Creating DML and DDL database objects

Author: Honorio Apaza

COMMANDS SQL / MATERIAL

BIKESTORE DATABASE

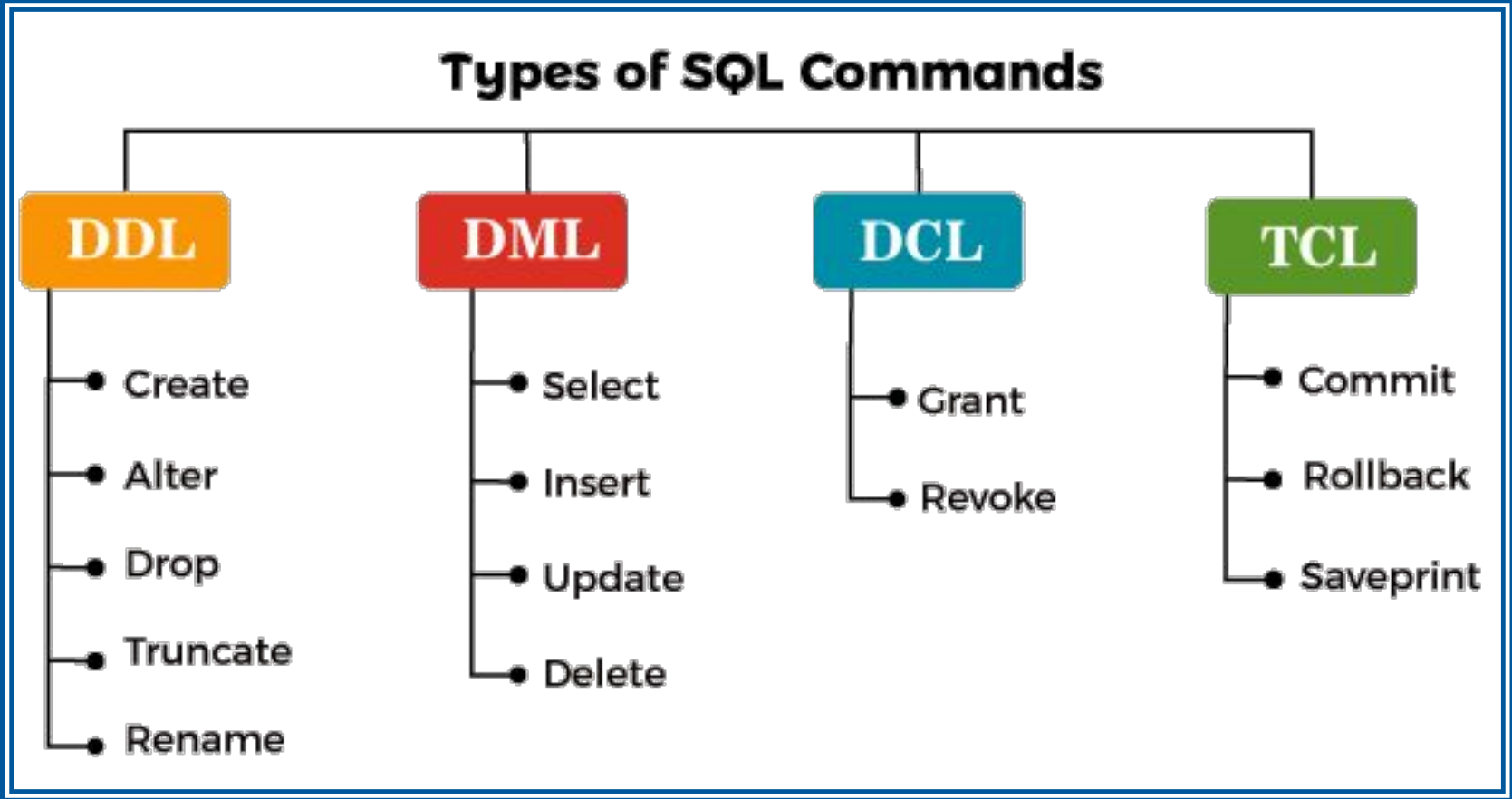
Overview
(DML vs DDL)

DDL:

DDL is Data **Definition** Language which is used to define data structures. For example: create table, alter table are instructions in SQL.

DML:

DML is Data **Manipulation** Language which is used to manipulate data itself. For example: insert, update, delete are instructions in SQL.


Types of SQL Commands
DDL
Create
Alter
Drop
Truncate
Rename
DML
Select
Insert
Update
Delete
DCL
Grant
Revoke
TCL
Commit
Rollback
Saveprint

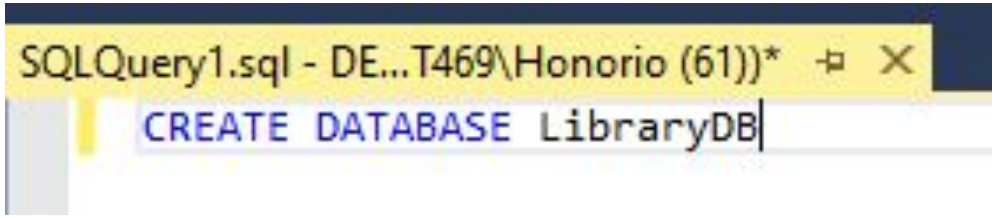
Data Definition Language (DDL)

CREATE

Creating database

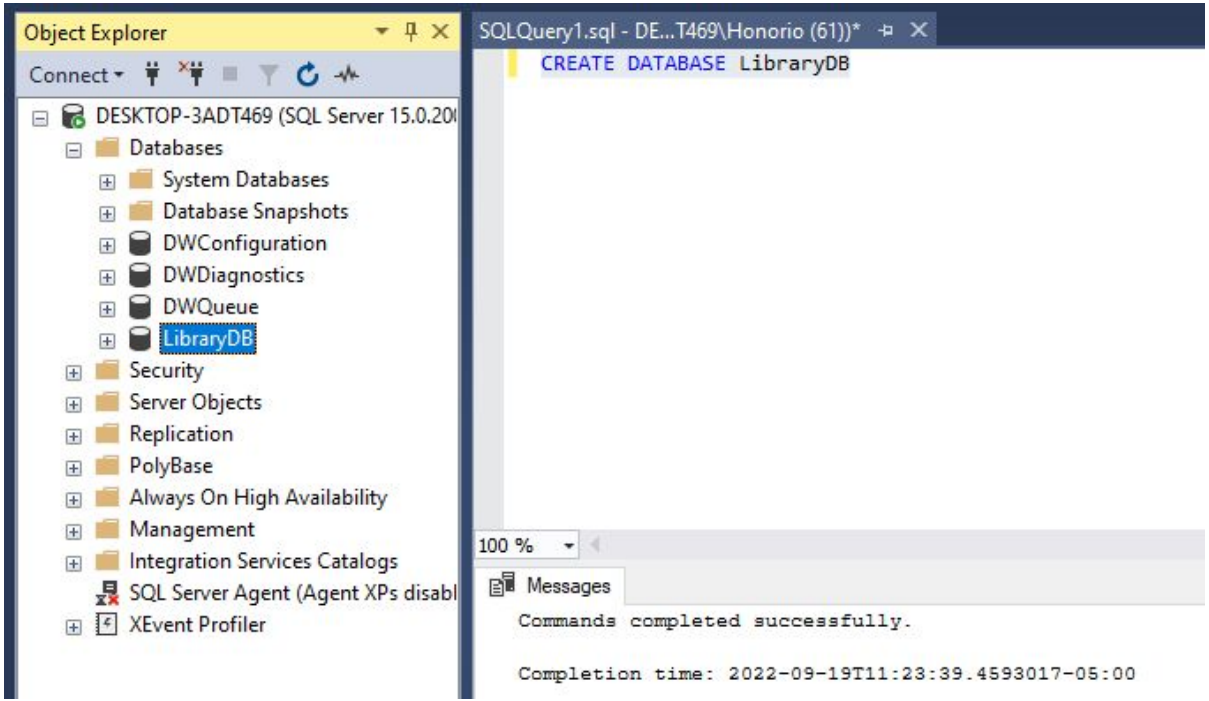
Following the next example to demonstrates how the CREATE query can be used to create a database in MS SQL Server:

```
CREATE DATABASE LibraryDB
```



Output

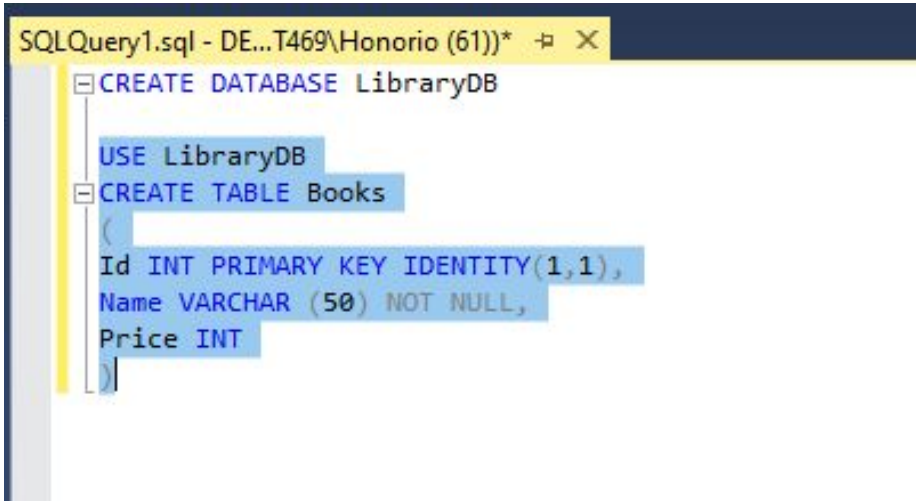
The script above creates a database named “LibraryDB” in MS SQL Server.



Creating a table

The CREATE query is also used to add tables in an existing database as shown in the following script:

```
USE LibraryDB
CREATE TABLE Books
(
Id INT PRIMARY KEY
IDENTITY(1,1),
Name VARCHAR (50) NOT
NULL,
Price INT
)
```



The “Books” table contains three columns: Id, Name, and Price. The Id column is the primary key column and it cannot be NULL. A column with a PRIMARY KEY constraint must contain unique values.

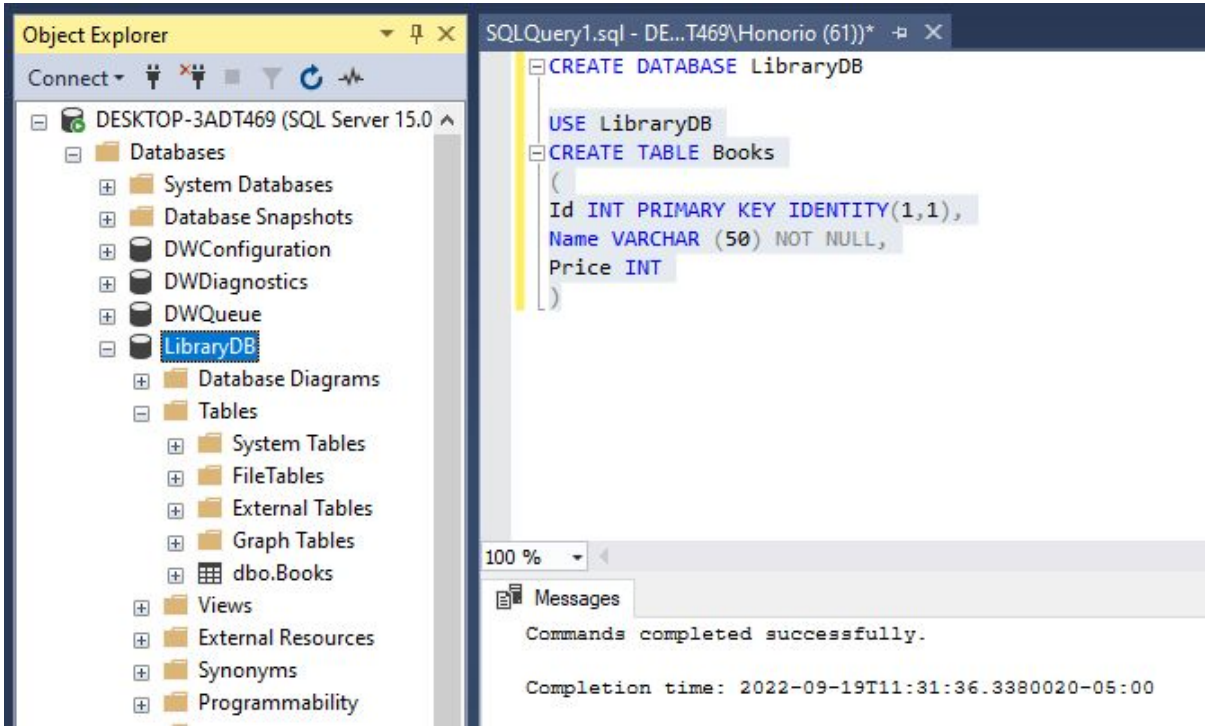
However, since we have set the IDENTITY property for the Id column, every time a new record is added in the Books table, the value of the Id column will be incremented by 1, starting from 1.

You need to specify the values for the Name column as well as it cannot have NULL.

Finally, the Price column can have NULL values

Output

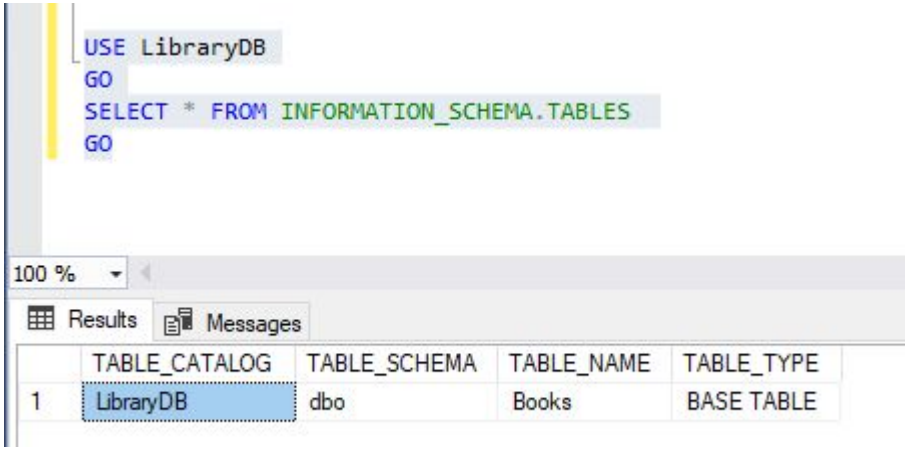
The above script creates a table named “Books” in the “LibraryDB” database that we created earlier.



SQL DDL

To view all the tables in the LibraryDB, execute the following QL DDL script:

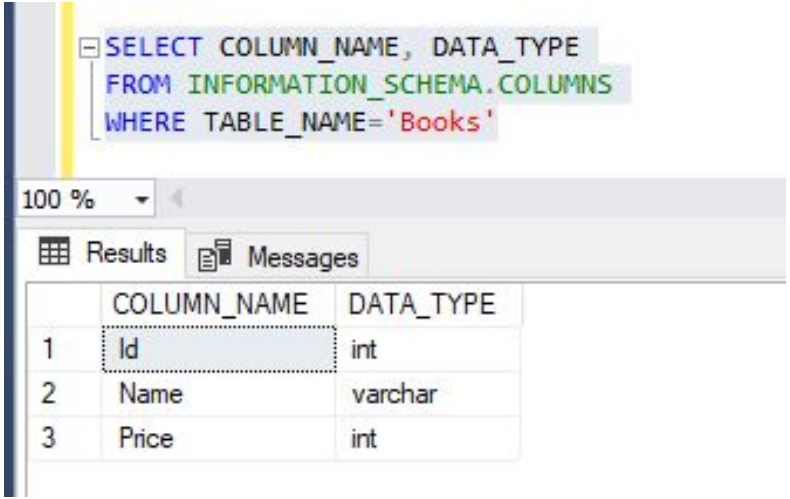
```
USE LibraryDB
GO
SELECT * FROM
INFORMATION_SCHEMA.TABLES
GO
```



SQL DDL

Similarly, to see all the columns in the Books table, run the following script:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='Books'
```

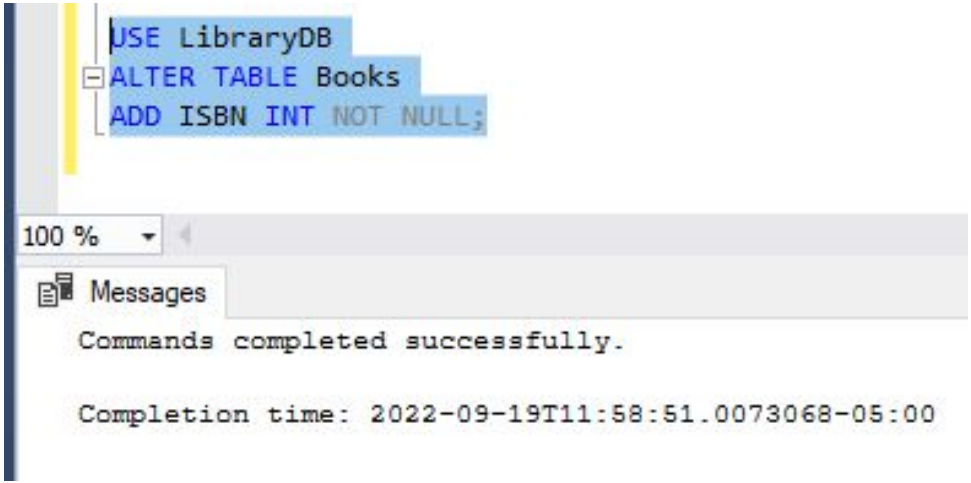


ALTER

Adding a new column

For example, if we want to add a new column e.g. ISBN to the existing Books table in the LibraryDB database, the ALTER command can be used as follows:

```
USE LibraryDB
ALTER TABLE Books
ADD ISBN INT NOT NULL;
```



Output

In the output, you can see the newly added ISBN column.

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='Books'
```

100 %

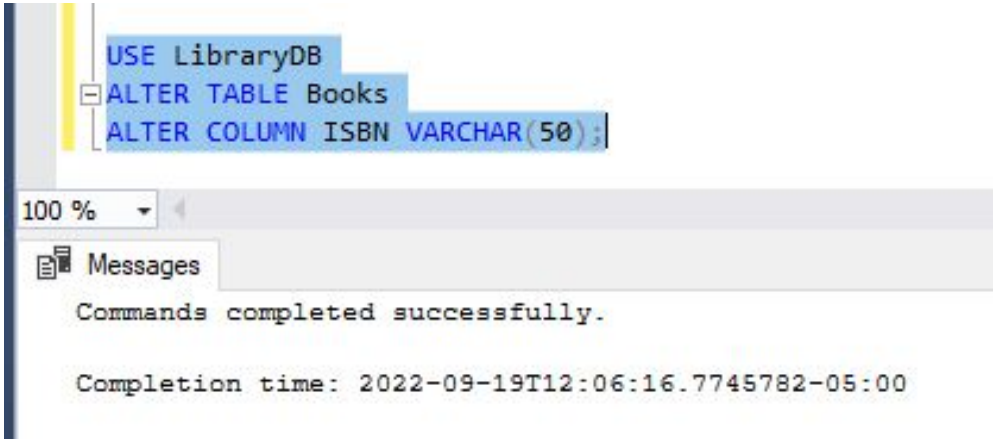
Results | Messages

	COLUMN_NAME	DATA_TYPE
1	Id	int
2	Name	varchar
3	Price	int
4	ISBN	int

Modifying an existing column

For example, you want to change the data type of the ISBN column from INT to VARCHAR (50). The ALTER query can be used as follows:

```
USE LibraryDB
ALTER TABLE Books
ALTER COLUMN ISBN
VARCHAR(50);
```



Output

If you again select the column names, you will see the updated data type (VARCHAR) for the ISBN column.

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='Books'
```

100 %

Results | Messages

	COLUMN_NAME	DATA_TYPE
1	Id	int
2	Name	varchar
3	Price	int
4	ISBN	varchar

DROP

Deleting a database

The following DROP command deletes the LibraryDB database that we created earlier:

```
DROP DATABASE LibraryDB
```

Exercise to student

Deleting a table

The DROP command is a type of SQL DDL command that is used to delete an existing table. For instance, the following command will delete the Books table:

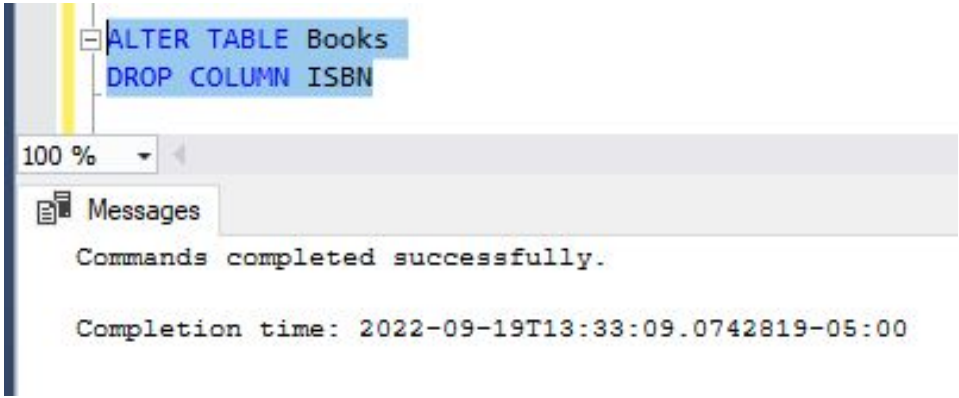
```
DROP TABLE Books
```

Exercise to student

Deleting a column

To delete a column within a database, the DROP query is used in combination with the ALTER query. The ALTER query specifies the table that you want to delete whereas the DROP query specifies the column to delete within the table specified by the ALTER query. Let's drop the ISBN column from the Books:

```
ALTER TABLE Books
DROP COLUMN ISBN
```



Output

To delete a column within a database, the DROP query is used in combination with the ALTER query. The ALTER query specifies the table that you want to delete whereas the DROP query specifies the column to delete within the table specified by the ALTER query. Let's drop the ISBN column from the Books:

```
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='Books'
```

100 %

Results | Messages

	COLUMN_NAME	DATA_TYPE
1	Id	int
2	Name	varchar
3	Price	int

TRUNCATE

Insert to truncate

The TRUNCATE command in SQL DDL is used to remove all the records from a table. Let's insert a few records in the Books table:

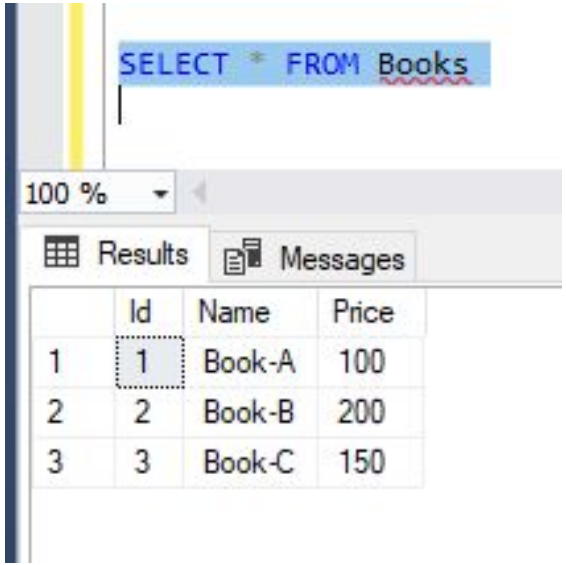
```
INSERT INTO Books
VALUES ('Book-A', 100),
('Book-B', 200),
('Book-C', 150)
```

```
--TRUNCATE
USE LibraryDB
INSERT INTO dbo.Books(Name, Price)
VALUES('Book-A', 100),
('Book-B', 200),
('Book-C', 150)
```

100 %

Output

You can see the three records that we inserted in the Books table.

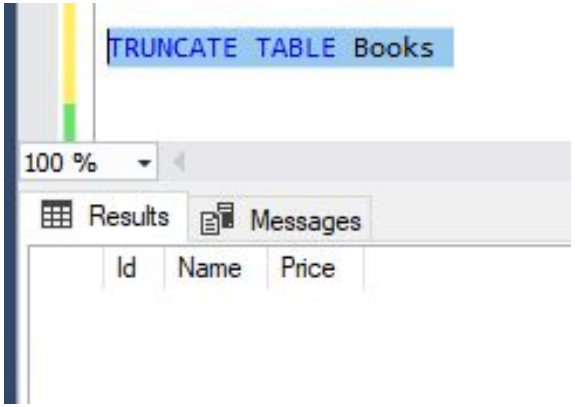


	Id	Name	Price
1	1	Book-A	100
2	2	Book-B	200
3	3	Book-C	150

Truncate

The TRUNCATE command will remove all the records from the Books table as shown below:

```
TRUNCATE TABLE Books
```



RENAME

Rename

The RENAME command will rename the Name column to Libros from the Libros table as shown below:

```
SP_RENAME
'Books.Name','Libros',
'COLUMN'
```

```
--RENAME

SP_RENAME 'Books.Name','Libros', 'COLUMN'
```

100 %

Messages

```
Caution: Changing any part of an object name could break scripts and stored procedures.

Completion time: 2022-09-19T14:23:08.9039287-05:00
```

Output

You can see the name changes from the Name column to Libros in the Books table.

```
--RENAME

SP_RENAME 'Books.Name','Libros', 'COLUMN'
```

100 %

Results | Messages

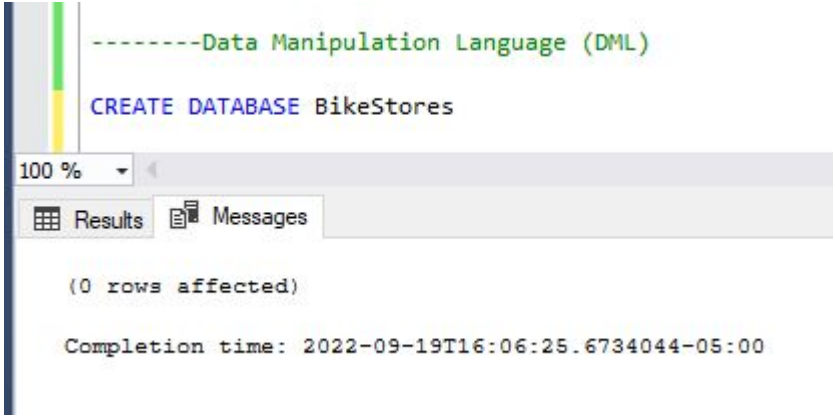
	Id	Libros	Price
1	1	Book-A	100
2	2	Book-B	200
3	3	Book-C	150

Data Manipulation Language (DML)

Create DB

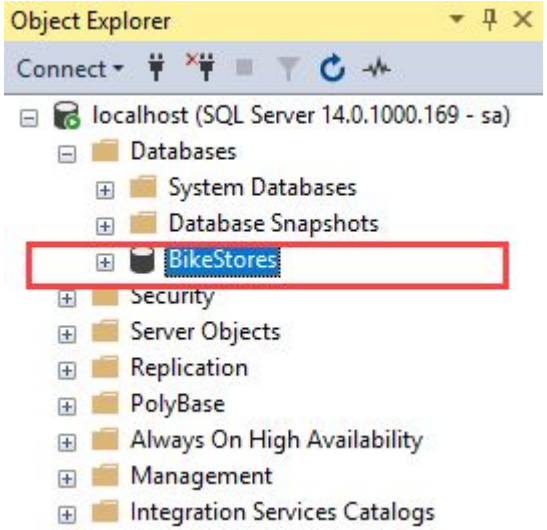
Next, we create the `BikeStores` database by running the following command:

```
CREATE DATABASE BikeStores
```

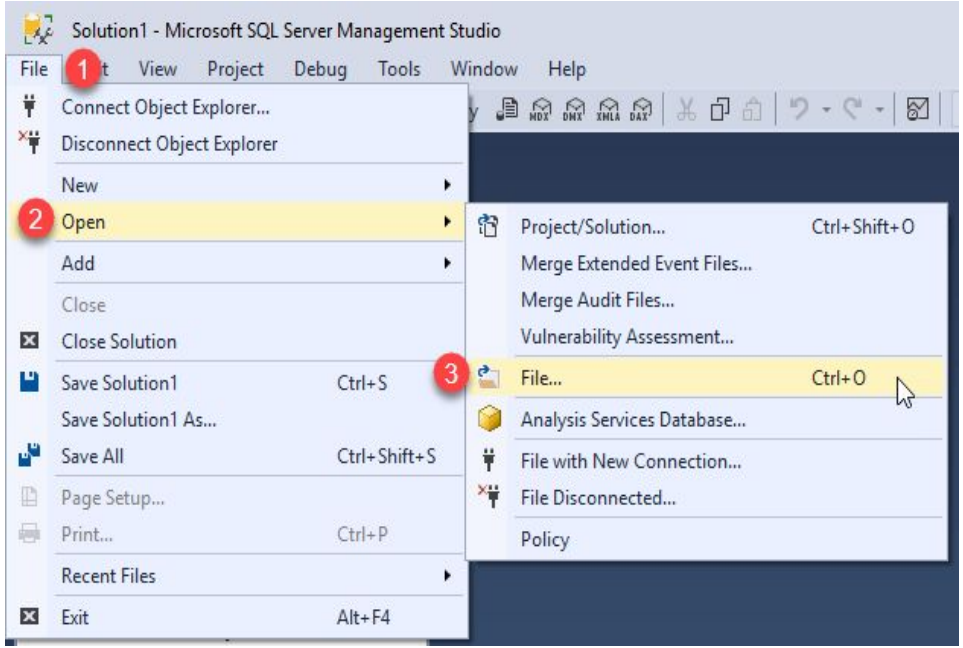


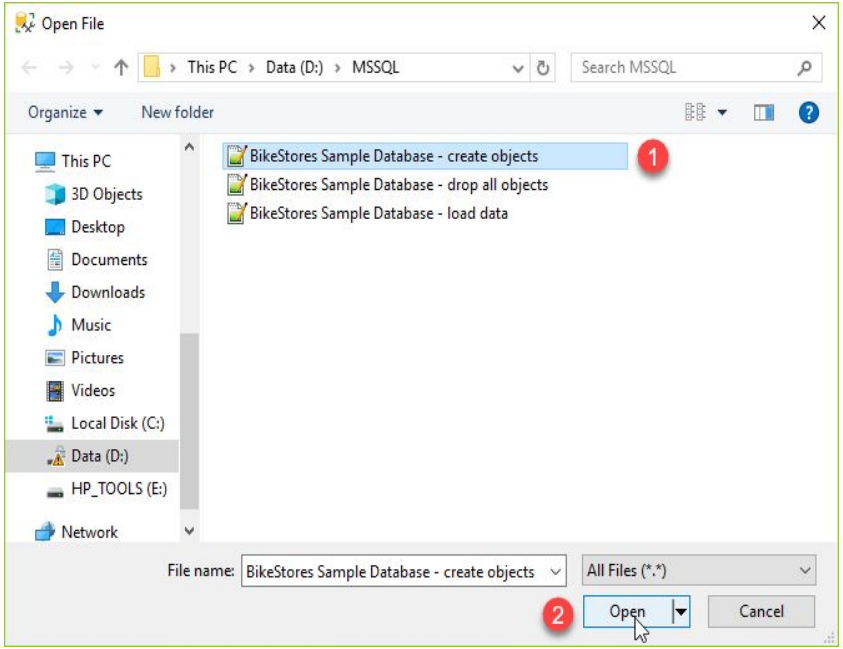
Output

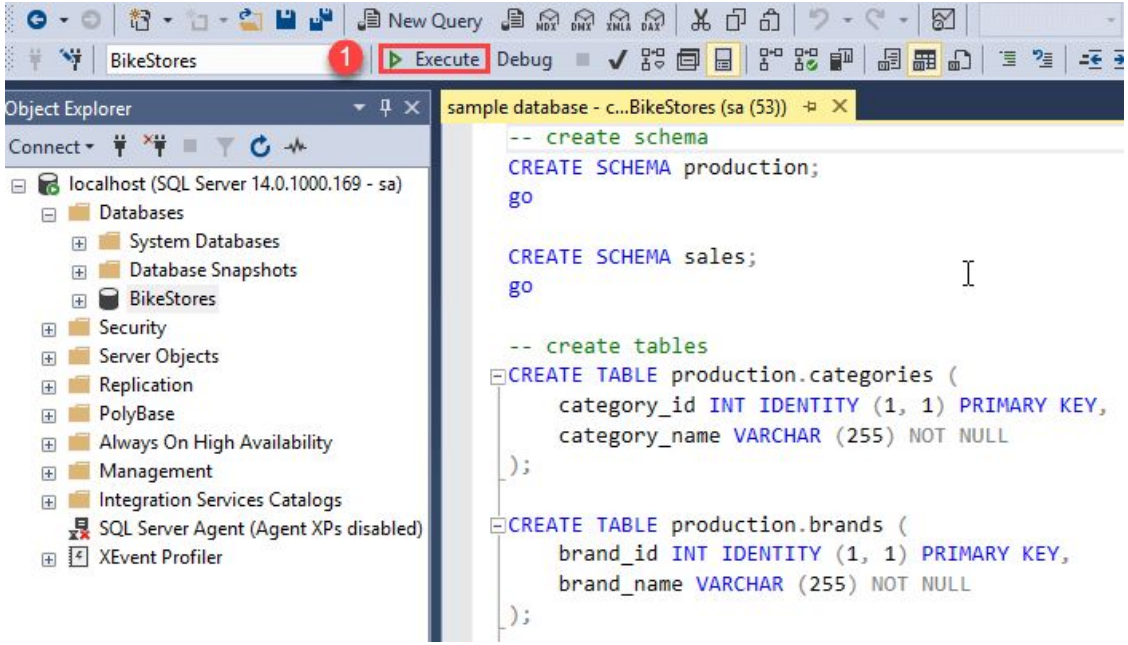
You can see the new database named XX and its respective tables and data records.

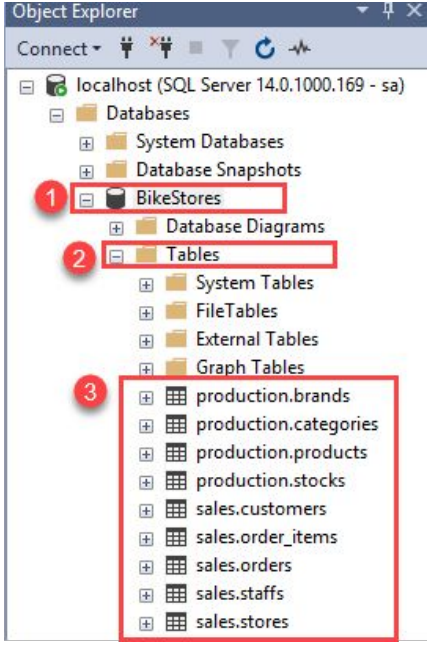


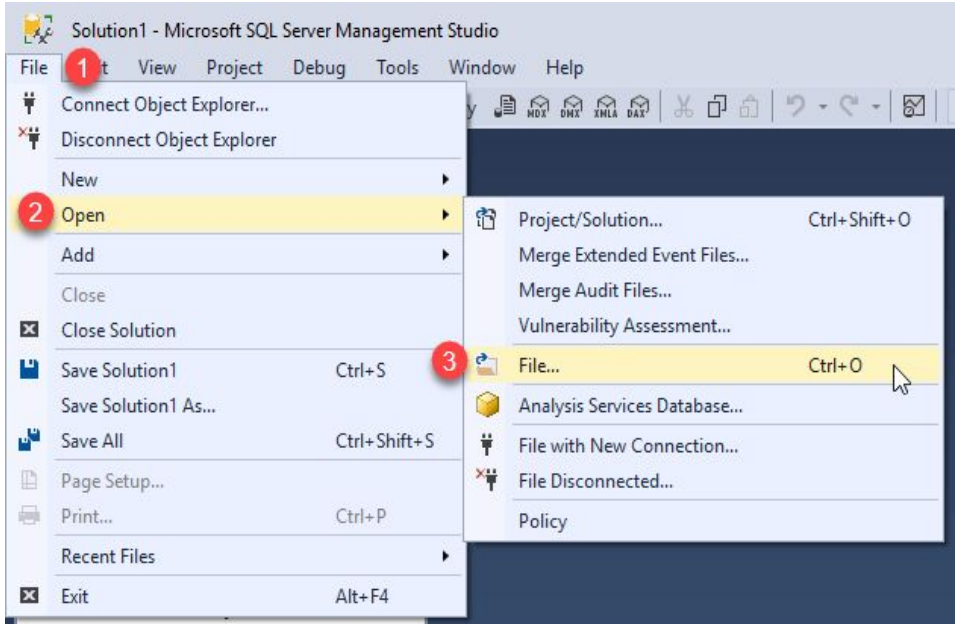
DataBase: https://github.com/Honorio-apz/BikeStores-DataBase

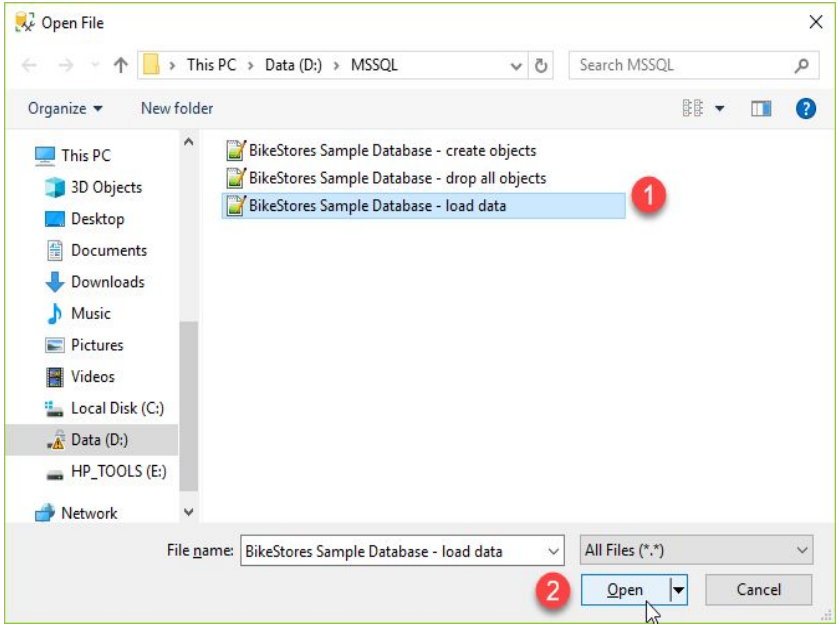





New Query
BikeStores
1
Execute
Debug
Object Explorer
Connect
localhost (SQL Server 14.0.1000.169 - sa)
Databases
System Databases
Database Snapshots
BikeStores
Security
Server Objects
Replication
PolyBase
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent (Agent XPs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


Object Explorer
Connect
localhost (SQL Server 14.0.1000.169 - sa)
Databases
System Databases
Database Snapshots
1
BikeStores
Database Diagrams
2
Tables
System Tables
FileTables
External Tables
Graph Tables
3
production.brands
production.categories
production.products
production.stocks
sales.customers
sales.order_items
sales.orders
sales.staffs
sales.stores


Solution1 - Microsoft SQL Server Management Studio
File
View
Project
Debug
Tools
Window
Help
Connect Object Explorer...
Disconnect Object Explorer
New
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...
File... Ctrl+O
Analysis Services Database...
File with New Connection...
File Disconnected...
Policy
1
2
3


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
File name: BikeStores Sample Database - load data
All Files (*.*)
Open
Cancel
1
2

SELECT

Select

In this example, we have fetched fields such as `order_id`, `customer_id`, `order_status` and `order_date` from `sales.orders` table.

```
SELECT [order_id],
[customer_id],
[order_status], [order_date]

FROM [sales].[orders]
```

```
--SELECT
USE [BikeStores]
SELECT [order_id], [customer_id], [order_status], [order_date]
FROM [sales].[orders]
```

100 %

Results | Messages

	order_id	customer_id	order_status	order_date
1	1	259	4	2016-01-01
2	2	1212	4	2016-01-01
3	3	523	4	2016-01-02
4	4	175	4	2016-01-03
5	5	1324	4	2016-01-03
6	6	94	4	2016-01-04
7	7	324	4	2016-01-04
8	8	1204	4	2016-01-04

Select

Next, suppose if we want to fetch all the records from the customers table. This can be achieved by a simple query as shown below.

```
SELECT * FROM customers;
```

```
SELECT * FROM [sales].[customers]
```

100 %

Results | Messages

	customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	NY	14127
2	2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
4	4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	NY	11553
5	5	Charolette	Rice	(916) 381-6003	charolette.rice@msn.com	107 River Dr.	Sacramento	CA	95820
6	6	Lyndsey	Bean	NULL	lyndsey.bean@hotmail.com	769 West Road	Fairport	NY	14450
7	7	Latasha	Hays	(716) 986-3359	latasha.hays@hotmail.com	7014 Manor Station Rd.	Buffalo	NY	14215
8	8	Jacquline	Duncan	NULL	jacquline.duncan@yahoo.com	15 Brown St.	Jackson Heights	NY	11372
9	9	Genoveva	Baldwin	NULL	genoveva.baldwin@msn.com	8550 Spruce Drive	Port Washington	NY	11050
10	10	Pamelia	Newman	NULL	pamelia.newman@gmail.com	476 Chestnut Ave.	Monroe	NY	10950
11	11	Deshawn	Mendoza	NULL	deshawn.mendoza@yahoo.com	8790 Cobblestone Street	Monsey	NY	10952
12	12	Robby	Sykes	(516) 583-7761	robby.sykes@hotmail.com	486 Rock Maple Street	Hempstead	NY	11550
13	13	Lashawn	Ortiz	NULL	lashawn.ortiz@msn.com	27 Washington Rd	Longview	TX	75604

INSERT

Insert

The basic syntax for writing INSERT statements in SQL is as follows :

By VALUES, we mean the value of the corresponding columns.

Here are a few examples to further illustrate the INSERT statement.

```sql
--INSERT
INSERT INTO [sales].[customers]([first_name],[last_name],
[phone],[email],[street],[city],[state],[zip_code])
VALUES ('Honorio', 'Apaza','998980','hon@gmail.pe',
'new zone','ilo','Peru','001');
```

100 %

Messages

```
(1 row affected)

Completion time: 2022-09-19T17:26:22.8314663-05:00
```

Insert

Suppose if we have to insert values into all the fields of the database table, then we need not specify the column names, unlike the previous query. Follow the following query for further illustration.

In this example, we have successfully inserted all the values without having to specify the fieldnames.

```
INSERT INTO [sales].[customers]
VALUES ('Maricial', 'Mamano','9989480','honee@gmail.pe',
'new zone','are','Peru','001');
```

100 %

Messages

```
(1 row affected)

Completion time: 2022-09-19T17:30:58.1297526-05:00
```

UPDATE

Update

The syntax for writing an UPDATE statement is as follows :

Having learnt the syntax, let us now try an example based on the UPDATE statement in SQL.

```
--UPDATE
UPDATE sales.customers
SET [state] = 'New York'
WHERE [state] = 'NY'
```

100 %

Messages

```
(1019 rows affected)

Completion time: 2022-09-19T21:41:20.8357876-05:00
```

Output

In this example, we have modified the value of store_state for a record where store_state was ‘NY’ and set it to a new value ‘New York’.

```sql
UPDATE sales.customers
SET [state] = 'New York'
WHERE [state] = 'NY'

SELECT*FROM [sales].[customers]
```

100 %

Results | Messages

	customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	New York	14127
2	2	Kasha	Todd	NULL	kasha.todd@yahoo.com	910 Vine Street	Campbell	CA	95008
3	3	Tameka	Fisher	NULL	tameka.fisher@aol.com	769C Honey Creek St.	Redondo Beach	CA	90278
4	4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	New York	11553
5	5	Charolette	Rice	(916) 381-6003	charolette.rice@msn.com	107 River Dr.	Sacramento	CA	95820
6	6	Lyndsey	Bean	NULL	lyndsey.bean@hotmail.com	769 West Road	Fairport	New York	14450
7	7	Latasha	Hays	(716) 986-3359	latasha.hays@hotmail.com	7014 Manor Station Rd.	Buffalo	New York	14215
8	8	Jacquline	Duncan	NULL	jacquline.duncan@yahoo.com	15 Brown St.	Jackson Heights	New York	11372
9	9	Genoveva	Baldwin	NULL	genoveva.baldwin@msn.com	8550 Spruce Drive	Port Washington	New York	11050
10	10	Pamelia	Newman	NULL	pamelia.newman@gmail.com	476 Chestnut Ave.	Monroe	New York	10950
11	11	Deshawn	Mendoza	NULL	deshawn.mendoza@yahoo.com	8790 Cobblestone Street	Monsey	New York	10952
12	12	Robby	Sykes	(516) 583-7761	robby.sykes@hotmail.com	486 Rock Maple Street	Hempstead	New York	11550

—

DELETE

Delete

DELETE statement in SQL is used to remove one or more rows from the database table. It does not delete the data records permanently. We can always perform a rollback operation to undo a DELETE command. With DELETE statements we can use the WHERE clause for filtering specific rows.

```sql
--DELETE
DELETE FROM sales.customers
WHERE [first_name] = 'Honorio'
AND [last_name] = 'Apaza';
```

100 %

Messages

```
(1 row affected)

Completion time: 2022-09-19T21:51:05.3027979-05:00
```

RollBack

Rolls back an explicit or implicit transaction to the beginning of the transaction, or to a savepoint inside the transaction. You can use ROLLBACK TRANSACTION to erase all data modifications made from the start of the transaction or to a savepoint.

```
--rollback
BEGIN TRANSACTION

DELETE FROM sales.customers
WHERE [first_name] = 'Honorio'
AND [last_name] = 'Apaza';

ROLLBACK TRANSACTION
```

100 %

Results | Messages

	customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	1449	Honorio	Apaza	998980	hon@gmail.pe	new zone	ilo	Peru	001

Others examples

Where / Order By

```
SELECT * FROM [sales].[order_items]

WHERE [quantity] >=2 AND [list_price] <= 150

ORDER BY list_price
```

```
SELECT * FROM [sales].[order_items]

WHERE list_price BETWEEN 80 AND 100

ORDER BY list_price
```

```
-- WHERE / ORDER BY
SELECT * FROM [sales].[order_items]
WHERE [quantity] >=2 AND [list_price] <= 150
ORDER BY list_price


SELECT * FROM [sales].[order_items]
WHERE list_price BETWEEN 80 AND 100
ORDER BY list_price
```

100 %

Results | Messages

	order_id	item_id	product_id	quantity	list_price	discount
1	1346	5	263	2	89.99	0.20
2	1379	3	263	2	89.99	0.10
3	1425	2	263	1	89.99	0.07
4	1439	2	263	1	89.99	0.20
5	1462	1	263	1	89.99	0.20
6	1520	4	263	2	89.99	0.10
7	1573	5	263	2	89.99	0.10

LIKE

```
SELECT * FROM sales.customers

WHERE [state] LIKE 'New%'
```

```
SELECT * FROM sales.customers
WHERE [state] LIKE 'New%'
```

100 %

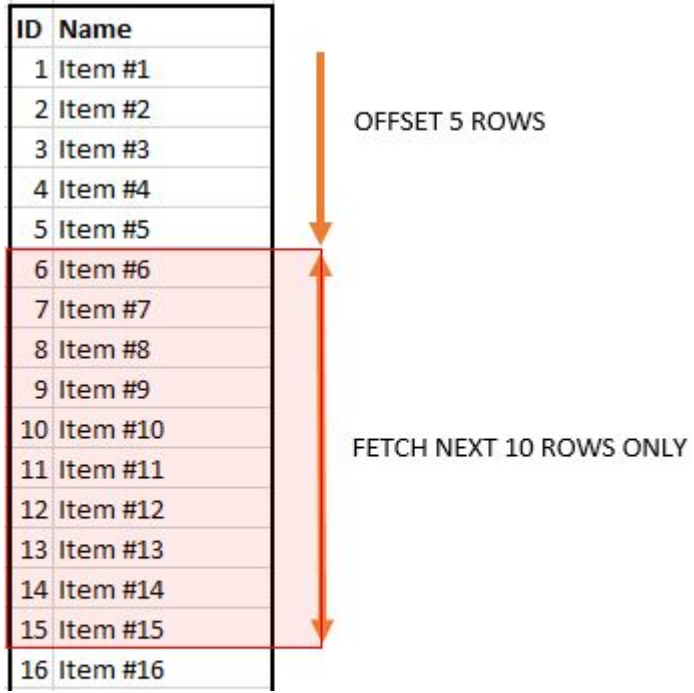
Results | Messages

	customer_id	first_name	last_name	phone	email	street	city	state	zip_code
1	1	Debra	Burks	NULL	debra.burks@yahoo.com	9273 Thorne Ave.	Orchard Park	New York	14127
2	4	Daryl	Spence	NULL	daryl.spence@aol.com	988 Pearl Lane	Uniondale	New York	11553
3	6	Lyndsey	Bean	NULL	lyndsey.bean@hotmail.com	769 West Road	Fairport	New York	14450
4	7	Latasha	Hays	(716) 986-3359	latasha.hays@hotmail.com	7014 Manor Station Rd.	Buffalo	New York	14215
5	8	Jacquline	Duncan	NULL	jacquline.duncan@yahoo.com	15 Brown St.	Jackson Heights	New York	11372
6	9	Genoveva	Baldwin	NULL	genoveva.baldwin@msn.com	8550 Spruce Drive	Port Washington	New York	11050
7	10	Pamelia	Newman	NULL	pamelia.newman@gmail.com	476 Chestnut Ave.	Monroe	New York	10950
8	11	Deshawn	Mendoza	NULL	deshawn.mendoza@yahoo.com	8790 Cobblestone Street	Monsey	New York	10952
9	12	Robby	Sykes	(516) 583-7761	robby.sykes@hotmail.com	486 Rock Maple Street	Hempstead	New York	11550
10	15	Linnie	Branch	NULL	linnie.branch@gmail.com	314 South Columbia Ave.	Plattsburgh	New York	12901
11	16	Emmitt	Sanchez	(212) 945-8823	emmitt.sanchez@hotmail.com	461 Squaw Creek Road	New York	New York	10002
12	17	Caren	Stephens	NULL	caren.stephens@msn.com	914 Brook St.	Scarsdale	New York	10583
13	18	Georgetta	Hardin	NULL	georgetta.hardin@aol.com	474 Chapel Dr.	Canandaigua	New York	14424

OFFSET and FETCH

The OFFSET and FETCH clauses are the options of the ORDER BY clause. They allow you to limit the number of rows to be returned by a query.

```
SELECT
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price,
    product_name
OFFSET 10 ROWS
FETCH NEXT 10 ROWS ONLY;
```



```sql
-- OFFSET AND FETCH
SELECT
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price,
    product_name
```

100 %

Results | Messages

	product_name	list_price
1	Strider Classic 12 Balance Bike - 2018	89.99
2	Sun Bicycles Lil Kitt'n - 2017	109.99
3	Trek Boy's Kickster - 2015/2017	149.99
4	Trek Girl's Kickster - 2017	149.99
5	Trek Kickster - 2018	159.99
6	Trek Precaliber 12 Boys - 2017	189.99
7	Trek Precaliber 12 Girls - 2017	189.99
8	Trek Precaliber 12 Boy's - 2018	199.99
9	Trek Precaliber 12 Girl's - 2018	199.99
10	Haro Shredder 20 - 2017	209.99
11	Haro Shredder 20 Girls - 2017	209.99
12	Trek Precaliber 16 Boy's - 2018	209.99

```sql
-- OFFSET AND FETCH
SELECT
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price,
    product_name
OFFSET 10 ROWS
FETCH NEXT 10 ROWS ONLY;
```

100 %

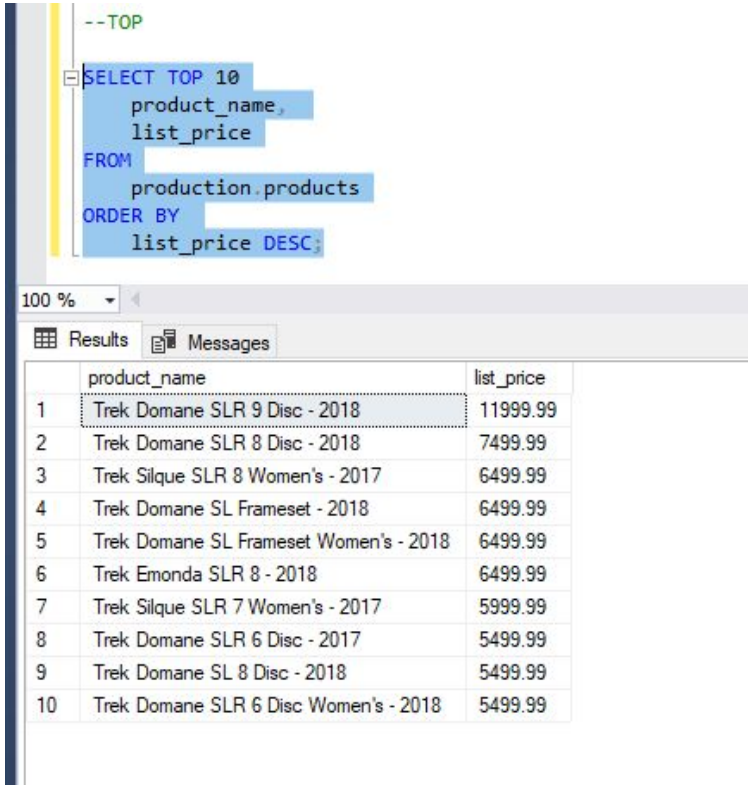
Results | Messages

	product_name	list_price
1	Haro Shredder 20 Girls - 2017	209.99
2	Trek Precaliber 16 Boy's - 2018	209.99
3	Trek Precaliber 16 Boys - 2017	209.99
4	Trek Precaliber 16 Girl's - 2018	209.99
5	Trek Precaliber 16 Girls - 2017	209.99
6	Trek Precaliber 20 Boy's - 2018	229.99
7	Trek Precaliber 20 Girl's - 2018	229.99
8	Haro Shredder Pro 20 - 2017	249.99
9	Strider Sport 16 - 2018	249.99
10	Trek MT 201 - 2018	249.99

Top

The following example uses a constant value to return the top 10 most expensive products.

```
SELECT TOP 10
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price DESC;
```

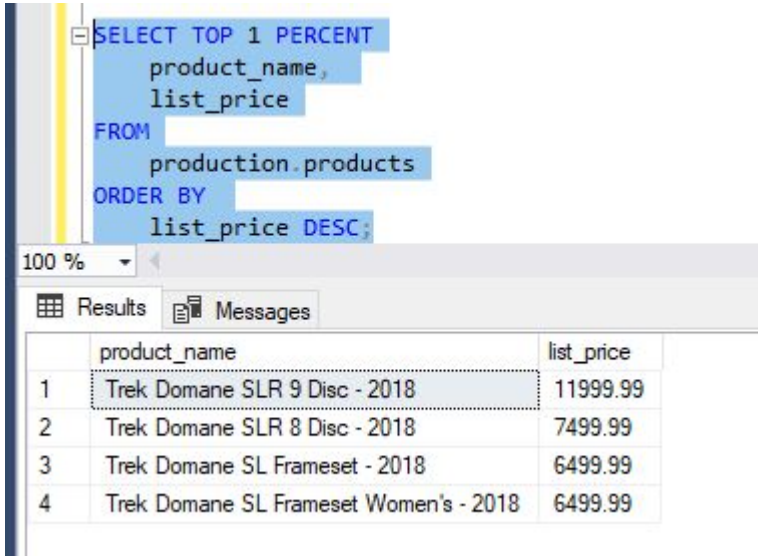


	product_name	list_price
1	Trek Domane SLR 9 Disc - 2018	11999.99
2	Trek Domane SLR 8 Disc - 2018	7499.99
3	Trek Silque SLR 8 Women's - 2017	6499.99
4	Trek Domane SL Frameset - 2018	6499.99
5	Trek Domane SL Frameset Women's - 2018	6499.99
6	Trek Emonda SLR 8 - 2018	6499.99
7	Trek Silque SLR 7 Women's - 2017	5999.99
8	Trek Domane SLR 6 Disc - 2017	5499.99
9	Trek Domane SL 8 Disc - 2018	5499.99
10	Trek Domane SLR 6 Disc Women's - 2018	5499.99

Top

The following example uses PERCENT to specify the number of products returned in the result set. The production.products table has 321 rows, therefore, one percent of 321 is a fraction value (3.21), SQL Server rounds it up to the next whole number which is four (4) in this case.

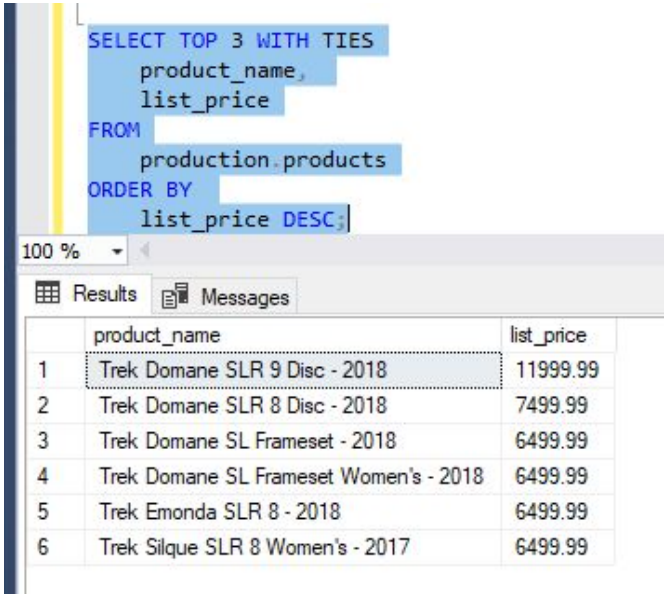
```
SELECT TOP 1 PERCENT
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price DESC;
```



Top

In this example, the third expensive product has a list price of 6499.99. Because the statement used TOP WITH TIES, it returned three more products whose list prices are the same as the third one.

```
SELECT TOP 3 WITH TIES
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price DESC;
```



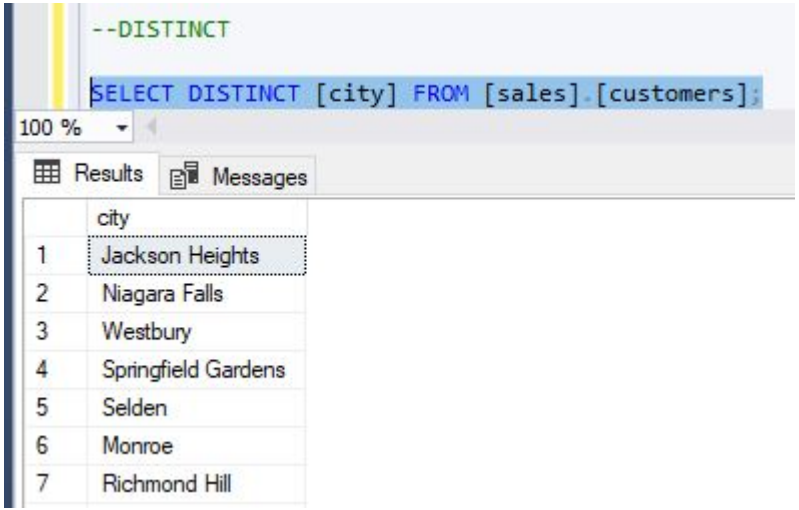
	product_name	list_price
1	Trek Domane SLR 9 Disc - 2018	11999.99
2	Trek Domane SLR 8 Disc - 2018	7499.99
3	Trek Domane SL Frameset - 2018	6499.99
4	Trek Domane SL Frameset Women's - 2018	6499.99
5	Trek Emonda SLR 8 - 2018	6499.99
6	Trek Silque SLR 8 Women's - 2017	6499.99

Distinct

One way to get results without repeating information in SQL Server is to use the different statement in your queries.

Distinct filters the content of one or more columns without repeating the data, taking into account some conditions.

```
SELECT DISTINCT [city] FROM
[sales].[customers];
```



References

1. https://www.sqlservertutorial.net/load-sample-database/
2. https://www.sqlshack.com/sql-ddl-getting-started-with-sql-ddl-commands-in-sql-server/
3. https://www.educba.com/sql-dml-commands/
4. https://github.com/Honorio-apz/BikeStores-DataBase
5. https://github.com/Honorio-apz/DML-DDL-SQL-SERVER

Thank you!

honorio.apz@gmail.com