**Assignment instructions**

30 minutes to complete

**Step 1:**Create a test table for **Employees** and insert some data. You can use the following SQL script to create a simple table:

1. CREATE TABLE Employees (    ID INT PRIMARY KEY,    Name NVARCHAR(50),    Department NVARCHAR(50));GO
2. INSERT INTO Employees (ID, Name, Department)VALUES (1, 'John Doe', 'HR'),       (2, 'Jane Smith', 'IT'),       (3, 'Mike Johnson', 'Sales');GO

**Step 2:** Open Multiple Query Windows Open multiple query windows in SQL Server Management Studio (SSMS) or any other SQL Server management tool. You will use these windows to simulate concurrent transactions.

**Step 3:** Start a Transaction in Window 1 In the first query window, start a transaction and update a row in the **Employees** table. Execute the following SQL statement:

1. BEGIN TRANUPDATE Employees SET Department = 'Finance' WHERE ID = 1;

**Note:** Keep the transaction open in this window. Do not commit or rollback yet.

**Step 4:**Start Another Transaction in Window 2 In the second query window, start another transaction and attempt to update the same row in the **Employees** table. Execute the following SQL statement:

1. BEGIN TRANUPDATE Employees SET Department = 'IT' WHERE ID = 1;

**Step 5:** Observe Blocking Now, if you switch back to the first query window and try to commit or rollback the transaction, you will notice that it is blocked by the second transaction in the second query window. This is because the second transaction is holding a lock on the row, preventing the first transaction from completing.

**Step 6:** Simulate Timeout To simulate a blocking timeout, you can adjust the lock timeout settings. In the first query window, execute the following SQL statement:

SET LOCK\_TIMEOUT 5000;

**Step 7:** Resolve Blocking To resolve the blocking, you can either wait for the second transaction to commit or rollback in the second query window, or you can manually kill the second transaction. In the second query window, execute the following SQL statement:

KILL <SPID>;

**Step 8:**Complete Transactions In the first query window, either commit or rollback the first transaction. Execute one of the following SQL statements:

To commit the transaction:

COMMIT;

**Assignment Questions**

Did this exercise assist you in grasping the concept of blocking?

Write your answer here:

**How did you do?**

**Compare the instructor's example to your own**

**Instructor example**



[Trevoir Williams](https://www.udemy.com/user/trevoirwilliams/)

Did this exercise assist you in grasping the concept of blocking?

SQL Server blocking occurs when one transaction holds a lock on a resource, such as a table or a row, and another transaction attempts to access the same resource but is blocked from proceeding until the lock is released. This blocking can lead to performance issues and delays in query execution.

In this tutorial, we simulated SQL Server blocking using a simple example. We created a test database called "BlockingTest" with an "Employees" table. We opened multiple query windows to represent concurrent transactions.

In the first query window, we started a transaction and updated a row in the "Employees" table. However, we did not commit or rollback the transaction, leaving it open.

In the second query window, we started another transaction and attempted to update the same row in the "Employees" table. Since the first transaction held a lock on the row, the second transaction was blocked from proceeding.

We observed the blocking by trying to commit or rollback the first transaction in the first query window, but it was blocked by the second transaction in the second query window.

To simulate a timeout, we adjusted the lock timeout settings in the first query window. We set the lock timeout to 5 seconds, and if we tried to commit or rollback the first transaction within that time, we received a timeout error.

To resolve the blocking, we either waited for the second transaction to complete or manually killed it by executing the KILL <SPID> statement in the second query window, where <SPID> represents the Server Process ID of the second transaction.

Finally, we completed the transactions by either committing or rolling back the first transaction in the first query window.

It's important to note that this tutorial showcases a basic blocking scenario. In real-world scenarios, blocking can involve multiple transactions, different types of locks, and more complex interactions. Additionally, caution should be exercised when killing transactions, as it may have unintended consequences.