SQL Server DML Fundamentals

SELECT, INSERT, UPDATE, and DELETE Hands-On Lab

A beginner-friendly tutorial for learning core SQL Server Data Manipulation Language commands using SSMS.

Audience	Beginners learning SQL Server and SSMS
Estimated Time	90 minutes to 2 hours
Tool	SQL Server Management Studio (SSMS)
Lab Database	DMLTrainingDB
Goal	Read, add, modify, and remove table data safely

Safety Reminder: Run this lab only in a test or training SQL Server environment. Never practice UPDATE or DELETE commands against production data.

Lab Objectives

- Understand what DML means and how SELECT, INSERT, UPDATE, and DELETE fit into daily database work.
- Create a small practice database and table in SQL Server.
- Use SELECT to retrieve rows, filter data, sort results, and view specific columns.
- Use INSERT to add one row and multiple rows.
- Use UPDATE to modify specific rows safely with a WHERE clause.
- Use DELETE to remove specific rows safely with a WHERE clause.
- Use transactions with COMMIT and ROLLBACK to practice safer data changes.
- Complete beginner challenge exercises and a final checklist.

Quick Command Summary

Command	Purpose	Beginner Syntax Pattern	Safety Tip
SELECT	Read data	SELECT columns FROM table WHERE condition;	Use WHERE to narrow results.
INSERT	Add rows	INSERT INTO table (columns) VALUES (values);	Always list target columns.
UPDATE	Modify rows	UPDATE table SET column = value WHERE condition;	Preview affected rows first.
DELETE	Remove rows	DELETE FROM table WHERE condition;	Never omit WHERE unless intentional.

Module 1: What Is DML?

DML stands for Data Manipulation Language. In this lab, DML means the commands used to read and change row data in SQL Server tables: SELECT, INSERT, UPDATE, and DELETE.

- SELECT reads data from a table.
- INSERT adds new rows to a table.
- UPDATE changes existing rows in a table.
- DELETE removes rows from a table.

Think of the four commands as CRUD operations: Create = INSERT, Read = SELECT, Update = UPDATE, Delete = DELETE.

Module 2: Build the Training Environment

Open SSMS, connect to a test SQL Server instance, and select New Query. Run the full setup script below.

```
CREATE DATABASE DMLTrainingDB;
GO

USE DMLTrainingDB;
GO

CREATE TABLE dbo.ServiceTickets
```

```
(
    TicketID INT IDENTITY(1,1) PRIMARY KEY,
    TicketTitle VARCHAR(100) NOT NULL,
    AssignedTo VARCHAR(50) NULL,
    Priority VARCHAR(20) NOT NULL,
    TicketStatus VARCHAR(20) NOT NULL,
    CreatedDate DATE NOT NULL DEFAULT (GETDATE()),
    LastUpdated DATETIME NOT NULL DEFAULT (GETDATE())
);
GO

INSERT INTO dbo.ServiceTickets
    (TicketTitle, AssignedTo, Priority, TicketStatus, CreatedDate)
VALUES
    ('Server patching request', 'Noe', 'High', 'Open', '2026-07-01'),
    ('Application restart request', 'Teresa', 'Medium', 'In Progress', '2026-07-02'),
    ('User access review', 'Cathy', 'Low', 'Open', '2026-07-03'),
    ('Database backup validation', 'Gary', 'High', 'Closed', '2026-07-04'),
    ('Linux disk space alert', 'Adam', 'High', 'Open', '2026-07-05');
GO
```

Verify the table was created and populated:

```
SELECT *
FROM dbo.ServiceTickets;
```

Module 3: SELECT - Read Data

SELECT retrieves rows from a database. You can retrieve all columns, specific columns, filtered rows, sorted rows, or a limited number of rows.

Lab 3.1: Select All Columns

```
SELECT *
FROM dbo.ServiceTickets;
```

Use SELECT * only for quick testing. In professional scripts, select only the columns you need.

Lab 3.2: Select Specific Columns

```
SELECT
    TicketID,
    TicketTitle,
    Priority,
    TicketStatus
FROM dbo.ServiceTickets;
```

Lab 3.3: Filter Rows with WHERE

```
SELECT
    TicketID,
    TicketTitle,
    AssignedTo,
    Priority,
    TicketStatus
FROM dbo.ServiceTickets
WHERE Priority = 'High';
```

Lab 3.4: Sort Rows with ORDER BY

```
SELECT
    TicketID,
    TicketTitle,
    Priority,
    CreatedDate
FROM dbo.ServiceTickets
ORDER BY CreatedDate DESC;
```

Lab 3.5: Combine WHERE and ORDER BY

```
SELECT
    TicketID,
    TicketTitle,
    AssignedTo,
    Priority,
    TicketStatus
FROM dbo.ServiceTickets
WHERE TicketStatus = 'Open'
ORDER BY Priority, TicketTitle;
```

Module 4: INSERT - Add New Rows

INSERT adds rows to a table. A strong best practice is to specify the columns you are inserting into so the script is easier to read and more resistant to table design changes.

Lab 4.1: Insert One Row

```
INSERT INTO dbo.ServiceTickets
    (TicketTitle, AssignedTo, Priority, TicketStatus, CreatedDate)
VALUES
    ('Firewall rule review', 'Rudy', 'Medium', 'Open', '2026-07-06');

SELECT *
FROM dbo.ServiceTickets
WHERE TicketTitle = 'Firewall rule review';
```

Lab 4.2: Insert Multiple Rows

```
INSERT INTO dbo.ServiceTickets
    (TicketTitle, AssignedTo, Priority, TicketStatus, CreatedDate)
VALUES
    ('Monthly patch report', 'Tricia', 'Low', 'Open', '2026-07-07'),
    ('Service account validation', 'Eric', 'Medium', 'In Progress', '2026-07-08');

SELECT *
FROM dbo.ServiceTickets
ORDER BY TicketID DESC;
```

Lab 4.3: Insert Using Default Values

```
INSERT INTO dbo.ServiceTickets
    (TicketTitle, AssignedTo, Priority, TicketStatus)
VALUES
    ('Default date test ticket', 'Noe', 'Low', 'Open');
```

```
SELECT *
FROM dbo.ServiceTickets
WHERE TicketTitle = 'Default date test ticket';
```

Notice that CreatedDate and LastUpdated are filled automatically because the table has DEFAULT values.

Module 5: UPDATE - Modify Existing Rows

UPDATE changes values in existing rows. Always preview the rows first with SELECT, then run UPDATE with a specific WHERE clause.

Critical Safety Rule: UPDATE without a WHERE clause can update every row in the table.

Lab 5.1: Preview Before Updating

```
SELECT *
FROM dbo.ServiceTickets
WHERE TicketID = 1;
```

Lab 5.2: Update One Row

```
UPDATE dbo.ServiceTickets
SET
    TicketStatus = 'In Progress',
    LastUpdated = GETDATE()
WHERE TicketID = 1;

SELECT *
FROM dbo.ServiceTickets
WHERE TicketID = 1;
```

Lab 5.3: Update Multiple Columns

```
SELECT *
FROM dbo.ServiceTickets
WHERE TicketTitle = 'User access review';

UPDATE dbo.ServiceTickets
SET
    AssignedTo = 'Noe',
    Priority = 'Medium',
    LastUpdated = GETDATE()
WHERE TicketTitle = 'User access review';

SELECT *
FROM dbo.ServiceTickets
WHERE TicketTitle = 'User access review';
```

Lab 5.4: Update Multiple Rows Carefully

```
SELECT *
FROM dbo.ServiceTickets
WHERE TicketStatus = 'Open'
  AND Priority = 'Low';

UPDATE dbo.ServiceTickets
```

```
SET
    TicketStatus = 'In Progress',
    LastUpdated = GETDATE()
WHERE TicketStatus = 'Open'
  AND Priority = 'Low';

SELECT *
FROM dbo.ServiceTickets
WHERE Priority = 'Low';
```

Module 6: DELETE - Remove Rows

DELETE removes rows from a table. Always preview the exact rows first, then delete using a specific WHERE clause.

Critical Safety Rule: DELETE without a WHERE clause can remove every row in the table.

Lab 6.1: Preview Before Deleting

```
SELECT *
FROM dbo.ServiceTickets
WHERE TicketTitle = 'Default date test ticket';
```

Lab 6.2: Delete One Row

```
DELETE FROM dbo.ServiceTickets
WHERE TicketTitle = 'Default date test ticket';

SELECT *
FROM dbo.ServiceTickets
WHERE TicketTitle = 'Default date test ticket';
```

Lab 6.3: Delete by Primary Key

```
SELECT *
FROM dbo.ServiceTickets
WHERE TicketID = 3;

DELETE FROM dbo.ServiceTickets
WHERE TicketID = 3;

SELECT *
FROM dbo.ServiceTickets
WHERE TicketID = 3;
```

Module 7: Safer Changes with Transactions

A transaction lets you test a data change and either save it with COMMIT or undo it with ROLLBACK.

Lab 7.1: Practice ROLLBACK

```
BEGIN TRANSACTION;

UPDATE dbo.ServiceTickets
SET TicketStatus = 'Closed', LastUpdated = GETDATE()
WHERE Priority = 'High';
```

```
SELECT *
FROM dbo.ServiceTickets
WHERE Priority = 'High';

ROLLBACK TRANSACTION;

SELECT *
FROM dbo.ServiceTickets
WHERE Priority = 'High';
```

Lab 7.2: Practice COMMIT

```
BEGIN TRANSACTION;

UPDATE dbo.ServiceTickets
SET TicketStatus = 'Closed', LastUpdated = GETDATE()
WHERE TicketID = 1;

SELECT *
FROM dbo.ServiceTickets
WHERE TicketID = 1;

COMMIT TRANSACTION;
```

Use ROLLBACK when the result does not look correct. Use COMMIT only after validating the affected rows.

Module 8: Beginner Troubleshooting Tips

Issue	What to Check
Invalid object name	Confirm you are using USE DMLTrainingDB; and that the table exists as dbo.ServiceTickets.
String or binary data would be truncated	One of the text values is longer than the column allows.
Conversion failed	A value does not match the target data type, such as text in a date field.
0 rows affected	The WHERE clause did not match any rows. Run the SELECT preview again.
Too many rows affected	Stop and review the WHERE clause before committing changes.

Challenge Exercises

Complete these exercises without looking at the earlier answer blocks first. Use SELECT to verify each result.

#	Exercise	Done

#	Exercise	Done
1	SELECT only TicketID, TicketTitle, and TicketStatus from dbo.ServiceTickets.	☐
2	SELECT all High priority tickets and sort by CreatedDate newest first.	☐
3	INSERT a new ticket called SQL practice ticket assigned to yourself.	☐
4	UPDATE the SQL practice ticket status from Open to In Progress.	☐
5	UPDATE the SQL practice ticket priority from Low to Medium.	☐
6	DELETE the SQL practice ticket using a specific WHERE clause.	☐
7	Use a transaction to test changing all Open tickets to In Progress, then ROLLBACK.	☐
8	Write a final SELECT query showing the current state of all tickets ordered by TicketID.	☐

Mini Cheat Sheet

```
-- SELECT: read data
SELECT Column1, Column2
FROM dbo.TableName
WHERE Column1 = 'Value'
ORDER BY Column2;

-- INSERT: add data
INSERT INTO dbo.TableName (Column1, Column2)
VALUES ('Value1', 'Value2');

-- UPDATE: modify data
UPDATE dbo.TableName
SET Column1 = 'NewValue'
WHERE PrimaryKeyColumn = 1;

-- DELETE: remove data
DELETE FROM dbo.TableName
WHERE PrimaryKeyColumn = 1;
```

Final Lab Completion Checklist

Skill	Complete
Created the DMLTrainingDB database	☐
Created the dbo.ServiceTickets table	☐
Inserted starter data	☐
Used SELECT * for quick testing	☐
Selected specific columns	☐
Filtered rows with WHERE	☐
Sorted rows with ORDER BY	☐
Inserted one row	☐
Inserted multiple rows	☐
Updated one row safely	☐
Updated multiple columns safely	☐
Deleted one row safely	☐
Used BEGIN TRANSACTION	☐
Used ROLLBACK	☐
Used COMMIT only after verifying	☐
Completed the challenge exercises	☐

References

Microsoft Learn: SELECT (Transact-SQL) - https://learn.microsoft.com/en-us/sql/t-sql/queries/select-transact-sql

Microsoft Learn: INSERT (Transact-SQL) - https://learn.microsoft.com/en-us/sql/t-sql/statements/insert-transact-sql

Microsoft Learn: UPDATE (Transact-SQL) - https://learn.microsoft.com/en-us/sql/t-sql/queries/update-transact-sql

Microsoft Learn: DELETE (Transact-SQL) - https://learn.microsoft.com/en-us/sql/t-sql/statements/delete-transact-sql