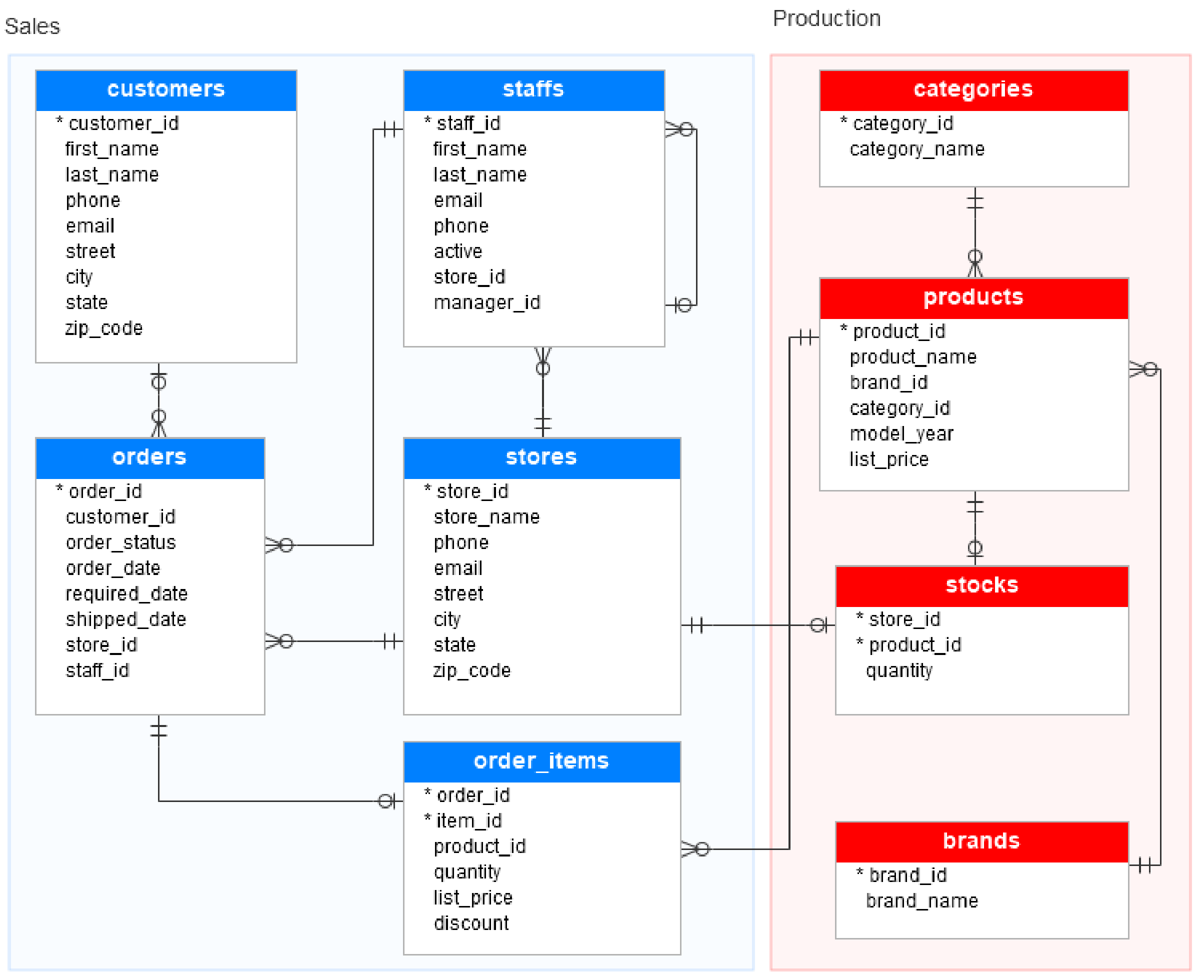
LAB - Installing Sample Database

In this lab, you'll learn working with a sample database called **BikeStores**.

The following diagram illustrates the **BikeStores** database diagram:



As you can see from the diagram, the BikeStores sample database has two schemas sales and production, and these schemas have nine tables.

Database Tables

Table sales.stores

The `sales.stores` table includes the store's information. Each store has a store name, contact information such as phone and email, and an address including street, city, state, and zip code.

```
CREATE TABLE sales.stores (
    store_id INT IDENTITY (1, 1) PRIMARY KEY,
    store_name VARCHAR (255) NOT NULL,
    phone VARCHAR (25),
    email VARCHAR (255),
    street VARCHAR (255),
    city VARCHAR (255),
    state VARCHAR (10),
    zip_code VARCHAR (5)
);
```

Table sales.staffs

The `sales.staffs` table stores the essential information of staffs including first name, last name. It also contains the communication information such as email and phone.

A staff works at a store specified by the value in the `store_id` column. A store can have one or more staffs.

A staff reports to a store manager specified by the value in the `manager_id` column. If the value in the manager_id is null, then the staff is the top manager.

If a staff no longer works for any stores, the value in the active column is set to zero.

```
CREATE TABLE sales.staffs (
    staff_id INT IDENTITY (1, 1) PRIMARY KEY,
    first_name VARCHAR (50) NOT NULL,
    last_name VARCHAR (50) NOT NULL,
    email VARCHAR (255) NOT NULL UNIQUE,
    phone VARCHAR (25),
    active tinyint NOT NULL,
    store_id INT NOT NULL,
    manager_id INT,
    FOREIGN KEY (store_id)
        REFERENCES sales.stores (store_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (manager_id)
        REFERENCES sales.staffs (staff_id)
        ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Table production.categories

The `production.categories` table stores the bike's categories such as children bicycles, comfort bicycles, and electric bikes.

```
CREATE TABLE production.categories (
    category_id INT IDENTITY (1, 1) PRIMARY KEY,
    category_name VARCHAR (255) NOT NULL
);
```

Table production.brands

The `production.brands` table stores the brand's information of bikes, for example, Electra, Haro, and Heller.

```
CREATE TABLE production.brands (
    brand_id INT IDENTITY (1, 1) PRIMARY KEY,
    brand_name VARCHAR (255) NOT NULL
);
```

Table production.products

The `production.products` table stores the product's information such as name, brand, category, model year, and list price.

Each product belongs to a brand specified by the `brand_id` column. Hence, a brand may have zero or many products.

Each product also belongs a category specified by the `category_id` column. Also, each category may have zero or many products.

```
CREATE TABLE production.products (
    product_id INT IDENTITY (1, 1) PRIMARY KEY,
    product_name VARCHAR (255) NOT NULL,
    brand_id INT NOT NULL,
    category_id INT NOT NULL,
    model_year SMALLINT NOT NULL,
    list_price DECIMAL (10, 2) NOT NULL,
    FOREIGN KEY (category_id)
        REFERENCES production.categories (category_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (brand_id)
        REFERENCES production.brands (brand_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Table sales.customers

The `sales.customers` table stores customer's information including first name, last name, phone, email, street, city, state and zip code.

```
CREATE TABLE sales.customers (
    customer_id INT IDENTITY (1, 1) PRIMARY KEY,
    first_name VARCHAR (255) NOT NULL,
    last_name VARCHAR (255) NOT NULL,
    phone VARCHAR (25),
    email VARCHAR (255) NOT NULL,
    street VARCHAR (255),
    city VARCHAR (50),
    state VARCHAR (25),
    zip_code VARCHAR (5)
);
```

Table sales.orders

The `sales.orders` table stores the sales order's header information including customer, order status, order date, required date, shipped date.

It also stores the information on where the sales transaction was created (store) and who created it (staff).

Each sales order has a row in the sales_orders table. A sales order has one or many line items stored in the `sales.order_items` table.

```
CREATE TABLE sales.orders (
    order_id INT IDENTITY (1, 1) PRIMARY KEY,
    customer_id INT,
    order_status tinyint NOT NULL,
    -- Order status: 1 = Pending; 2 = Processing; 3 = Rejected; 4 = Completed
    order_date DATE NOT NULL,
    required_date DATE NOT NULL,
    shipped_date DATE,
    store_id INT NOT NULL,
    staff_id INT NOT NULL,
    FOREIGN KEY (customer_id)
        REFERENCES sales.customers (customer_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (store_id)
        REFERENCES sales.stores (store_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (staff_id)
        REFERENCES sales.staffs (staff_id)
        ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Table sales.order_items

The `sales.order_items` table stores the line items of a sales order. Each line item belongs to a sales order specified by the `order_id` column.

A sales order line item includes product, order quantity, list price, and discount.

```
CREATE TABLE sales.order_items(
    order_id INT,
    item_id INT,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    list_price DECIMAL (10, 2) NOT NULL,
    discount DECIMAL (4, 2) NOT NULL DEFAULT 0,
    PRIMARY KEY (order_id, item_id),
    FOREIGN KEY (order_id)
        REFERENCES sales.orders (order_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (product_id)
        REFERENCES production.products (product_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Table production.stocks

The `production.stocks` table stores the inventory information i.e. the quantity of a particular product in a specific store.

```
CREATE TABLE production.stocks (
    store_id INT,
    product_id INT,
    quantity INT,
    PRIMARY KEY (store_id, product_id),
    FOREIGN KEY (store_id)
        REFERENCES sales.stores (store_id)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (product_id)
        REFERENCES production.products (product_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

Click the following link to download the sample database script:

Download SQL Server Sample Database

Now, you should be familiar with the BikeStores sample database and ready to load it into the SQL Server.

Load Sample Database

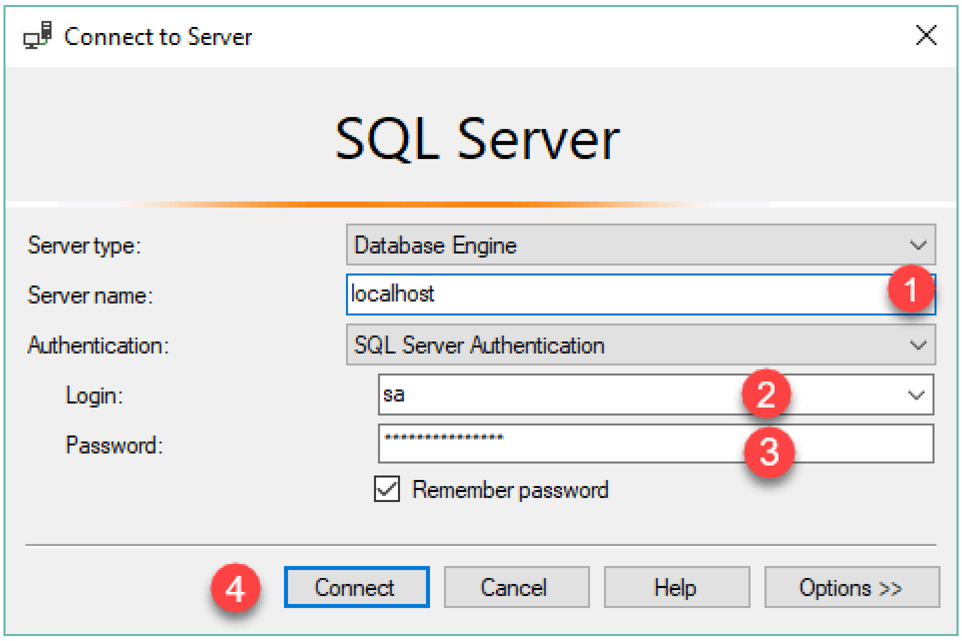
Unzip file and you will see three SQL script files:

- `BikeStores Sample Database - create objects.sql` – this file is for creating database objects including schemas and tables.
- `BikeStores Sample Database - load data.sql` – this file is for inserting data into the tables
- `BikeStores Sample Database - drop all objects.sql` – this file is for removing the tables and their schemas from the sample database. It is useful when you want to refresh the sample database.

Third, let’s create a database, create the schemas and tables, and load the sample data.

Step 1

Connect to the SQL Server by (1) choosing the server name, (2) enter the user and (3) password and (4) click the **Connect** button.

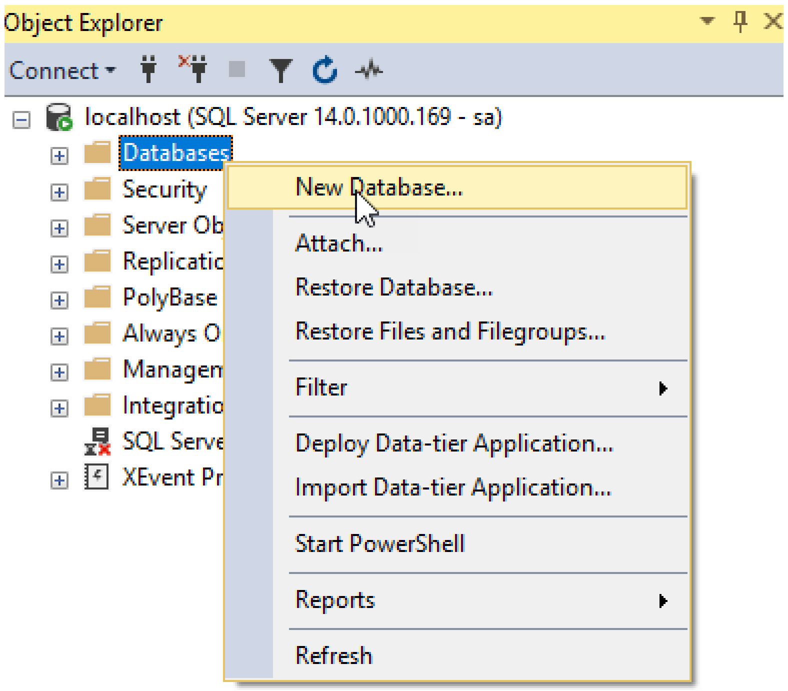

Connect to Server

SQL Server

Server type: Database Engine
Server name: localhost
Authentication: SQL Server Authentication
Login: sa
Password: ***************
☑ Remember password

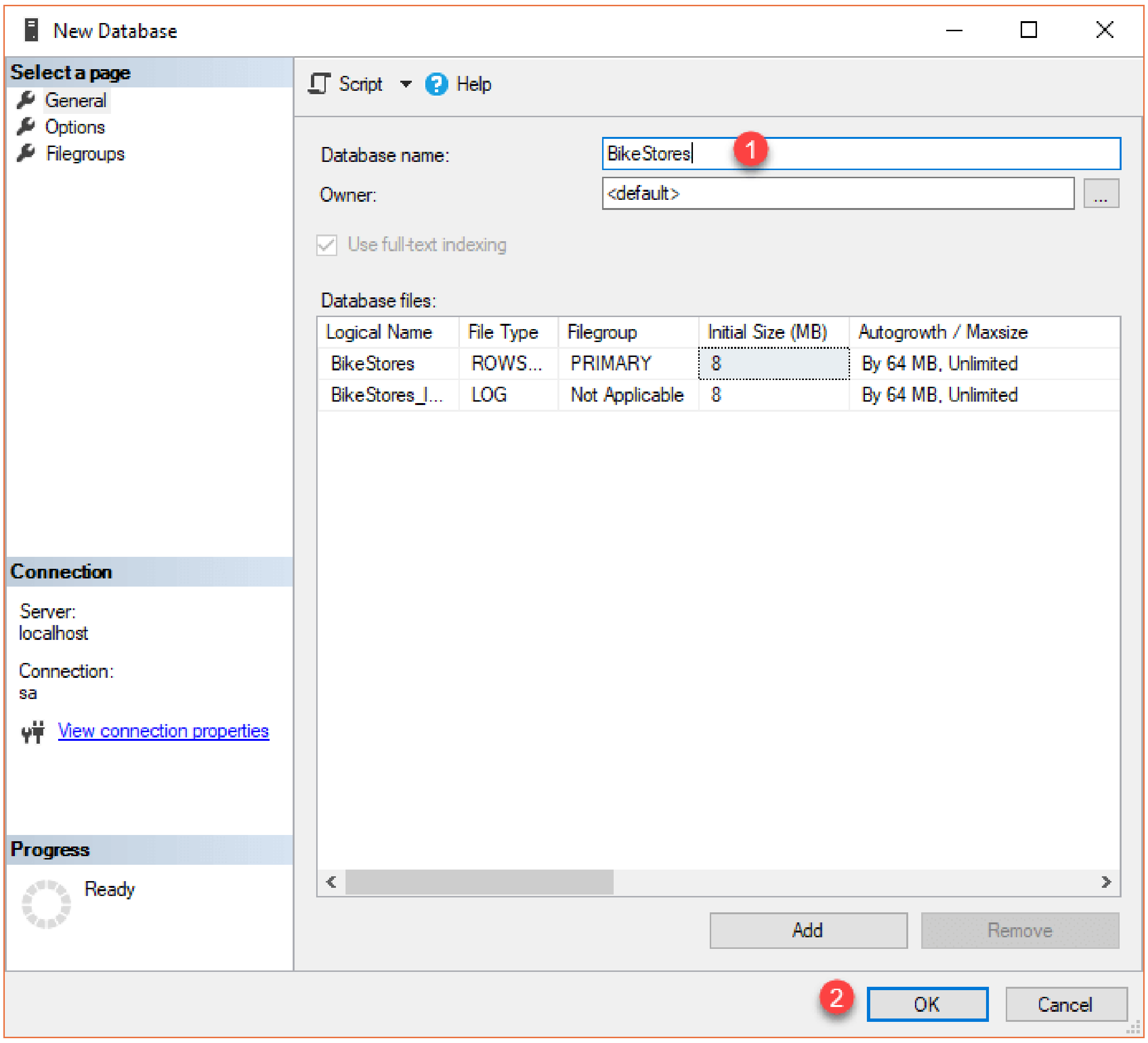
Step 2

Right-click the **Databases** node in the **Object Explorer** and select the **New Database...** menu item



Step 3

(1) Enter the **Database name** as BikeStores and (2) click the **OK** button to create the new database.



Database name: BikeStores

Owner: <default>

Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize
BikeStores	ROWS...	PRIMARY	8	By 64 MB, Unlimited
BikeStores_l...	LOG	Not Applicable	8	By 64 MB, Unlimited

Connection

Server:
localhost

Connection:
sa

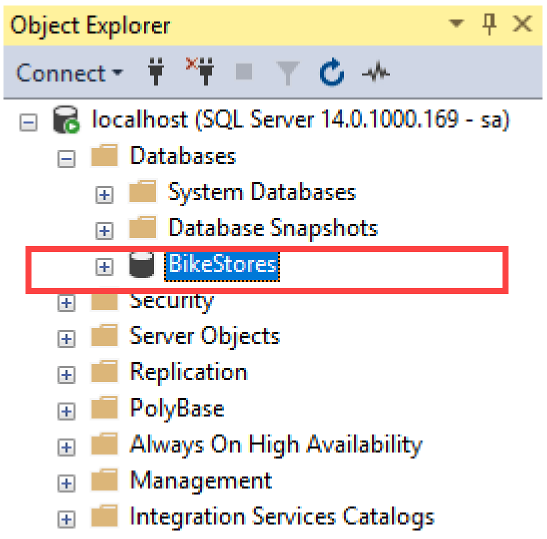
View connection properties

Progress

Ready

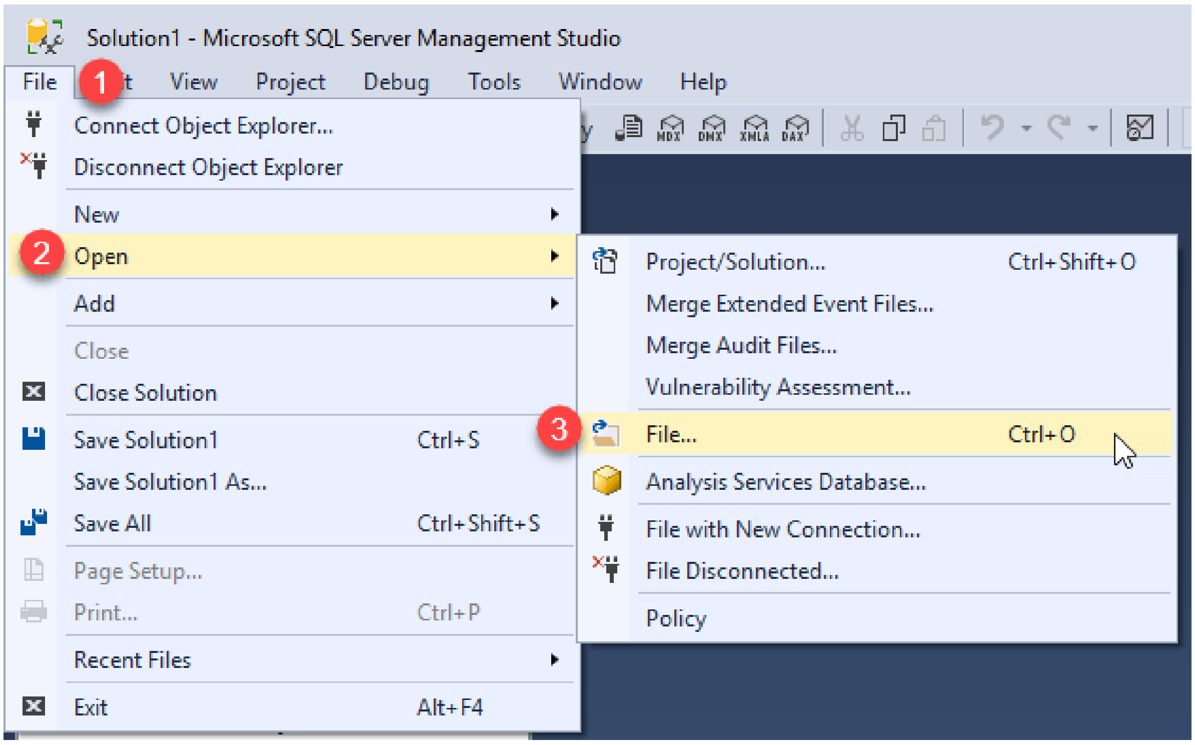
Step 4

If everything is fine, you will see the database **BikeStores** appears under Databases node as shown in the screenshot below:



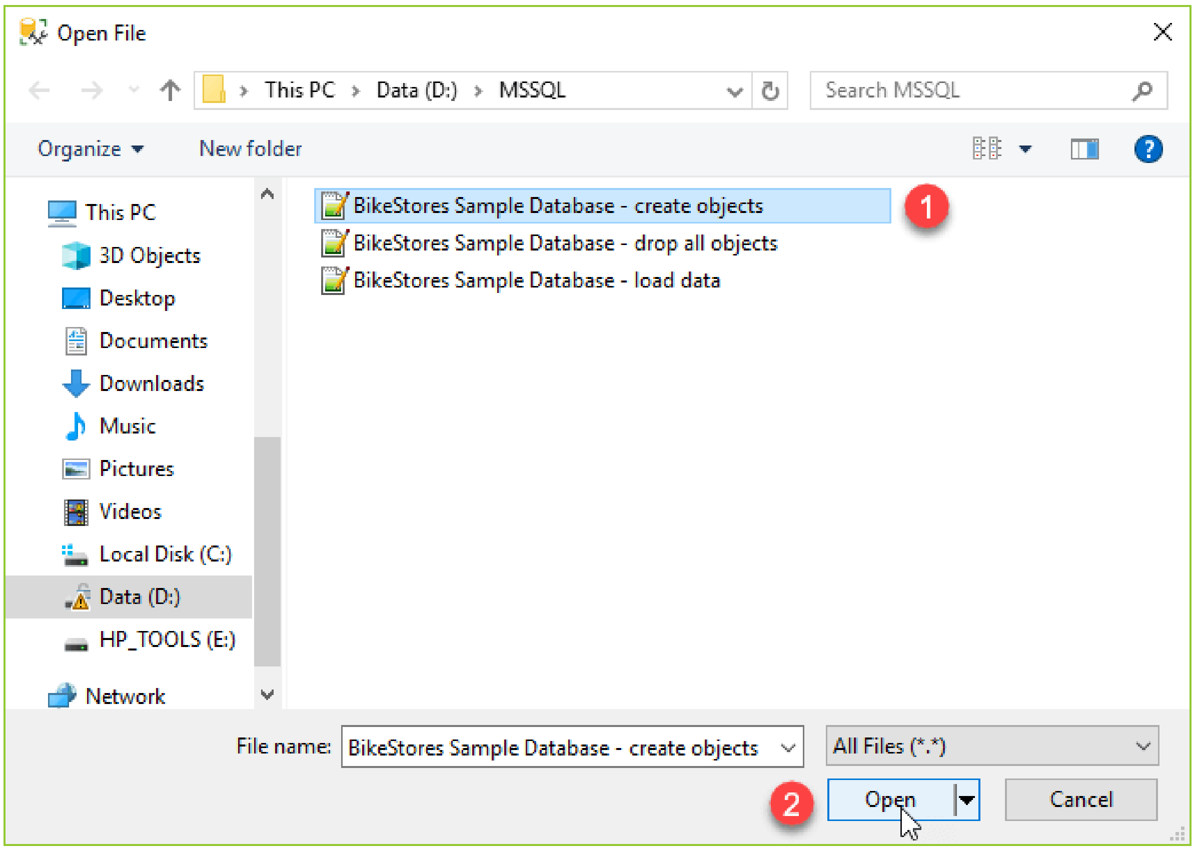
Step 5

From the File menu, choose Open > File… menu item to open a script file.


Solution1 - Microsoft SQL Server Management Studio
File
1
Edit
View
Project
Debug
Tools
Window
Help
Connect Object Explorer...
Disconnect Object Explorer
New
2
Open
Add
Close
Close Solution
Save Solution1 Ctrl+S
Save Solution1 As...
Save All Ctrl+Shift+S
Page Setup...
Print... Ctrl+P
Recent Files
Exit Alt+F4
Project/Solution... Ctrl+Shift+O
Merge Extended Event Files...
Merge Audit Files...
Vulnerability Assessment...
3
File... Ctrl+O
Analysis Services Database...
File with New Connection...
File Disconnected...
Policy

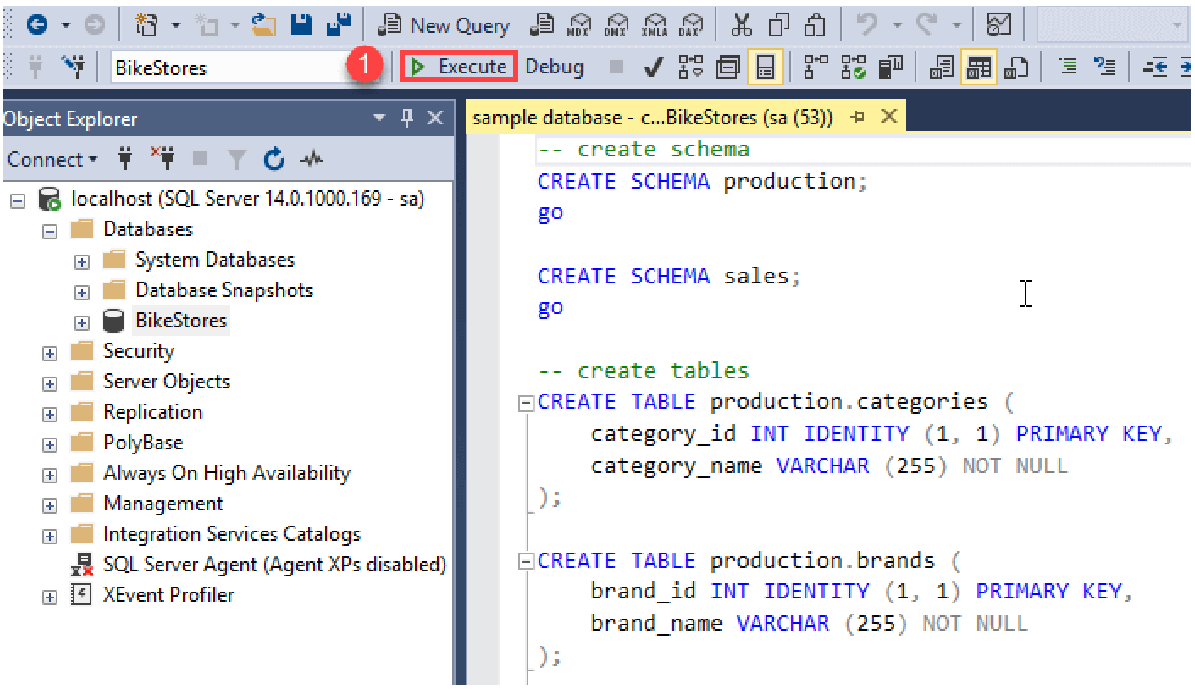
Step 6

Select the **BikeStores Sample Database – create** objects.sql file and click the `Open` button


Open File
This PC > Data (D:) > MSSQL
Search MSSQL
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Data (D:)
HP_TOOLS (E:)
Network
BikeStores Sample Database - create objects
1
BikeStores Sample Database - drop all objects
BikeStores Sample Database - load data
File name: BikeStores Sample Database - create objects
All Files (*.*)
2
Open
Cancel

Step 7

Click the **Execute** button to execute the SQL script.


New Query
BikeStores
1
Execute
Debug
Object Explorer
sample database - c...BikeStores (sa (53))
Connect
localhost (SQL Server 14.0.1000.169 - sa)
Databases
System Databases
Database Snapshots
BikeStores
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs disabled)
XEvent Profiler

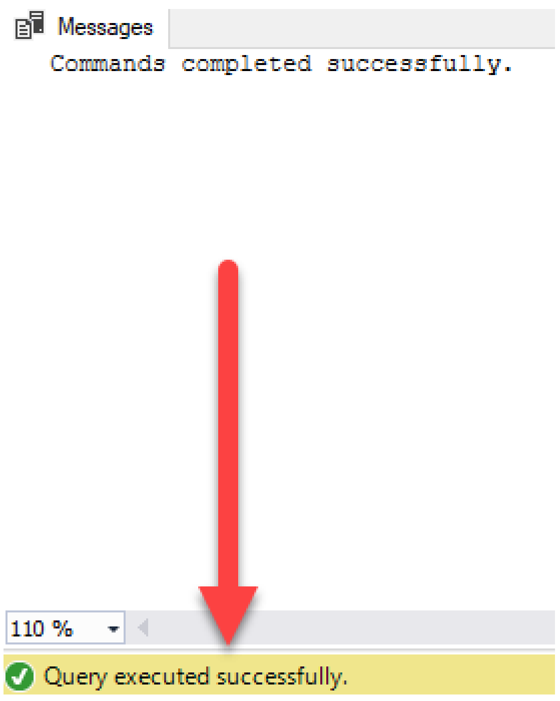
```sql
-- create schema
CREATE SCHEMA production;
go

CREATE SCHEMA sales;
go

-- create tables
CREATE TABLE production.categories (
    category_id INT IDENTITY (1, 1) PRIMARY KEY,
    category_name VARCHAR (255) NOT NULL
);

CREATE TABLE production.brands (
    brand_id INT IDENTITY (1, 1) PRIMARY KEY,
    brand_name VARCHAR (255) NOT NULL
);
```

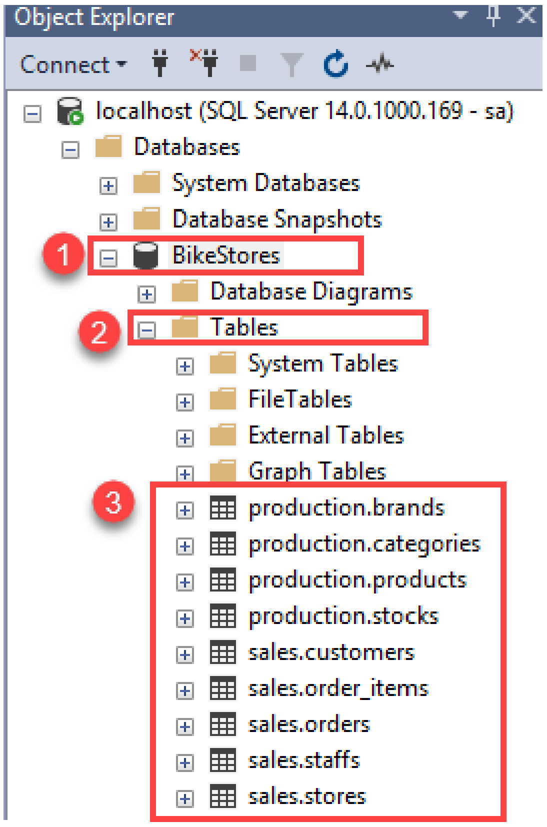
You should see the following result indicated that the query executed successfully.



```
Commands completed successfully.
```

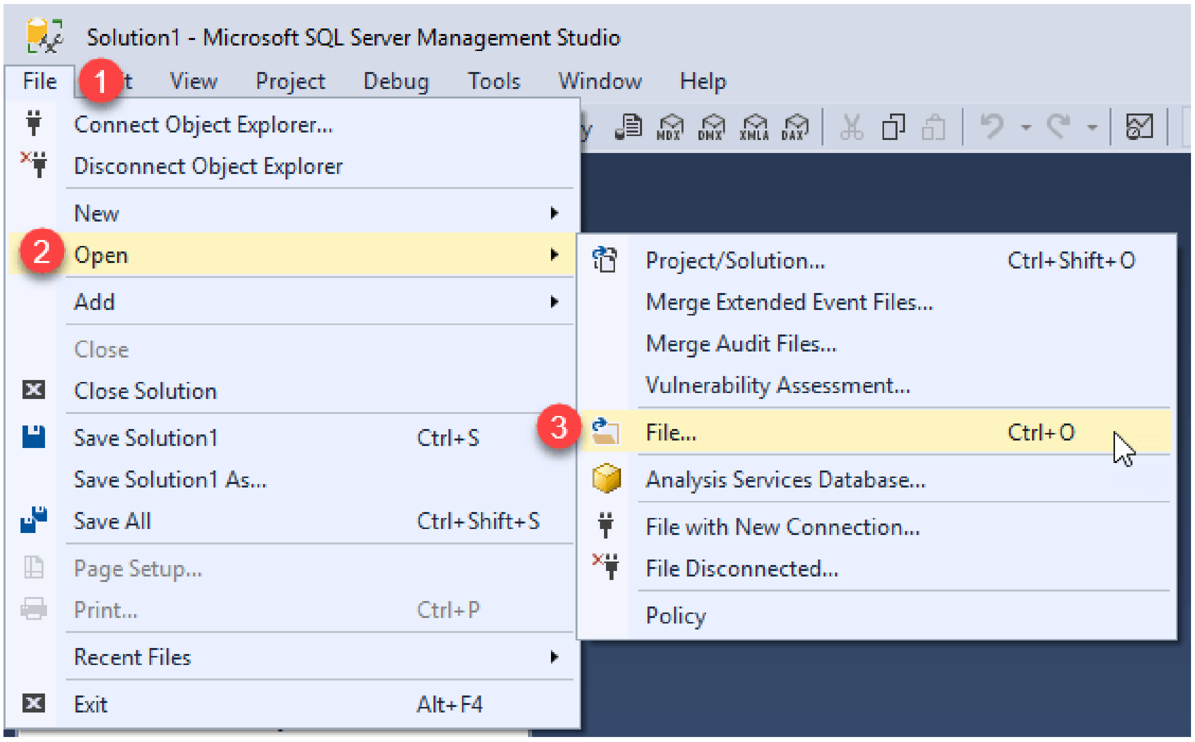
Query executed successfully.

If you expand the **BikeStores > Tables**, you will see the schemas and their tables are created as shown below:



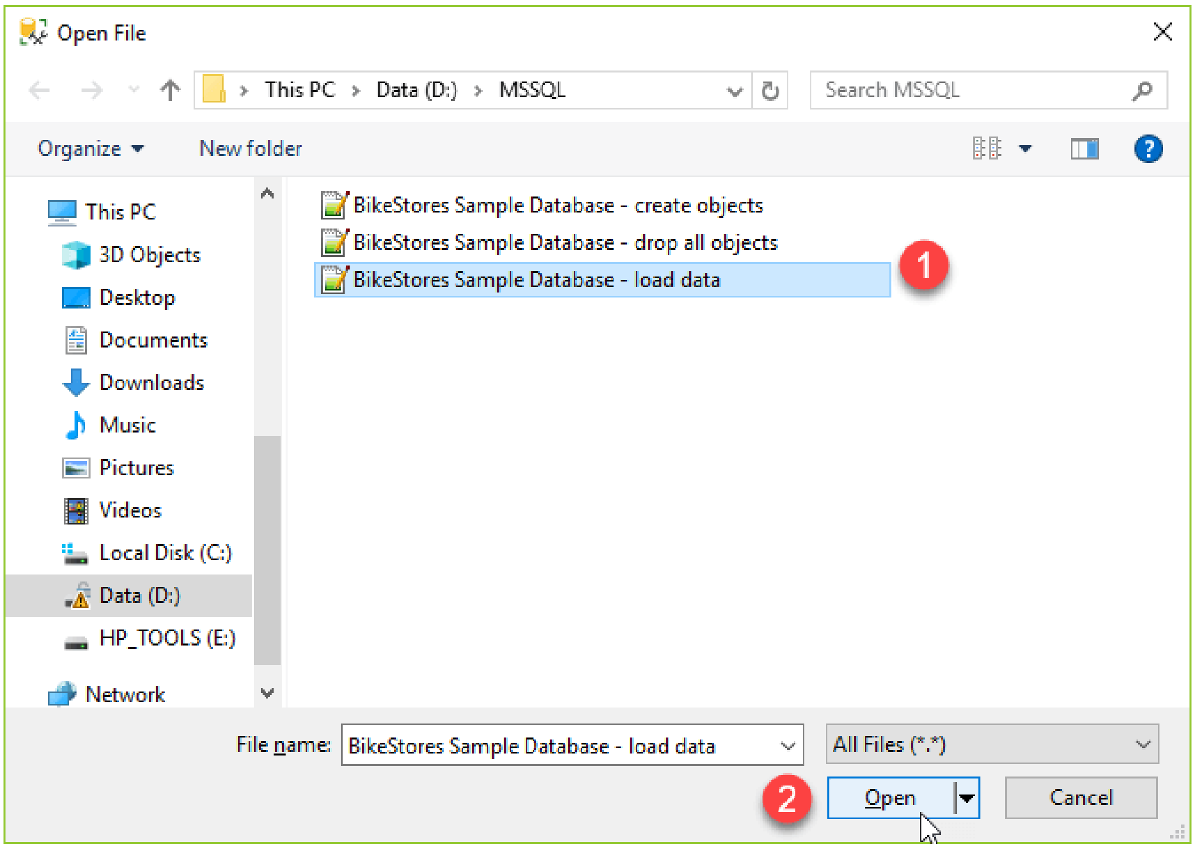
Step 8

Open the file for loading data into the tables.


Solution1 - Microsoft SQL Server Management Studio
File
1
View
Project
Debug
Tools
Window
Help
Connect Object Explorer...
Disconnect Object Explorer
New
2
Open
Add
Close
Close Solution
Save Solution1 Ctrl+S
Save Solution1 As...
Save All Ctrl+Shift+S
Page Setup...
Print... Ctrl+P
Recent Files
Exit Alt+F4
Project/Solution... Ctrl+Shift+O
Merge Extended Event Files...
Merge Audit Files...
Vulnerability Assessment...
3
File... Ctrl+O
Analysis Services Database...
File with New Connection...
File Disconnected...
Policy

Step 9

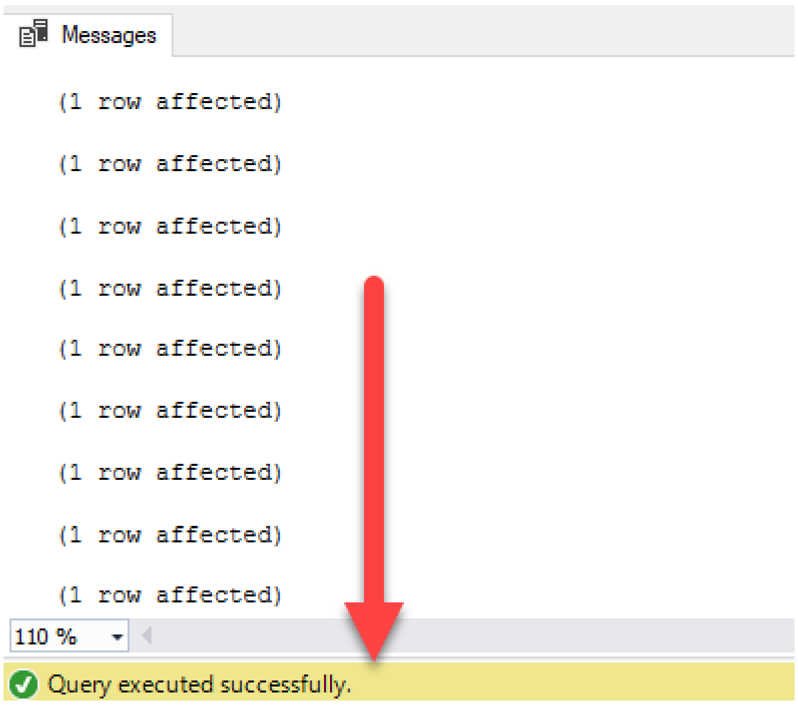
Choose the **BikeStores Sample Database – load data.sql** file and click the Open button.


Open File
This PC > Data (D:) > MSSQL
Search MSSQL
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Data (D:)
HP_TOOLS (E:)
Network
BikeStores Sample Database - create objects
BikeStores Sample Database - drop all objects
BikeStores Sample Database - load data
1
File name: BikeStores Sample Database - load data
All Files (*.*)
2
Open
Cancel

Step 10

Click the **Execute** button to load data into the tables.

You should see the following message indicating that all the statements in the script were executed successfully.



```
(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)
```

Query executed successfully.

In this lab, you have learned how to load the BikeStores sample database into the SQL Server.