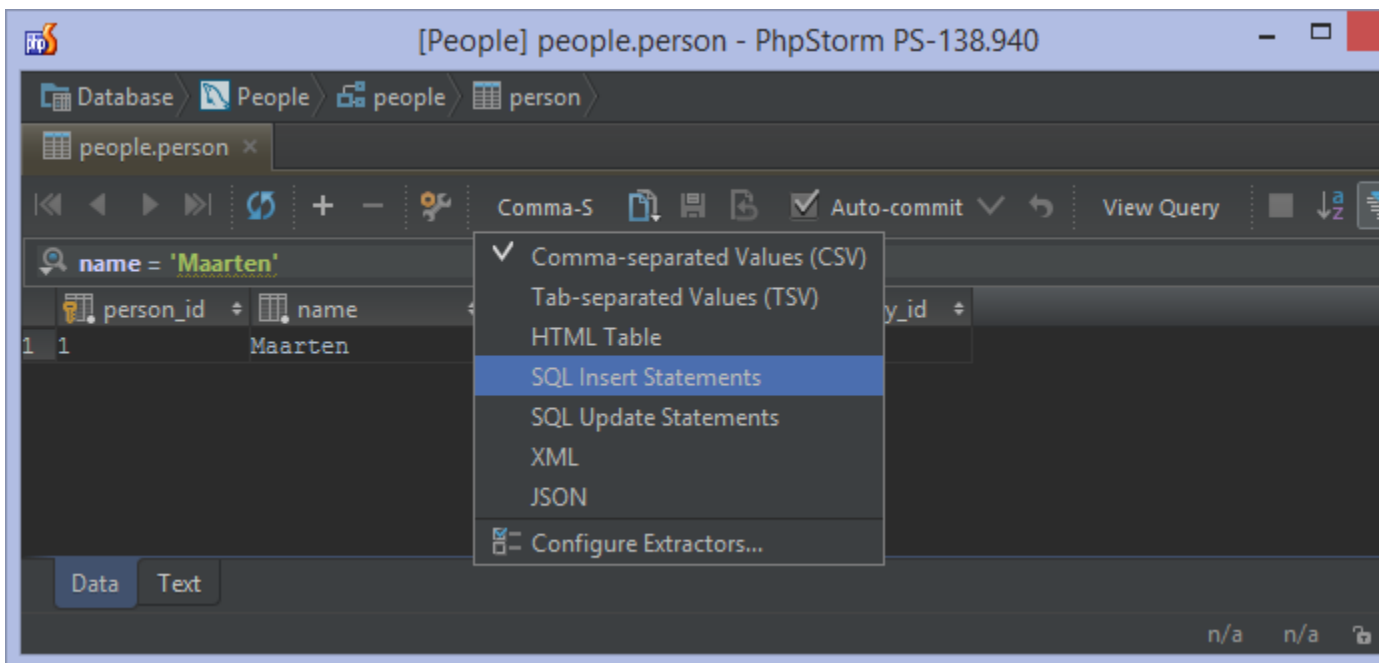
After selecting a table, we can bring up the Table Editor using the context menu or by pressing F4. From the Table Editor, we can view the data in the table and add/edit/remove values. Clicking column headers will sort the data on that column. If the database system supports transactions, we can enable/disable the Auto-commit option. Disabling it will enforce the use of transactions and require us to explicitly commit edits, inserts and deletes.



We can also filter data by adding a filter statement (use Ctrl+Space for completion on columns). Clicking View Query will show the resulting query.

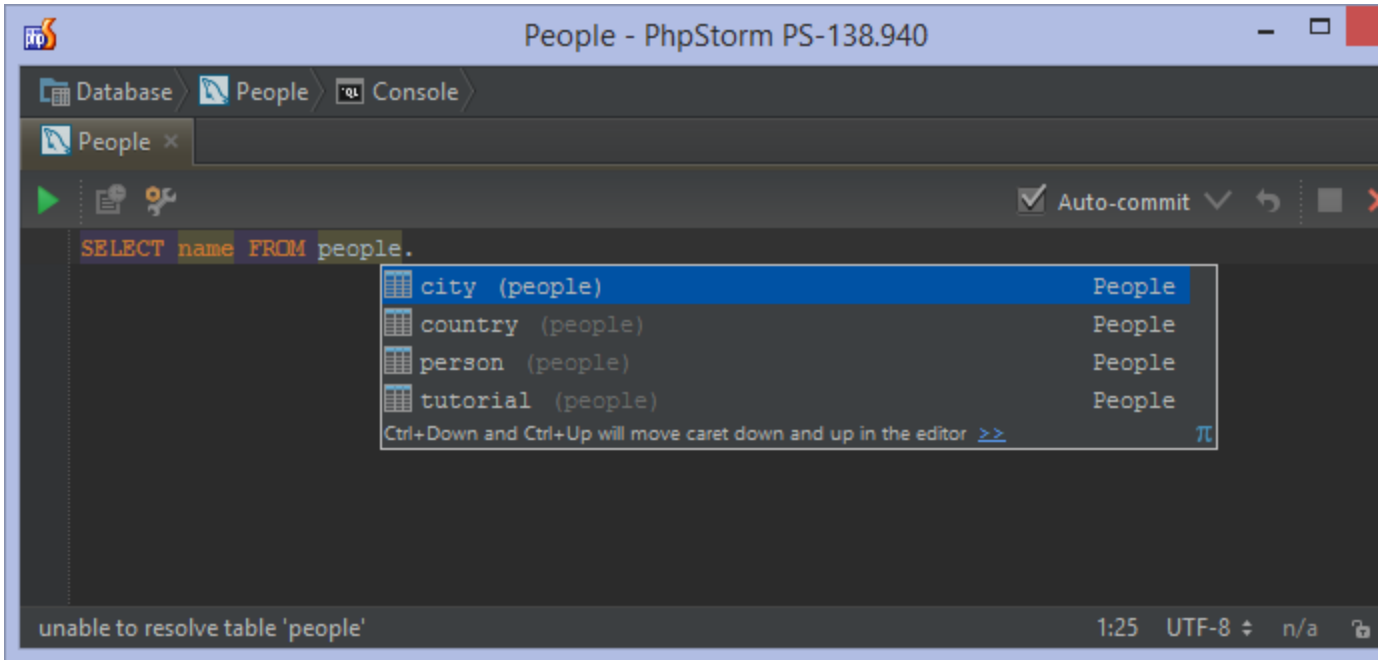


The Table Editor as well as the Database Console support exporting data to the clipboard or a file. From the toolbar, we can select the desired output format, such as Comma-Separated Values (CSV), XML or JSON. We can also generate INSERT and UPDATE statements from the data so we can easily import it into a different database afterwards.

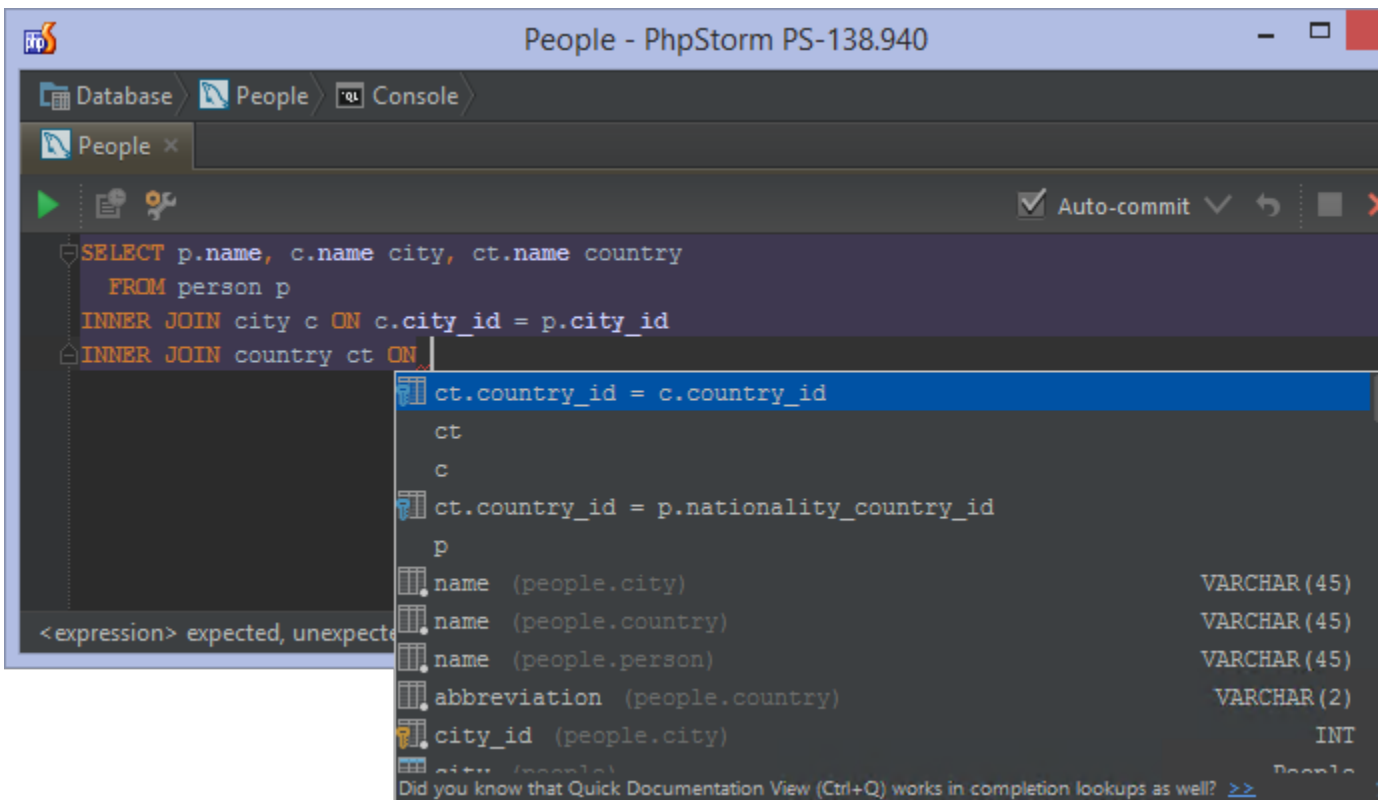


## Using the Database Console

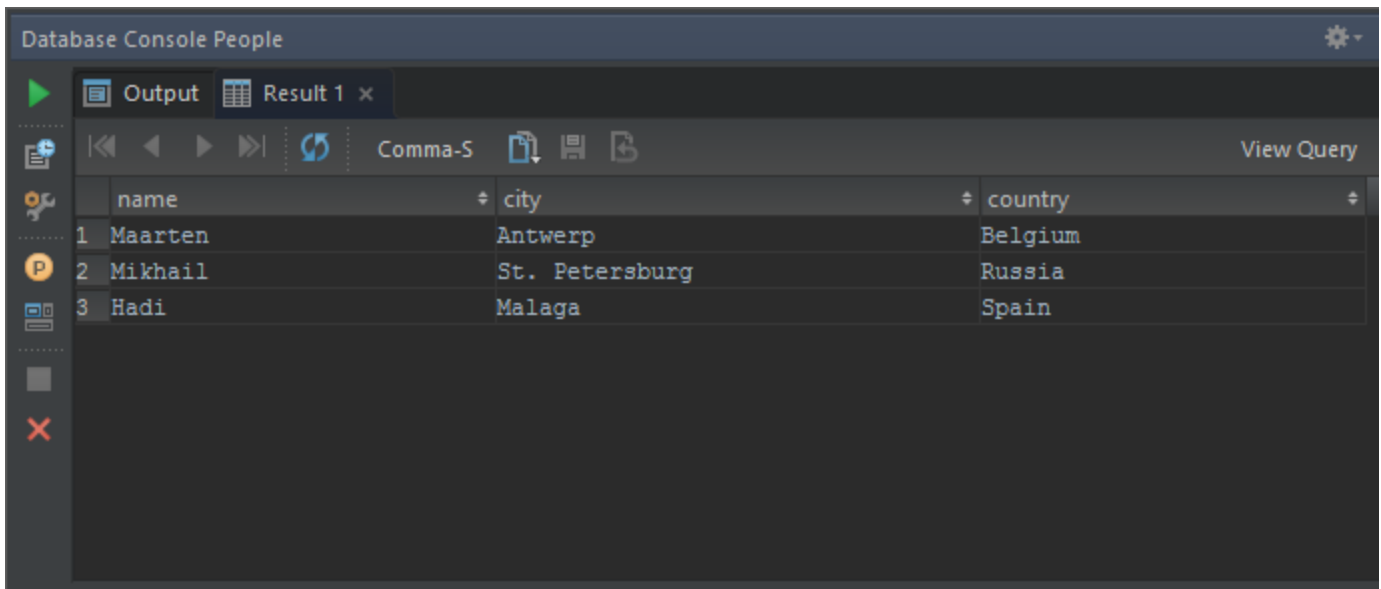
While the Table Editor gives us a number of options for working with data, it's limited to just one table and does not allow us to run an arbitrary SQL statement. The Database Console is a query editor in which we can write any SQL statement and run it. We can open the Database Console by pressing Ctrl+Shift+F10 (Shift-CMD-F10 on Mac OS X).



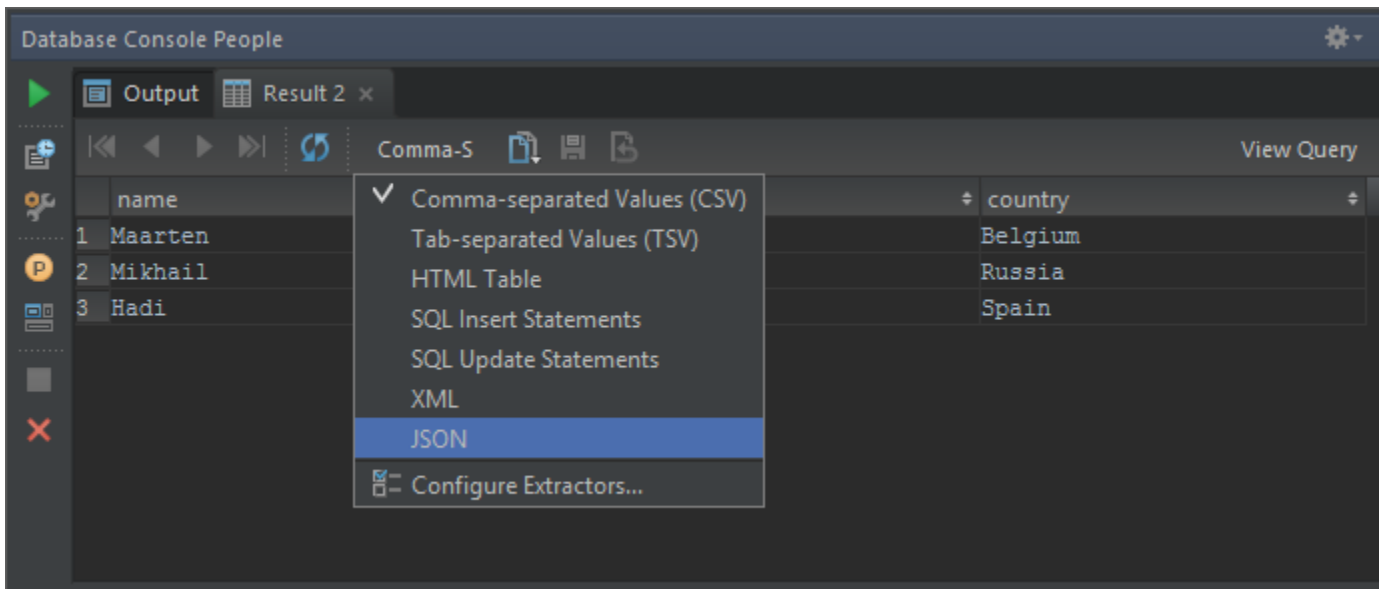
We can enter a query in the console and get syntax highlighting, completion (Ctrl+Space), on-the-fly code analysis, multiple intentions, and navigation. Completion can even figure out foreign key constraints and suggest JOIN conditions.



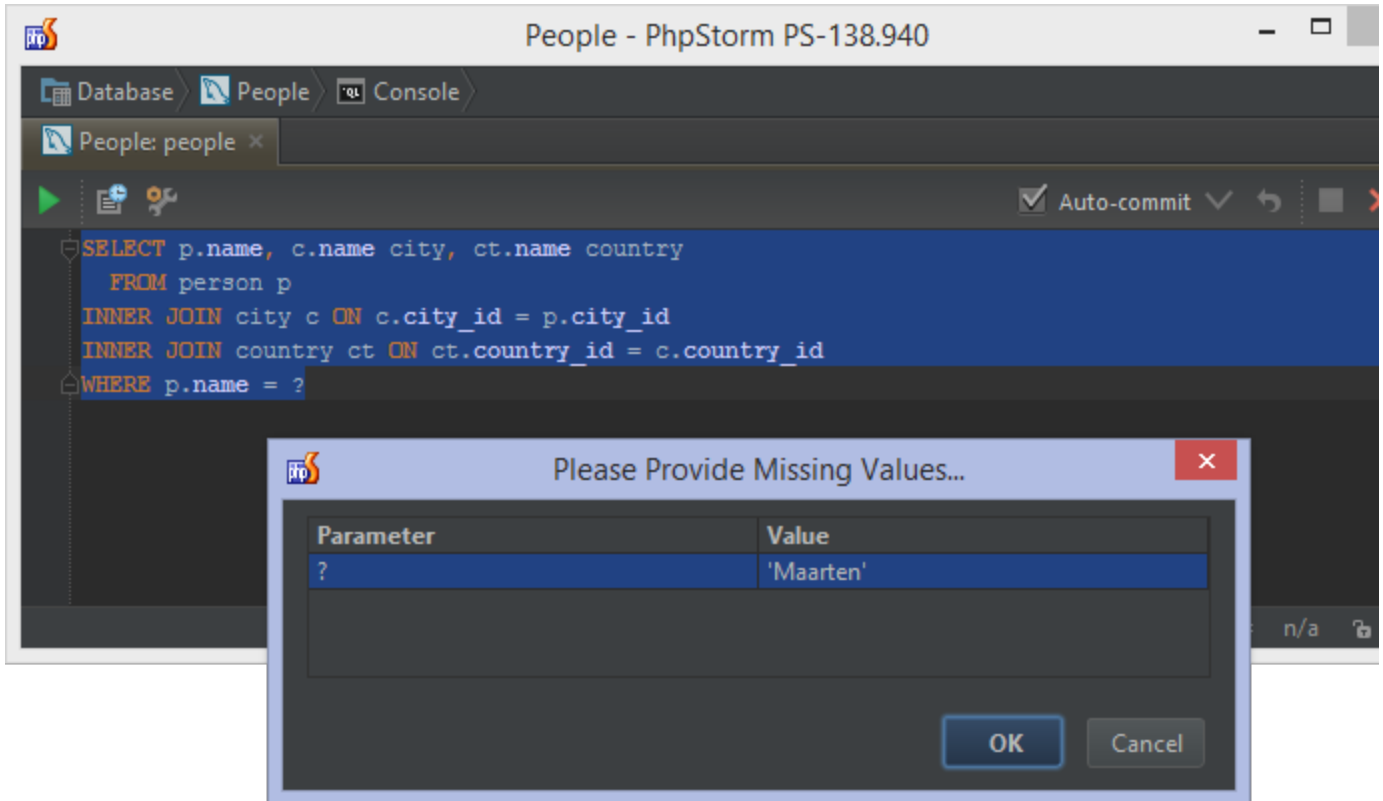
To run the query, we select the statement and press Ctrl+Enter. The result set (or query log) is displayed in a separate tool window.



The Table Editor as well as the Database Console support exporting data to the clipboard or a file. From the toolbar, we can select the desired output format, such as Comma-Separated Values (CSV), XML or JSON. We can also generate INSERT and UPDATE statements from the data so we can easily import it into a different database afterwards.

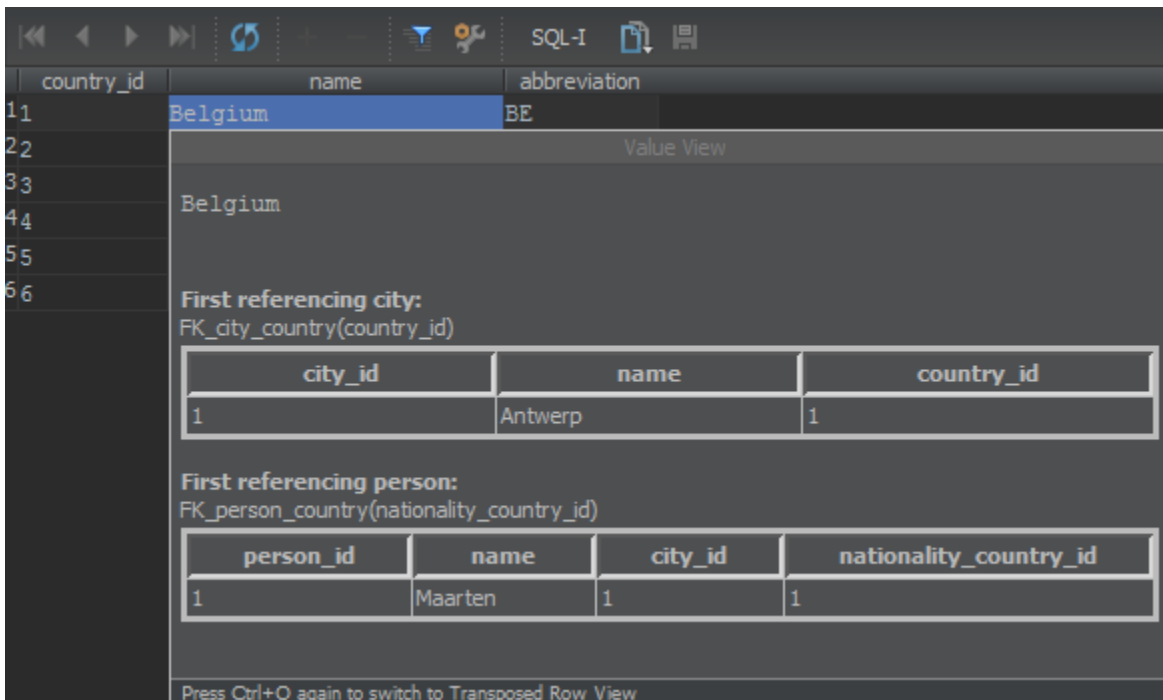


When building queries in the Database Console, we can make use of variables and substitute them when running the query. By default, variables are recognized as strings similar to `?`, `:variable`, `@variable`, `#variable#` or `$variable$`. By using variables, PhpStorm will prompt for the values to use in the query during execution.



## Value View / Single Record Transpose View

When browsing a table, we may want to see an example of where the selected value is referenced. Rather than [generating a database diagram](#), we can instead press Ctrl+Q (Ctrl-J on Mac OS X) and get an example of how a given row is used throughout the database.



If we press the keyboard shortcut again, we get a transposed view of the selected row. When we work with a table that has many fields and requires vertical scrolling, the Single Record Transpose View makes it easier to view the data by flipping the column and row axis.



country_id	name	abbreviation
1	Belgium	BE

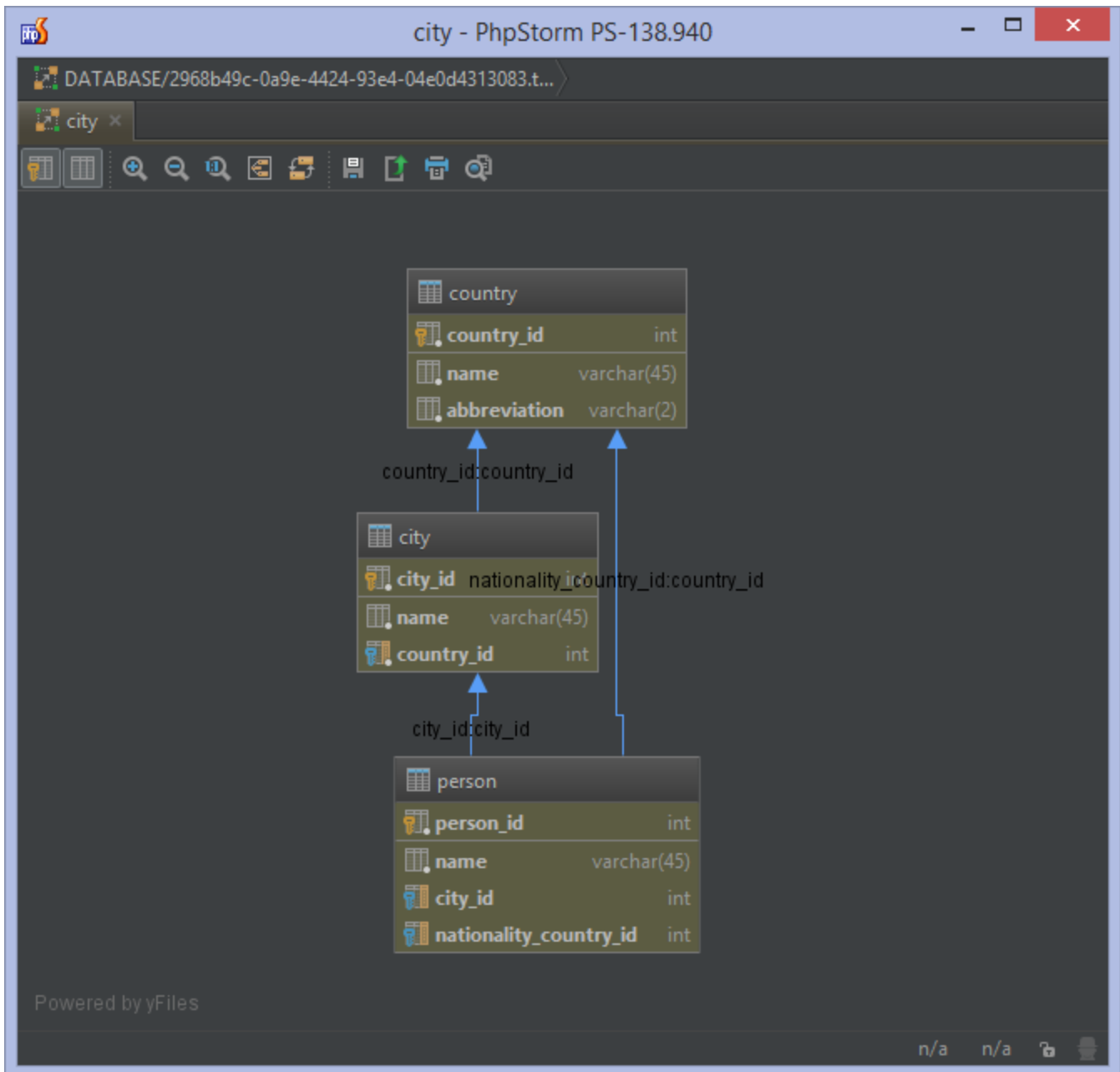
Transposed Row View

Name	Value
country_id	1
name	Belgium
abbreviation	BE

Press F4 to navigate to related data

## Generating a UML database diagram

If we have multiple tables with foreign keys between them, we can easily generate a database diagram. Select the tables to be included in the diagram and use the Diagrams | Show Visualization... context menu or press Ctrl+Alt+Shift+U (Alt-Shift-CMD-U on Mac OS X). PhpStorm will then generate a database diagram for these tables, showing how they relate to each other.



We can export the diagram to a file, add annotations to it, or print it.



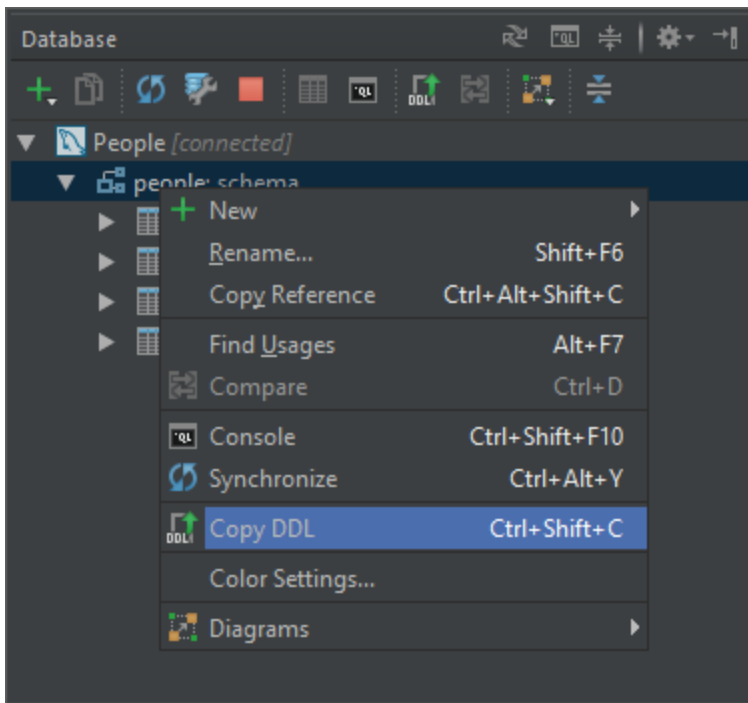
If all we need is a quick overview of the database, Ctrl+Alt+U (Alt-CMD-U on Mac OS X) will show the diagram in a popup instead.

## Database Migration

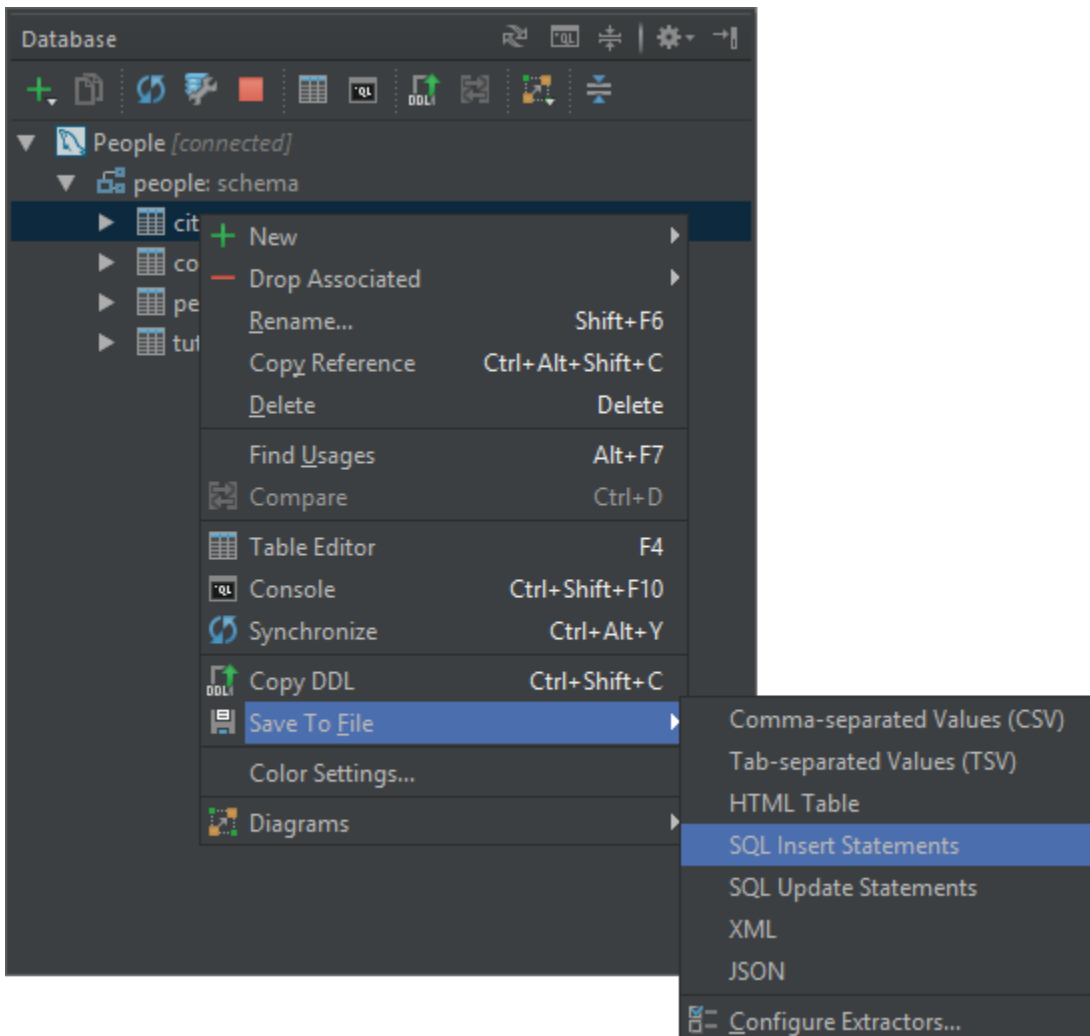
### Exporting Database Tables as SQL Statements

Often, we want to be able to export databases so we can import them in other databases. For example: copying the schema of a newly created table on our development machine to a test server, or vice versa.

PhpStorm allows exporting tables as (DDL) SQL statements: using the Copy DDL context menu on one or more tables scripts the tables, columns, indexes and foreign keys to the clipboard so we can paste them in database console.

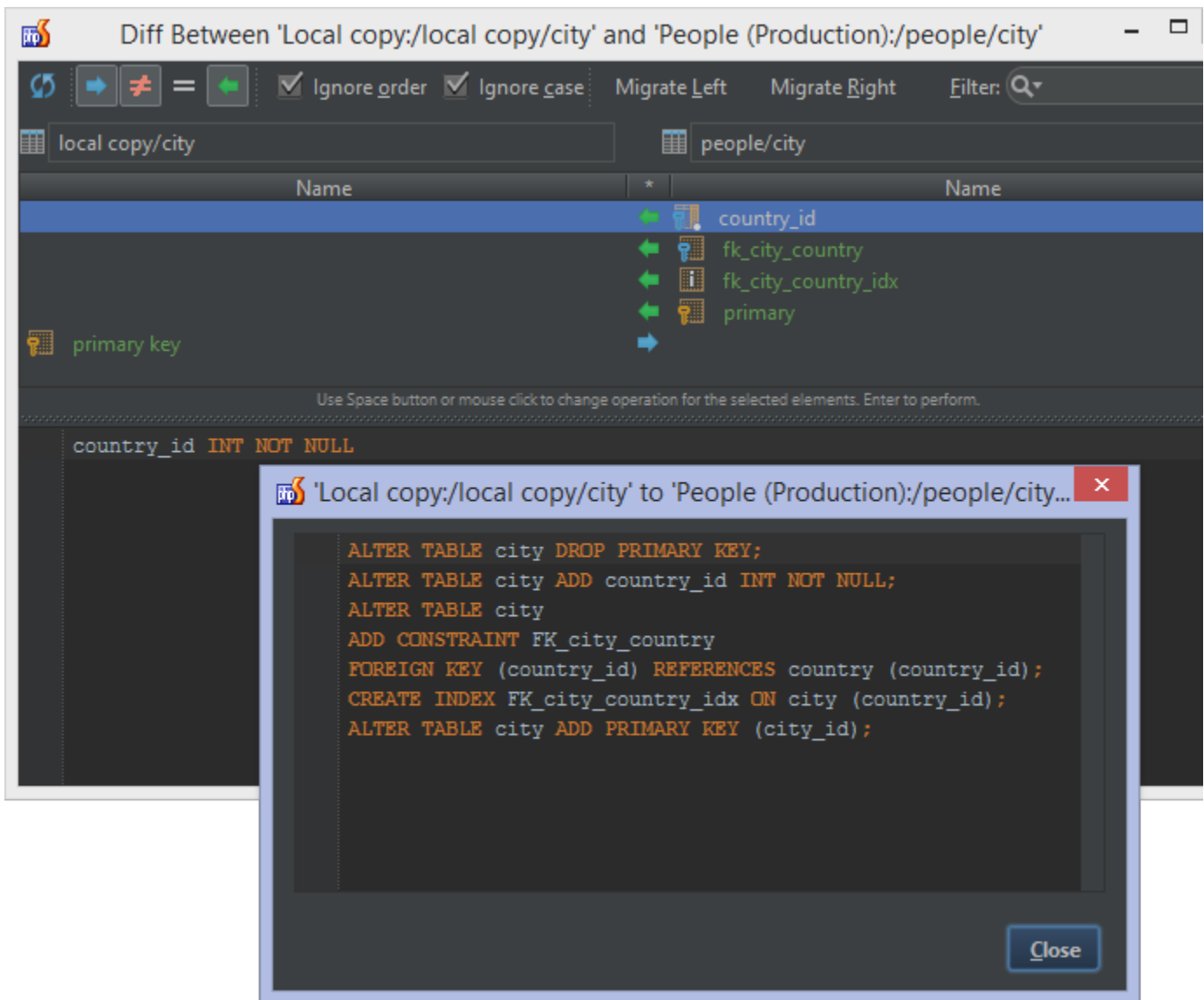


If we also need the contents of the various tables, we can use the Save To File | SQL Insert Statements context menu and export the data of the table(s) as well.



## Comparing and Migrating Database Code

When we have several data sources configured, for example a production database and a local copy, we can select both of them (or just select two specific tables in each) and use the Compare context menu (Ctrl+D, or CMD-D on Mac OS X). This will generate a "diff" of both data sources or tables and allows us to generate a migration script.



Above, we can see our local database table does not have the `country_id` column and related primary and foreign keys. Clicking Migrate Left will generate a script that would bring our local database table to the latest version. Migrate Right would do the opposite.

## Database Code Completion while working in PHP

While writing our code, we can get all the features of the Database Console inside PHP, JavaScript, HTML or other languages. We can just start writing our SQL statement and after a while (typically right after the FROM), PhpStorm will provide us with code completion (Ctrl+Space). If completion is required earlier, for example to be able to complete column names, we can place the cursor in the string literal, write "SELECT", press Alt+Enter, and select the language to use (MySQL, for example).

```
<?php
$result = mysql_query('SELECT name FROM pe'); // TODO: get all persons
while ($row = mysql_fetch_assoc($result)) {
    $people[$row['id']] = $row['name'];
}
```

We will get completion on JOIN statements, too!

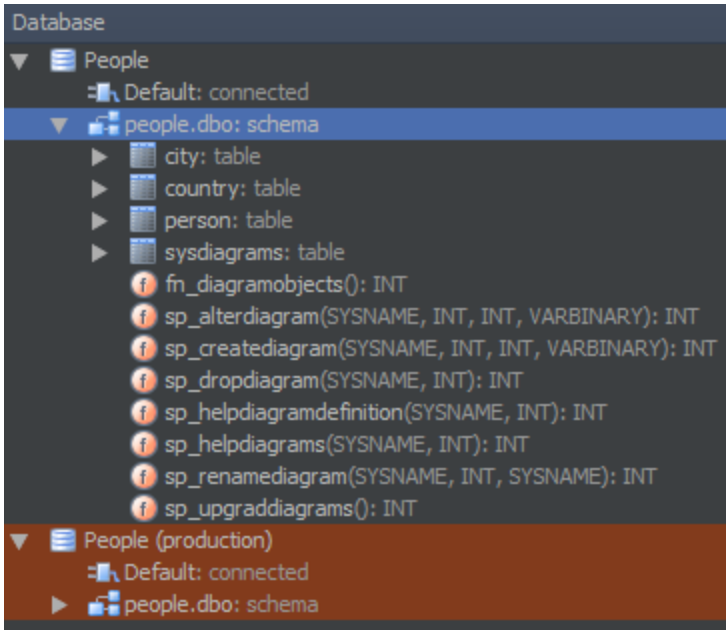
```
$query = "select person.name, city.name city, country.name nationality from person
inner join city on city.city_id = person.city_id
inner join country on ";
```

After selecting (part of) a query, we can press Alt+Enter+\* and edit the MySQL fragment, or pick the Run Query in Console entry to do exactly that: copy the query into a Database Console and run it there.

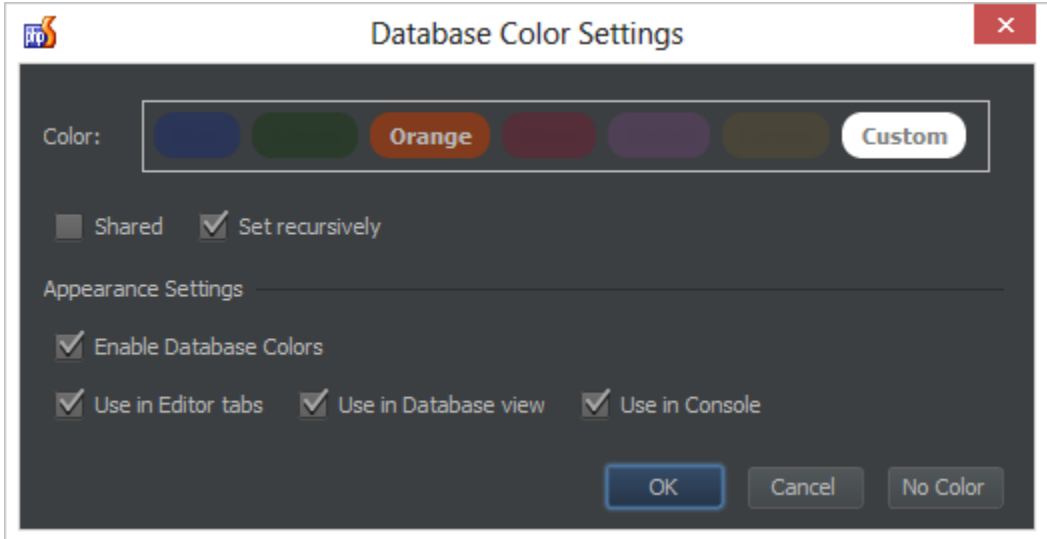
```
$query = "select person.name, city.name city, country.name nationality from person
inner join city on city.city_id = person.city_id
inner join country on country.country_id = person.nationality_country_id
where country.abbreviation = 'BE' ";
```

## Coloring Data Sources

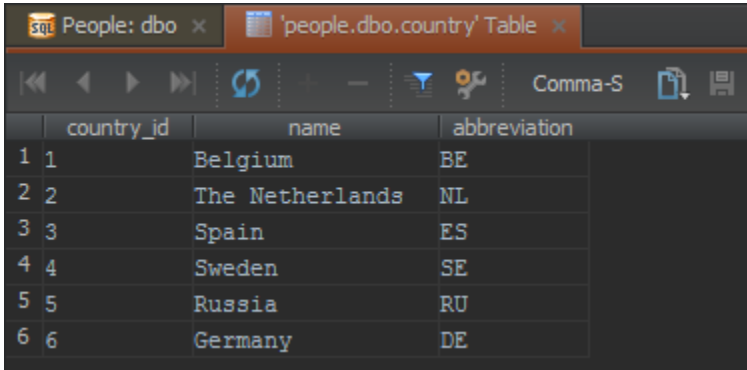
When working with multiple data sources or data sources that have multiple schemas, we can assign different colors to each of them. For example, when working with a development and production database at the same time, it can be useful to give both a different color to have a visual clue about the database we are currently working with.



Colors can be enabled and assigned by using the Color Settings context menu on a data source, schema or object.



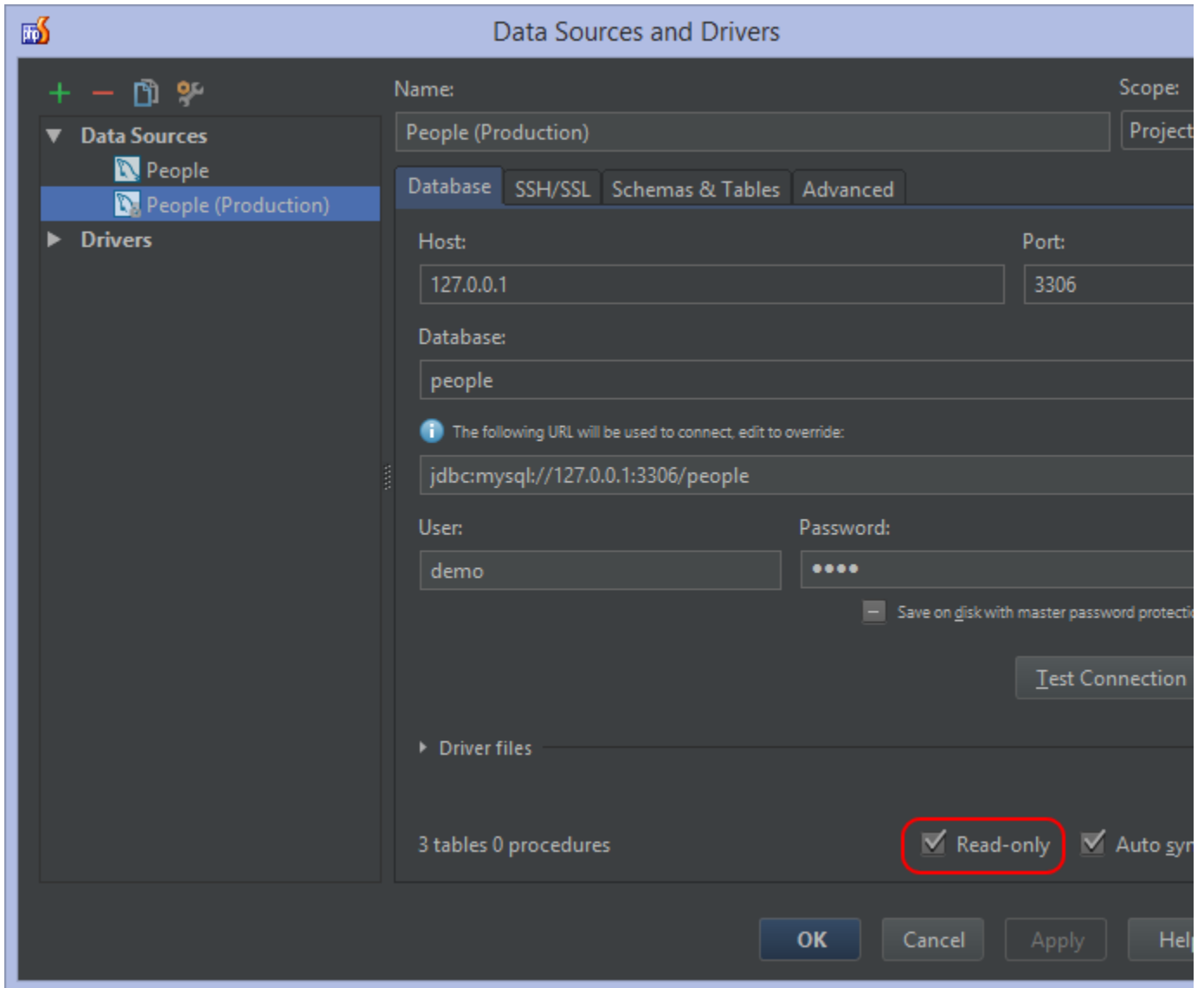
When coloring is enabled, we can clearly distinguish the data source from the tab header by its color. Note that PhpStorm also displays the current database schema in the tab title.



## Read-Only Data Sources

When creating or editing Data Source properties, we can mark it as read-only. This will prevent us from accidentally editing or

removing data or elements from our schema and only provides read access to the data source, even if the user connecting to the database has write privileges.



[Tweet](#)