SQL Server Joins, Filtering, and Aggregation

Beginner Hands-On Lab Manual Using SSMS and T-SQL

Learn how to combine data from related tables, filter rows with WHERE, group records with GROUP BY, and summarize data with aggregate functions.

Audience	Beginners continuing SQL Server fundamentals
Estimated Time	2 to 3 hours
Tool	SQL Server Management Studio (SSMS)
Lab Database	QueryTrainingDB
Goal	Build practical SELECT queries using JOIN, WHERE, GROUP BY, HAVING, and aggregate functions

Training Safety: Use a test SQL Server instance only. This lab creates and queries a small practice database.

Lab Objectives

- Understand INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN, and CROSS JOIN at a beginner level.
- Use WHERE to filter rows before aggregation.
- Use logical operators such as AND, OR, IN, BETWEEN, LIKE, and IS NULL.
- Use aggregate functions such as COUNT, SUM, AVG, MIN, and MAX.
- Use GROUP BY to summarize rows by category.
- Use HAVING to filter grouped results.
- Combine JOIN, WHERE, GROUP BY, HAVING, and ORDER BY in realistic IT reporting queries.

Quick Concept Reference

Concept	Purpose	Beginner Pattern
JOIN	Combine related tables	FROM A INNER JOIN B ON A.ID = B.ID
WHERE	Filter individual rows	WHERE Environment = 'Production'
GROUP BY	Create summary groups	GROUP BY TeamName
HAVING	Filter grouped results	HAVING COUNT(*) > 2
ORDER BY	Sort final results	ORDER BY TotalTickets DESC

Module 1: Create the Training Database and Sample Tables

This lab uses a small IT operations dataset with teams, servers, applications, and service tickets.

```
CREATE DATABASE QueryTrainingDB;
GO

USE QueryTrainingDB;
GO

CREATE TABLE dbo.Teams
(
    TeamID INT IDENTITY(1,1) CONSTRAINT PK_Teams PRIMARY KEY,
    TeamName VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE dbo.Servers
(
    ServerID INT IDENTITY(1,1) CONSTRAINT PK_Servers PRIMARY KEY,
    ServerName VARCHAR(100) NOT NULL UNIQUE,
    OperatingSystem VARCHAR(100) NOT NULL,
    Environment VARCHAR(30) NOT NULL,
    TeamID INT NOT NULL,
    CONSTRAINT FK_Servers_Teams FOREIGN KEY (TeamID) REFERENCES dbo.Teams(TeamID)
);

CREATE TABLE dbo.Applications
(
```

```
    AppID INT IDENTITY(1,1) CONSTRAINT PK_Applications PRIMARY KEY,
    AppName VARCHAR(100) NOT NULL,
    VersionNumber VARCHAR(50) NULL,
    ServerID INT NOT NULL,
    CONSTRAINT FK_Applications_Servers FOREIGN KEY (ServerID) REFERENCES dbo.Servers(ServerID)
);

CREATE TABLE dbo.ServiceTickets
(
    TicketID INT IDENTITY(1,1) CONSTRAINT PK_ServiceTickets PRIMARY KEY,
    ServerID INT NOT NULL,
    TicketTitle VARCHAR(150) NOT NULL,
    Priority VARCHAR(20) NOT NULL,
    TicketStatus VARCHAR(30) NOT NULL,
    CreatedDate DATE NOT NULL,
    ClosedDate DATE NULL,
    CONSTRAINT FK_ServiceTickets_Servers FOREIGN KEY (ServerID) REFERENCES dbo.Servers(ServerID)
);
GO
INSERT INTO dbo.Teams (TeamName)
VALUES ('ET Applications'), ('Linux Team'), ('Windows Team'), ('Database Team');

INSERT INTO dbo.Servers (ServerName, OperatingSystem, Environment, TeamID)
VALUES
('APPWEB01','Windows Server 2022','Production',1),
('APPLNX01','Red Hat Enterprise Linux 9','Production',2),
('SQLDB01','Windows Server 2022','Production',4),
('APPTEST01','Windows Server 2019','Test',1),
('DEVLNX01','Red Hat Enterprise Linux 8','Development',2);

INSERT INTO dbo.Applications (AppName, VersionNumber, ServerID)
VALUES
('IIS','10.0',1), ('Apache HTTP Server','2.4',2), ('SQL Server','2022',3),
('Tomcat','10.1',2), ('Internal Test App','1.0',4), ('Batch Processor','3.2',5);

INSERT INTO dbo.ServiceTickets (ServerID, TicketTitle, Priority, TicketStatus, CreatedDate, ClosedDate)
VALUES
(1,'Review IIS certificate','High','Open','2026-07-01',NULL),
(2,'Apache restart request','Medium','In Progress','2026-07-02',NULL),
(3,'SQL backup validation','High','Closed','2026-07-03','2026-07-04'),
(4,'Test deployment validation','Low','Open','2026-07-04',NULL),
(5,'Linux disk usage alert','High','Closed','2026-07-05','2026-07-06'),
(2,'Tomcat log review','Medium','Open','2026-07-06',NULL),
(3,'Database index review','Low','Closed','2026-07-07','2026-07-09');
GO
```

Module 2: Filtering Rows with WHERE

WHERE filters individual rows before grouping or aggregation occurs.

```
-- Find production servers
SELECT ServerName, OperatingSystem, Environment
FROM dbo.Servers
WHERE Environment = 'Production';
```

```
-- Find open or in-progress tickets
SELECT TicketID, TicketTitle, Priority, TicketStatus
FROM dbo.ServiceTickets
WHERE TicketStatus IN ('Open', 'In Progress');

-- Find tickets created within a date range
SELECT TicketID, TicketTitle, CreatedDate
FROM dbo.ServiceTickets
WHERE CreatedDate BETWEEN '2026-07-01' AND '2026-07-05';

-- Search by text pattern
SELECT AppName, VersionNumber
FROM dbo.Applications
WHERE AppName LIKE '%SQL%';

-- Find tickets that are not closed yet
SELECT TicketID, TicketTitle, ClosedDate
FROM dbo.ServiceTickets
WHERE ClosedDate IS NULL;
```

Beginner Tip: WHERE filters rows. HAVING filters groups after GROUP BY.

Module 3: Join Related Tables

A JOIN combines rows from two or more tables based on a matching condition. Most beginner SQL Server reports use INNER JOIN or LEFT JOIN.

Lab 3.1: INNER JOIN

```
SELECT
    s.ServerName,
    s.Environment,
    t.TeamName
FROM dbo.Servers s
INNER JOIN dbo.Teams t
    ON s.TeamID = t.TeamID
ORDER BY s.ServerName;
```

Lab 3.2: LEFT JOIN

```
SELECT
    s.ServerName,
    a.AppName,
    a.VersionNumber
FROM dbo.Servers s
LEFT JOIN dbo.Applications a
    ON s.ServerID = a.ServerID
ORDER BY s.ServerName, a.AppName;
```

Lab 3.3: Join Three Tables

```
SELECT
    t.TeamName,
    s.ServerName,
    a.AppName,
```

```
    a.VersionNumber
FROM dbo.Teams t
INNER JOIN dbo.Servers s
    ON t.TeamID = s.TeamID
INNER JOIN dbo.Applications a
    ON s.ServerID = a.ServerID
ORDER BY t.TeamName, s.ServerName, a.AppName;
```

Lab 3.4: Join with Filtering

```
SELECT
    s.ServerName,
    s.Environment,
    st.TicketTitle,
    st.Priority,
    st.TicketStatus
FROM dbo.Servers s
INNER JOIN dbo.ServiceTickets st
    ON s.ServerID = st.ServerID
WHERE s.Environment = 'Production'
  AND st.TicketStatus <> 'Closed'
ORDER BY st.Priority, s.ServerName;
```

Module 4: Aggregation with COUNT, SUM, AVG, MIN, and MAX

Aggregate functions summarize multiple rows into a single value or a value per group.

```
-- Count all tickets
SELECT COUNT(*) AS TotalTickets
FROM dbo.ServiceTickets;

-- Count open tickets
SELECT COUNT(*) AS OpenTickets
FROM dbo.ServiceTickets
WHERE TicketStatus = 'Open';

-- Earliest and latest ticket dates
SELECT
    MIN(CreatedDate) AS EarliestTicketDate,
    MAX(CreatedDate) AS LatestTicketDate
FROM dbo.ServiceTickets;

-- Average days to close completed tickets
SELECT
    AVG(DATEDIFF(DAY, CreatedDate, ClosedDate)) AS AvgDaysToClose
FROM dbo.ServiceTickets
WHERE ClosedDate IS NOT NULL;
```

Module 5: GROUP BY - Summarize by Category

GROUP BY organizes rows into groups so aggregate functions can produce totals per team, environment, status, or priority.

```
-- Count servers by environment
SELECT
    Environment,
```

```
    COUNT(*) AS TotalServers
FROM dbo.Servers
GROUP BY Environment
ORDER BY TotalServers DESC;

-- Count tickets by status
SELECT
    TicketStatus,
    COUNT(*) AS TotalTickets
FROM dbo.ServiceTickets
GROUP BY TicketStatus
ORDER BY TotalTickets DESC;

-- Count applications per server
SELECT
    s.ServerName,
    COUNT(a.AppID) AS TotalApplications
FROM dbo.Servers s
LEFT JOIN dbo.Applications a
    ON s.ServerID = a.ServerID
GROUP BY s.ServerName
ORDER BY TotalApplications DESC;
```

Module 6: HAVING - Filter Grouped Results

HAVING filters aggregate groups, such as servers with more than one application or teams with more than one ticket.

```
-- Servers with more than one application
SELECT
    s.ServerName,
    COUNT(a.AppID) AS TotalApplications
FROM dbo.Servers s
LEFT JOIN dbo.Applications a
    ON s.ServerID = a.ServerID
GROUP BY s.ServerName
HAVING COUNT(a.AppID) > 1
ORDER BY TotalApplications DESC;

-- Teams with two or more tickets
SELECT
    tm.TeamName,
    COUNT(st.TicketID) AS TotalTickets
FROM dbo.Teams tm
INNER JOIN dbo.Servers s
    ON tm.TeamID = s.TeamID
INNER JOIN dbo.ServiceTickets st
    ON s.ServerID = st.ServerID
GROUP BY tm.TeamName
HAVING COUNT(st.TicketID) >= 2
ORDER BY TotalTickets DESC;
```

Remember: WHERE comes before GROUP BY and filters rows. HAVING comes after GROUP BY and filters summarized groups.

Module 7: Real-World IT Reporting Queries

Lab 7.1: Open High-Priority Production Tickets

```
SELECT
    s.ServerName,
    s.Environment,
    st.TicketID,
    st.TicketTitle,
    st.Priority,
    st.TicketStatus
FROM dbo.ServiceTickets st
INNER JOIN dbo.Servers s
    ON st.ServerID = s.ServerID
WHERE s.Environment = 'Production'
  AND st.Priority = 'High'
  AND st.TicketStatus <> 'Closed'
ORDER BY st.CreatedDate;
```

Lab 7.2: Ticket Summary by Team and Status

```
SELECT
    tm.TeamName,
    st.TicketStatus,
    COUNT(st.TicketID) AS TotalTickets
FROM dbo.Teams tm
INNER JOIN dbo.Servers s
    ON tm.TeamID = s.TeamID
INNER JOIN dbo.ServiceTickets st
    ON s.ServerID = st.ServerID
GROUP BY tm.TeamName, st.TicketStatus
ORDER BY tm.TeamName, TotalTickets DESC;
```

Lab 7.3: Production Application Inventory Summary

```
SELECT
    tm.TeamName,
    s.Environment,
    COUNT(DISTINCT s.ServerID) AS TotalServers,
    COUNT(a.AppID) AS TotalApplications
FROM dbo.Teams tm
INNER JOIN dbo.Servers s
    ON tm.TeamID = s.TeamID
LEFT JOIN dbo.Applications a
    ON s.ServerID = a.ServerID
WHERE s.Environment = 'Production'
GROUP BY tm.TeamName, s.Environment
ORDER BY TotalApplications DESC;
```

Challenge Exercises

#	Exercise	Done
1	Write a query that lists all servers and	☐

#	Exercise	Done
	their owning team.	
2	Write a query that lists all applications on Production servers only.	☐
3	Write a query that lists all open tickets with ServerName and TeamName.	☐
4	Count tickets by Priority.	☐
5	Count tickets by TeamName.	☐
6	Show only teams that have two or more tickets using HAVING.	☐
7	Find servers that have more than one application installed.	☐
8	Calculate average days to close tickets that have a ClosedDate.	☐
9	Create a report showing TeamName, Environment, TotalServers, and TotalApplications.	☐
10	Explain in your own words the difference between WHERE and HAVING.	☐

Mini Cheat Sheet

```
-- JOIN pattern
SELECT columns
FROM TableA a
INNER JOIN TableB b
    ON a.KeyColumn = b.KeyColumn;

-- WHERE filters rows
SELECT columns
FROM TableName
WHERE ColumnName = 'Value';

-- GROUP BY summarizes rows
SELECT GroupColumn, COUNT(*) AS TotalRows
FROM TableName
GROUP BY GroupColumn;

-- HAVING filters grouped summaries
SELECT GroupColumn, COUNT(*) AS TotalRows
FROM TableName
```

```
GROUP BY GroupColumn
HAVING COUNT(*) > 1;

-- Common report pattern
SELECT group_columns, aggregate_functions
FROM tables_with_joins
WHERE row_filters
GROUP BY group_columns
HAVING aggregate_filters
ORDER BY sort_columns;
```

Final Lab Completion Checklist

Skill	Complete
Created the QueryTrainingDB database	☐
Created Teams, Servers, Applications, and ServiceTickets tables	☐
Inserted sample data	☐
Filtered rows with WHERE	☐
Used IN, BETWEEN, LIKE, and IS NULL	☐
Used INNER JOIN	☐
Used LEFT JOIN	☐
Joined three tables	☐
Used COUNT, AVG, MIN, and MAX	☐
Grouped rows with GROUP BY	☐
Filtered groups with HAVING	☐
Built IT reporting queries	☐
Completed challenge exercises	☐

References (https://tovartech.org/sql-server.html)

Microsoft Learn: Joins (SQL Server) - https://learn.microsoft.com/en-us/sql/relational-databases/performance/joins

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