TRANSACTIONS ASSIGNMENT

- **Task A: Dirty Read (READ UNCOMMITTED vs READ COMMITTED)**

Isolation Level: READ UNCOMMITTED:

```sql
-- Task A: Dirty Read
BEGIN TRANSACTION;

UPDATE Rooms
SET Price = 100.00
WHERE RoomID = 1;

SELECT * FROM Rooms WHERE RoomID = 1;
```

Results | Messages

RoomID	RoomNumber	Type	Price
1	101	Single	100.00

```sql
-- 1. Using READ UNCOMMITTED

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
SELECT * FROM Rooms WHERE RoomID = 1;
```

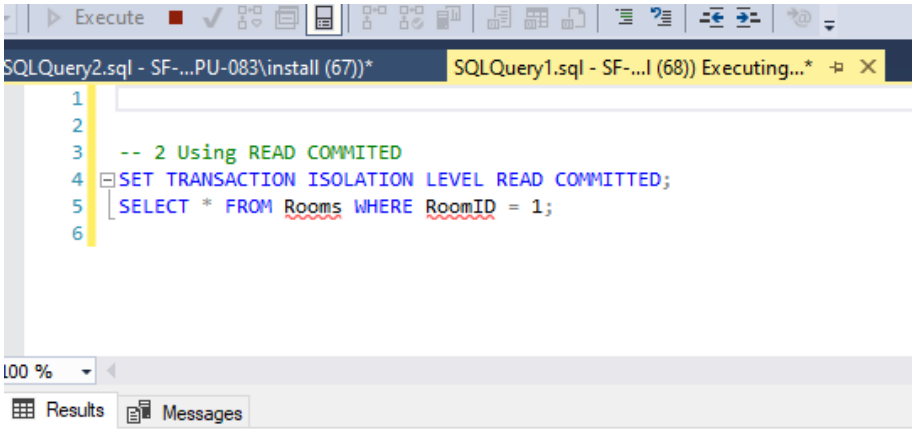
100 %

Results | Messages

	RoomID	RoomNumber	Type	Price
1	1	101	Single	100.00

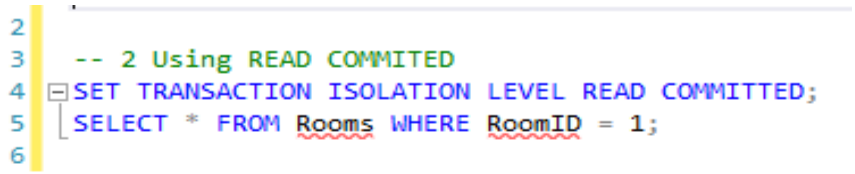
Since `READ UNCOMMITTED` allows dirty reads, **Session 2 sees the updated price** even though it is uncommitted.

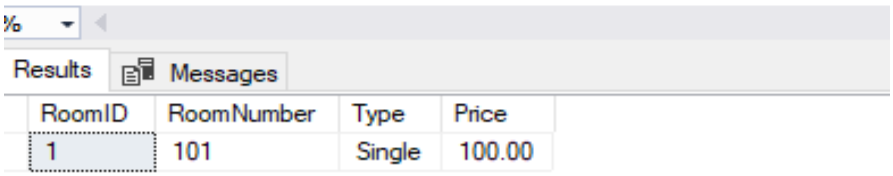
Isolation Level: READ COMMITTED:



If **Session 1** has an **open transaction (not committed or rolled back)**, **Session 2 must wait** until the transaction is completed. `READ COMMITTED` in **Session 2** waits for the exclusive lock to be released before reading.

After commit-





- **Task B: Non-Repeatable Read (READ COMMITTED vs REPEATABLE READ)**

Isolation Level: READ COMMITTED:

SQLQuery2.sql - SF-...PU-083\install (67))* | SQLQuery1.sql - SF-...PU-083\install (68))

```sql
-- Task B: Non-Repeatable Read
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN TRANSACTION;
SELECT BookingStatus FROM Bookings WHERE BookingID = 4;
```

100 %

Results | Messages

	BookingStatus
1	Confirmed

SQLQuery2.sql - SF-...PU-083\install (67))* | SQLQuery1.sql - SF-...PU-083\install (68))*

```sql
--B
--Scenario 1: READ COMMITTED
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN TRANSACTION;
UPDATE Bookings SET BookingStatus = 'Cancelled' WHERE BookingID = 4
COMMIT;
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-02-25T17:00:19.3825599+05:30
```

SQLQuery2.sql - SF-...PU-083\install (67))* | SQLQuery1.sql - SF-...PU-083\install (68))

```sql
-- Task B: Non-Repeatable Read
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
BEGIN TRANSACTION;
SELECT BookingStatus FROM Bookings WHERE BookingID = 4;
```

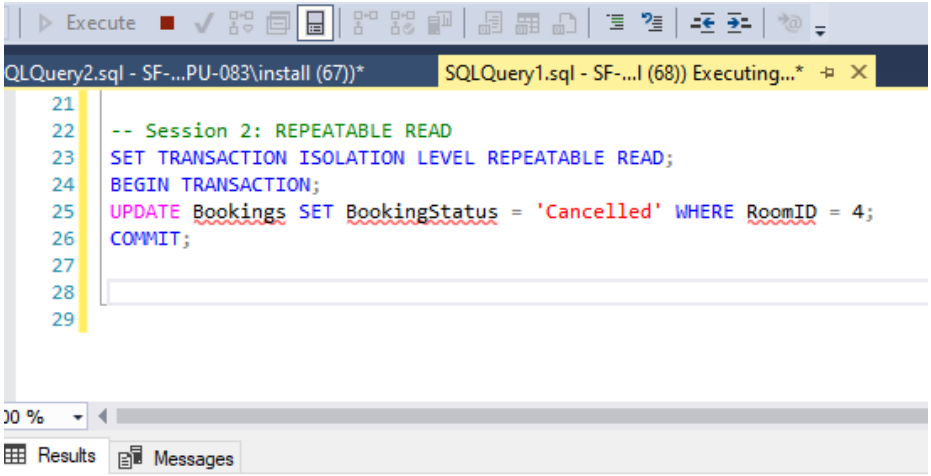
100 %

Results | Messages

	BookingStatus
1	Cancelled

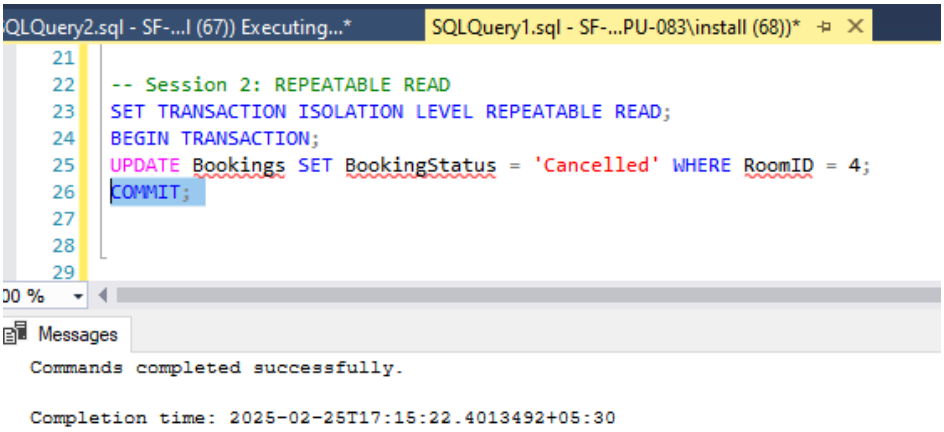
The **READ COMMITTED** isolation level ensures that **Session 1** can read **committed data**. Once **Session 2** commits the change (status update to 'Cancelled'), **Session 1** will see the new status immediately in its next read.

Isolation Level: REPEATABLE READ:



The reason **Session 2**'s update statement is still running (blocked) is due to the fact that **REPEATABLE READ** isolation level ensures **consistent reads** by locking the data for the duration of **Session 1**'s transaction. Once **Session 1** commits, **Session 2** will be able to acquire the lock and proceed with its update.

After commit in session 1-



- **Task C: Phantom Read (REPEATABLE READ vs SERIALIZABLE)**

Isolation Level: REPEATABLE READ:

```sql
-- Task C: Phantom Read

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
BEGIN TRANSACTION;

SELECT COUNT(*) FROM Rooms WHERE Type = 'Suite';
```

100 %

Results | Messages

	(No column name)
1	3

SQLQuery2.sql - SF-...PU-083\install (67))* | SQLQuery1.sql - SF-...PU-083\install (68))*

```sql
-- C
-- Session 1: REPEATABLE READ
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
BEGIN TRANSACTION;

INSERT INTO Rooms (RoomNumber, Type, Price) VALUES ('111', 'Suite', 15
COMMIT;
```

100 %

Messages

```
Commands completed successfully.

Completion time: 2025-02-25T17:22:32.0027143+05:30
```

```sql
-- Task C: Phantom Read

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
BEGIN TRANSACTION;

SELECT COUNT(*) FROM Rooms WHERE Type = 'Suite';
```

	(No column name)
1	4

Isolation Level: SERIALIZABLE:

```sql
-- C
-- Session 2: SERIALIZABLE
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;

INSERT INTO Rooms (RoomNumber, Type, Price) VALUES ('111', 'Suite', 150.00);
COMMIT;
```

```
The statement has been terminated.
Query was canceled by user.
```

Session 2 is **blocked** from inserting the new row into the **'Suite'** category because the **SERIALIZABLE** lock held by **Session 1** prevents any changes to rows that could affect the result of the query in **Session 1**.