SQL Server with SSMS

Relational Database Fundamentals Hands-On Lab

Beginner step-by-step tutorial for creating databases, tables, relationships, queries, JOINs, views, and backups using SQL Server Management Studio.

Audience	Beginners / IT professionals new to SQL Server
Tool	SQL Server Management Studio (SSMS)
Estimated Time	2 to 3 hours
Outcome	Build and query a starter relational database

Training Note: This lab is designed to be safe for a training environment. Use a test SQL Server instance only. Do not run practice scripts against production databases.

Lab Roadmap

Module	Topic	Complete
1	Relational database core concepts	☐
2	Connect to SQL Server using SSMS	☐
3	Create a training database	☐
4	Create tables with primary and foreign keys	☐
5	Insert and verify sample data	☐
6	Query, filter, and sort results	☐
7	Update and delete records safely	☐
8	Use JOINs and aggregate functions	☐
9	Create a view and identity table	☐
10	Back up and restore the database	☐
11	Complete real-world IT inventory challenges	☐

Module 1: Understanding Relational Databases

A relational database stores information in related tables. Tables use rows and columns, and relationships are created by matching key values between tables.

- Database: an organized collection of data such as employees, inventory, tickets, or applications.
- Table: a structured object that stores data in rows and columns.
- Row: one record in a table.
- Column: one field or attribute in a table.
- Primary Key: uniquely identifies each row.
- Foreign Key: connects one table to another table.

DepartmentID	DepartmentName	Purpose
1	IT	Supports technology systems
2	HR	Manages employee services
3	Finance	Handles financial operations

Concept: The Employees table will contain a DepartmentID foreign key that points back to the Departments table. This is how SQL Server represents relationships.

```
Departments
+------------------------+
| DepartmentID  PK       |
| DepartmentName         |
+------------------------+
            |
            | one department can have many employees
            v
Employees
+------------------------+
| EmployeeID    PK       |
| FirstName              |
| LastName               |
| DepartmentID  FK       |
+------------------------+
```

Module 2: Connect to SQL Server Using SSMS

Open SQL Server Management Studio and connect to a SQL Server instance.

- Server type: Database Engine
- Server name: localhost, .\SQLEXPRESS, or your training server name
- Authentication: Windows Authentication if available
- Select Connect

If you do not know the correct server name, ask your database administrator or use a local SQL Server Developer or Express instance in a lab environment.

Module 3: Create Your First Database

In SSMS, select New Query and run the following script.

```
CREATE DATABASE TrainingDB;
GO

USE TrainingDB;
GO
```

After execution, right-click Databases in Object Explorer and select Refresh. You should see TrainingDB.

Module 4: Create Tables

Create the Departments table first because Employees will reference it.

```
CREATE TABLE Departments
(
```

```
    DepartmentID INT PRIMARY KEY,
    DepartmentName VARCHAR(50) NOT NULL
);
GO
```

Now create the Employees table with a foreign key relationship to Departments.

```
CREATE TABLE Employees
(
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    DepartmentID INT NOT NULL,

    CONSTRAINT FK_Employees_Departments
        FOREIGN KEY (DepartmentID)
        REFERENCES Departments(DepartmentID)
);
GO
```

Verify that the tables exist:

```
SELECT TABLE_NAME
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE';
```

Module 5: Insert and Verify Data

Insert department records first, then employees.

```
INSERT INTO Departments (DepartmentID, DepartmentName)
VALUES
(1, 'IT'),
(2, 'HR'),
(3, 'Finance');
GO

INSERT INTO Employees (EmployeeID, FirstName, LastName, DepartmentID)
VALUES
(1001, 'John', 'Smith', 1),
(1002, 'Sarah', 'Davis', 2),
(1003, 'Robert', 'Jones', 3),
(1004, 'Amy', 'Wilson', 1);
GO
```

Verify the data:

```
SELECT * FROM Departments;
SELECT * FROM Employees;
```

Module 6: Query, Filter, and Sort Data

Practice common SELECT statements.

```
-- Select all employee records
SELECT *
FROM Employees;

-- Select specific columns
SELECT FirstName, LastName
FROM Employees;

-- Filter records
SELECT *
FROM Employees
WHERE DepartmentID = 1;

-- Sort results
SELECT *
FROM Employees
ORDER BY LastName;
```

Module 7: Update and Delete Records Safely

Important: Always use a WHERE clause when updating or deleting specific rows. Without a WHERE clause, SQL Server may update or delete every row in the table.

Update Amy Wilson from IT to HR:

```
UPDATE Employees
SET DepartmentID = 2
WHERE EmployeeID = 1004;

SELECT *
FROM Employees
WHERE EmployeeID = 1004;
```

Delete Robert Jones from the training data:

```
DELETE FROM Employees
WHERE EmployeeID = 1003;

SELECT *
FROM Employees;
```

Module 8: JOINs and Aggregate Functions

JOINs combine rows from related tables. This is one of the most important skills in relational databases.

```
SELECT
    e.EmployeeID,
    e.FirstName,
    e.LastName,
    d.DepartmentName
FROM Employees e
INNER JOIN Departments d
    ON e.DepartmentID = d.DepartmentID;
```

Use aggregate functions to summarize data:

```
-- Count all employees
SELECT COUNT(*) AS TotalEmployees
FROM Employees;

-- Count employees by department
SELECT
    d.DepartmentName,
    COUNT(e.EmployeeID) AS TotalEmployees
FROM Departments d
LEFT JOIN Employees e
    ON d.DepartmentID = e.DepartmentID
GROUP BY d.DepartmentName
ORDER BY d.DepartmentName;
```

Module 9: Create a View and an Identity Table

A view acts like a saved query. It can simplify reporting and repeated lookups.

```
CREATE VIEW vwEmployeesByDepartment
AS
SELECT
    e.EmployeeID,
    e.FirstName,
    e.LastName,
    d.DepartmentName
FROM Employees e
INNER JOIN Departments d
    ON e.DepartmentID = d.DepartmentID;
GO

SELECT *
FROM vwEmployeesByDepartment;
```

An identity column automatically generates a numeric value for new rows.

```
CREATE TABLE Tickets
(
```

```
    TicketID INT IDENTITY(1,1) PRIMARY KEY,
    TicketTitle VARCHAR(100) NOT NULL,
    TicketStatus VARCHAR(20) NOT NULL
);
GO

INSERT INTO Tickets (TicketTitle, TicketStatus)
VALUES
('Server Down', 'Open'),
('Patch Required', 'Closed');

SELECT *
FROM Tickets;
```

Module 10: Backup and Restore the Training Database

Backups protect your database and allow you to recover if something goes wrong.

Backup Using SSMS

- Right-click TrainingDB.
- Select Tasks > Back Up.
- Backup type: Full.
- Choose a destination such as C:\SQLBackups\TrainingDB.bak.
- Select OK and confirm success.

Backup Using T-SQL

```
BACKUP DATABASE TrainingDB
TO DISK = 'C:\SQLBackups\TrainingDB.bak'
WITH INIT, NAME = 'TrainingDB Full Backup';
```

Restore Practice Note

> Only restore in a test environment. Restoring a database can overwrite existing data if not done carefully.

- Right-click Databases.
- Select Restore Database.
- Choose Device and browse to the .bak file.
- Restore as a test database name if needed.
- Verify the restored database appears in Object Explorer.

Module 11: Real-World IT Operations Scenario

Practice with an IT inventory scenario. This is similar to tracking servers, applications, versions, owners, environments, and service tickets.

```
CREATE TABLE Servers
(
    ServerID INT IDENTITY(1,1) PRIMARY KEY,
```

```
    ServerName VARCHAR(100) NOT NULL,
    OperatingSystem VARCHAR(100) NOT NULL,
    Environment VARCHAR(30) NOT NULL
);
GO

CREATE TABLE Applications
(
    AppID INT IDENTITY(1,1) PRIMARY KEY,
    AppName VARCHAR(100) NOT NULL,
    VersionNumber VARCHAR(50) NULL,
    OwnerName VARCHAR(100) NULL,
    ServerID INT NOT NULL,

    CONSTRAINT FK_Applications_Servers
        FOREIGN KEY (ServerID)
        REFERENCES Servers(ServerID)
);
GO

INSERT INTO Servers (ServerName, OperatingSystem, Environment)
VALUES
('APPWEB01', 'Windows Server 2022', 'Production'),
('APPLNX01', 'Red Hat Enterprise Linux 9', 'Production'),
('APPTEST01', 'Windows Server 2019', 'Test');

INSERT INTO Applications (AppName, VersionNumber, OwnerName, ServerID)
VALUES
('SQL Server', '2022', 'DBA Team', 1),
('Apache HTTP Server', '2.4', 'Linux Team', 2),
('Tomcat', '10.1', 'Application Team', 2),
('Internal Test App', '1.0', 'ET Apps', 3);
```

Query the inventory using a JOIN:

```
SELECT
    s.ServerName,
    s.OperatingSystem,
    s.Environment,
    a.AppName,
    a.VersionNumber,
    a.OwnerName
FROM Servers s
INNER JOIN Applications a
```

```
    ON s.ServerID = a.ServerID
ORDER BY s.ServerName, a.AppName;
```

Challenge Exercises

#	Exercise	Complete
1	Challenge 1: Create a new table named MaintenanceWindows with MaintenanceID, ServerID, WindowStart, WindowEnd, and Notes.	☐
2	Challenge 2: Add a foreign key from MaintenanceWindows.ServerID to Servers.ServerID.	☐
3	Challenge 3: Insert at least three maintenance window records.	☐
4	Challenge 4: Write a JOIN query that displays ServerName, Environment, WindowStart, WindowEnd, and Notes.	☐
5	Challenge 5: Count how many servers exist by Environment.	☐
6	Challenge 6: Create a view named vwServerApplicationInventory that combines Servers and Applications.	☐

Final Lab Completion Checklist

Task	Complete
Connected to SQL Server using SSMS	☐
Created TrainingDB	☐
Created Departments and Employees tables	☐
Defined primary keys and foreign keys	☐
Inserted sample data	☐
Queried data with SELECT	☐
Filtered data with WHERE	☐

Task	Complete
Sorted data with ORDER BY	☐
Updated a record safely	☐
Deleted a record safely	☐
Created and executed JOIN queries	☐
Used COUNT and GROUP BY	☐
Created a view	☐
Created an identity table	☐
Backed up the database	☐
Reviewed restore process	☐
Completed IT inventory challenge	☐

Key SQL Commands Learned

CREATE DATABASE	**USE**	**CREATE TABLE**
PRIMARY KEY	**FOREIGN KEY**	**INSERT INTO**
SELECT	**WHERE**	**ORDER BY**
UPDATE	**DELETE**	**INNER JOIN**
LEFT JOIN	**COUNT**	**GROUP BY**
CREATE VIEW	**IDENTITY**	**BACKUP DATABASE**

Recommended Next Lessons

- SQL Server security basics: logins, users, roles, and permissions.
- Stored procedures and parameters.
- Indexes and basic query performance.
- SQL Server Agent jobs and automation.
- Monitoring SQL Server health and reviewing error logs.
- Disaster recovery fundamentals: recovery models, backup strategy, restore testing.