Writing and Executing SQL Statements

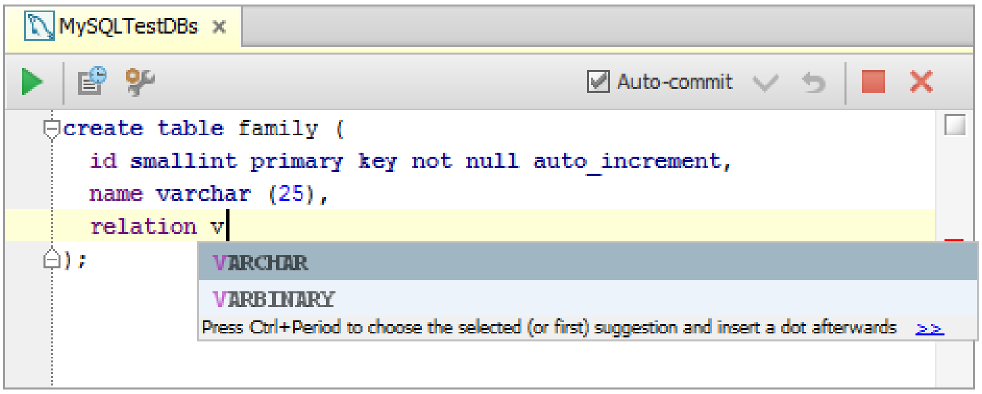
Use the input pane of the database console to compose and execute your SQL statements, and also to perform other, associated tasks. (The input pane is shown as a tab in the editor.)

- Composing SQL statements
- Navigating to a table or column view in the Database tool window
- Executing an SQL statement
- Executing parameterized statements
- Executing a group of statements
- Using the error notification bar
- Executing a part of a statement (e.g. a subquery)
- Executing auto-memorized statements
- Managing database transactions

Composing SQL statements

When composing your SQL statements, use:

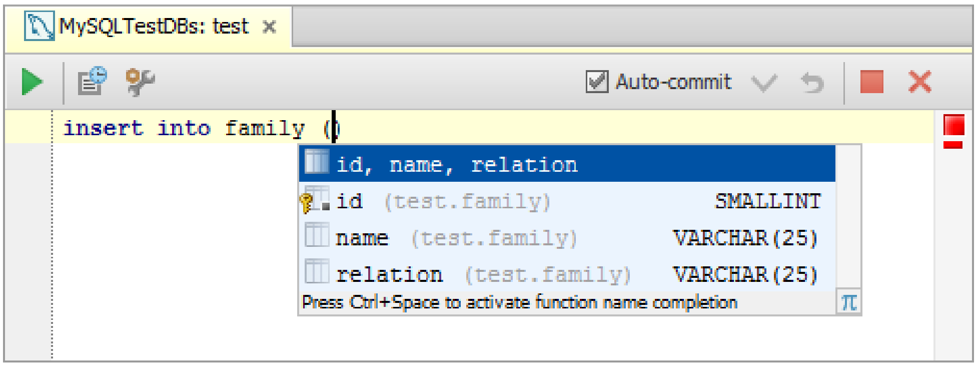
- Auto-completion and highlighting of SQL keywords, and table and column names.



```
create table family (
  id smallint primary key not null auto_increment,
  name varchar (25),
  relation v
);
```

VARCHAR
VARBINARY
Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >>

- Context-sensitive suggestions (a.k.a. basic code completion) activated by means of `Ctrl+Space` or **Code | Completion | Basic**.

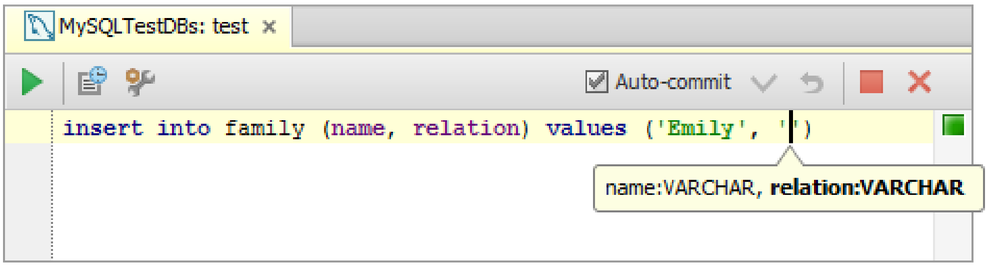


```
insert into family ()
```

```
id, name, relation
id        (test.family)        SMALLINT
name      (test.family)     VARCHAR(25)
relation  (test.family)     VARCHAR(25)
```

Press Ctrl+Space to activate function name completion

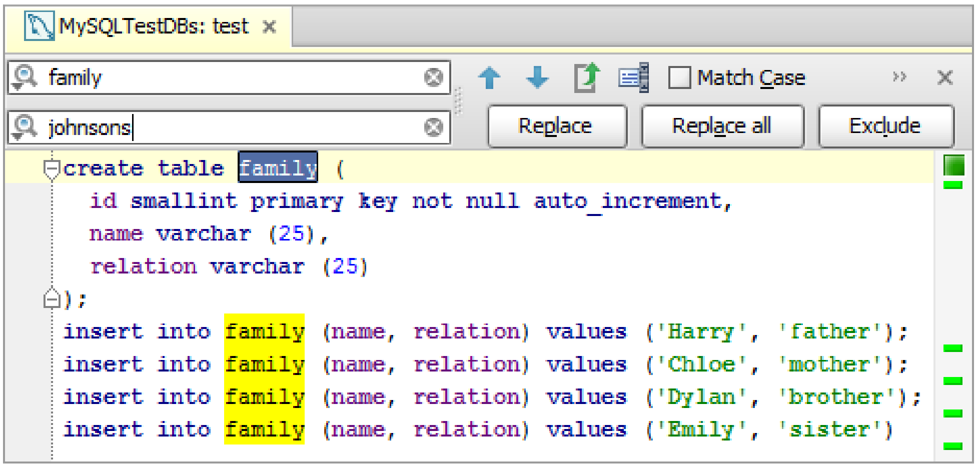
- Data type prompts for columns (fields) invoked by `Ctrl+P` or **View | Parameter Info**.



```
insert into family (name, relation) values ('Emily', '')
```

name:VARCHAR, **relation:VARCHAR**

- Advanced find and replace capabilities (`Ctrl+F` or `Alt+F3` or **Edit | Find | Find**, and `Ctrl+R` or **Edit | Find | Replace**).



```
create table family (
  id smallint primary key not null auto_increment,
  name varchar (25),
  relation varchar (25)
);
insert into family (name, relation) values ('Harry', 'father');
insert into family (name, relation) values ('Chloe', 'mother');
insert into family (name, relation) values ('Dylan', 'brother');
insert into family (name, relation) values ('Emily', 'sister')
```

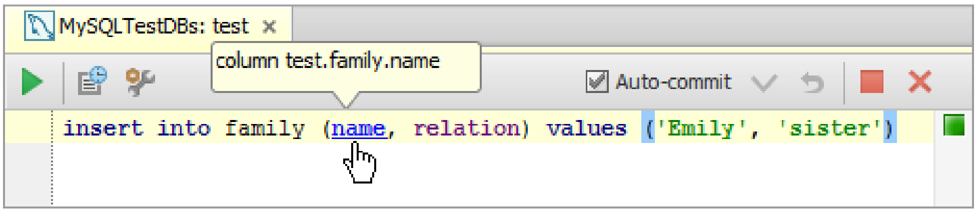
See also, Navigating to a table or column view in the Database tool window.

Navigating to a table or column view in the Database tool window

When composing a statement, it's sometimes useful to take a look at the structure of a table, or to see the info about a column (field) in the context of the table to which it belongs. For such purposes, IntelliJ IDEA provides the ability to switch from a table or column referenced in the input pane to its view in the **Database** tool window.

The following ways are available for using this feature:

- Place the cursor within the name of the table or column of interest. Then use `Ctrl+B`, `Ctrl+Button1 Click` or `Button2 Click`. (Alternatively, you can use **Navigate | Declaration** from the main menu or **Go To | Declaration** from the context menu.)
- Press and hold the `Ctrl` key, and point to the name of interest. When the text turns into a hyperlink, click the hyperlink.



```
insert into family (name, relation) values ('Emily', 'sister')
```

Executing an SQL statement

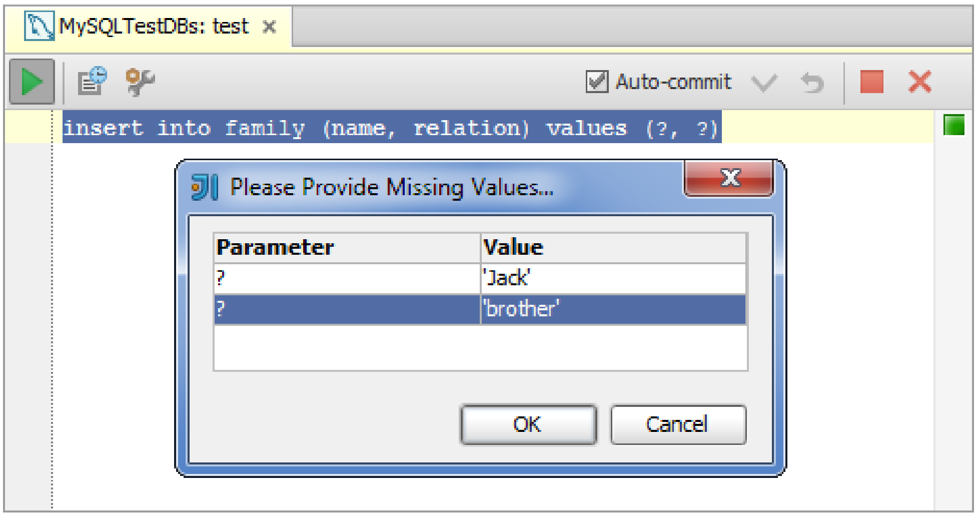
To execute a statement, place the cursor within that statement or within the line that contains the statement. Then do one of the following:

- Click ▶ on the toolbar.
- Press `Ctrl+Enter`.

Executing parameterized statements

Your statements can contain parameters, however, by the time you execute such statements the values of the parameters must be specified. There are the following ways of specifying the parameter values:

- Click ▶ on the toolbar or press `Ctrl+Enter` to execute the statement. In the dialog that opens, specify the parameter values and click **OK**.

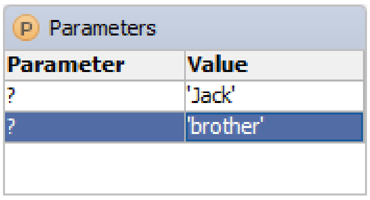


```
insert into family (name, relation) values (?, ?)
```

Parameter	Value
?	'Jack'
?	'brother'

(To start editing a value, double-click the corresponding table cell, or click the cell and press `F2` or `Alt+Enter`. To indicate that you have finished editing a value, press `Enter` or click a different cell. To quit the editing mode and restore an initial value, press `Escape`.)

- Alternatively, you can open the **Parameters** pane in the **Database Console** tool window (P on the toolbar) and specify the corresponding values there. (The values are edited in the same way as in the corresponding dialog.) Then execute the statement (▶ on the toolbar or `Ctrl+Enter`).



Parameter	Value
?	'Jack'
?	'brother'

For more information, see Parameters pane.

See also, Use explicit parameter patterns in SQL and Enable explicit parameter patterns (RegExp).

Executing a group of statements

To execute a group of statements that follow one another in the console, select (highlight) the statements to be executed (to select all the statements, use `Ctrl+A`) and do one of the following:

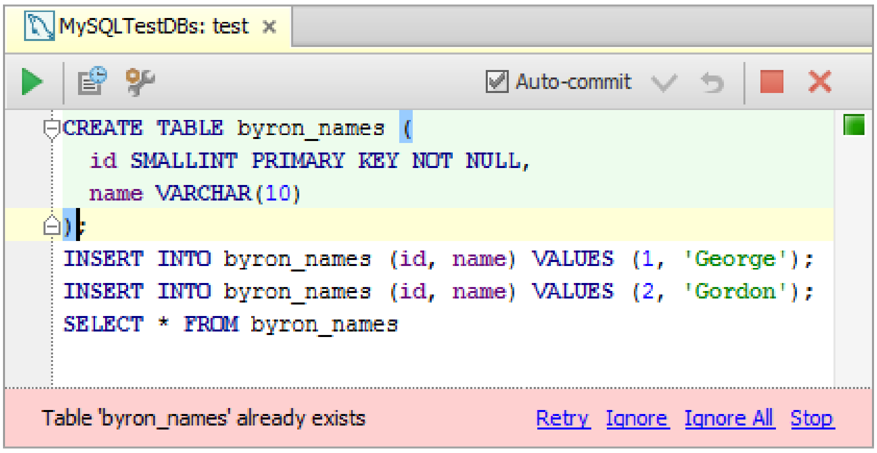
- Click ▶ on the toolbar.
- Press `Ctrl+Enter`.

See also, Using the error notification bar.

Using the error notification bar

If when running a statement an error occurs, an error notification bar appears in the lower part of the input pane.

This bar may be particularly useful when executing a sequence of statements (see Executing a group of statements) because in such a case it lets you select how to react.



```
CREATE TABLE byron_names (
  id SMALLINT PRIMARY KEY NOT NULL,
  name VARCHAR(10)
);
INSERT INTO byron_names (id, name) VALUES (1, 'George');
INSERT INTO byron_names (id, name) VALUES (2, 'Gordon');
SELECT * FROM byron_names
```

Table 'byron_names' already exists

The options are:

- **Retry.** Execute the sequence of statements starting from the one that caused the error.
- **Ignore.** Skip the erroneous statement and execute the sequence starting from the next statement. If another error occurs, the error notification bar will appear again.
- **Ignore All.** Skip the erroneous statement and execute the sequence starting from the next statement. If other errors occur, all the erroneous statements will be skipped and the error notification bar won't appear for these statements.
- **Stop.** Stop the execution of the sequence.

Showing the error notification bar in the input pane is enabled or disabled by means of the Show error notifications in editor check box on the Database page of the **Settings** dialog.

Executing a part of a statement (e.g. a subquery)

To execute a part of a statement (e. g. a subquery):

1. Select (highlight) the fragment that you want to execute.
2. Right-click the fragment and select **Execute Selection** from the context menu.

See also, Execute Selection.

Executing auto-memorized statements

As you run SQL statements in the console, IntelliJ IDEA memorizes them. So, at a later time, you can view the statements you have already run and, if necessary, run them again.

To open the dialog where the auto-memorized statements are shown (the **History** dialog), do one of the following:

- Click 🕓 on the toolbar.
- Press `Ctrl+Alt+E`.

There are two panes in the **History** dialog. The left-hand pane shows the list of the statements that you have run. For "long" statements, only their beginnings are shown. When you select a statement in this pane, the overall statement is shown in the pane to the right.

You can copy the statements from the **History** dialog into the input pane of the console. To copy a statement do one of the following:

- Double-click the statement to be copied.
- Select the statement of interest and press `Enter`.
- Select the statement and click **OK**.

(Once the statement is in the input pane, you can run it straight away.)

You can delete unnecessary memorized statements. To delete a statement select the statement in the **History** dialog and press `Delete`.

Finally, you can copy the statements from the console history into the input pane without opening the **History** dialog. When in the input pane, press `Ctrl+Down` or `Ctrl+Up`. As a result, the previous or next of the memorized statements is copied into the input pane.

- There is a separate history of executed statements for each of your DB data sources.
- The number of executed SQL statements to be memorized can be changed by editing the value in **Console commands history size** field on the Editor page of the **Settings** dialog.
- The history files that haven't been used for two weeks are automatically deleted. That is, if you haven't run SQL statements for a certain data source within this time, the corresponding history is discarded.

Managing database transactions

The **Auto-commit** check box, and the **Commit** ✓ and **Rollback** ↶ icons on the toolbar let you manage database transactions.

The **Auto-commit** check box is used to turn the autocommit mode for the database connection on or off.

In the autocommit mode, each SQL statement is executed in its own transaction that is implicitly committed. Consequently, the SQL statements executed in this mode cannot be rolled back.

If the autocommit mode is off, transactions are committed or rolled back explicitly by means of the **Commit** (✓) or **Rollback** (↶) command. Each commit or rollback starts a new transaction which provides grouping for a series of subsequent SQL statements.

In this case, the data manipulations in the transaction scope are committed or rolled back all at once when the transaction is committed or rolled back.

See Also

Procedures:

- Configuring Third-Party Tools
- Viewing Method Parameter Information

Language and Framework-Specific Guidelines:

- Data Sources
- Working with the Table Editor
- Running Injected SQL Statements from the Editor

Reference:

- Database Console
- Data Sources and Drivers dialog
- Database Tool Window
- Table Editor

Web Resources:

- Developer Community