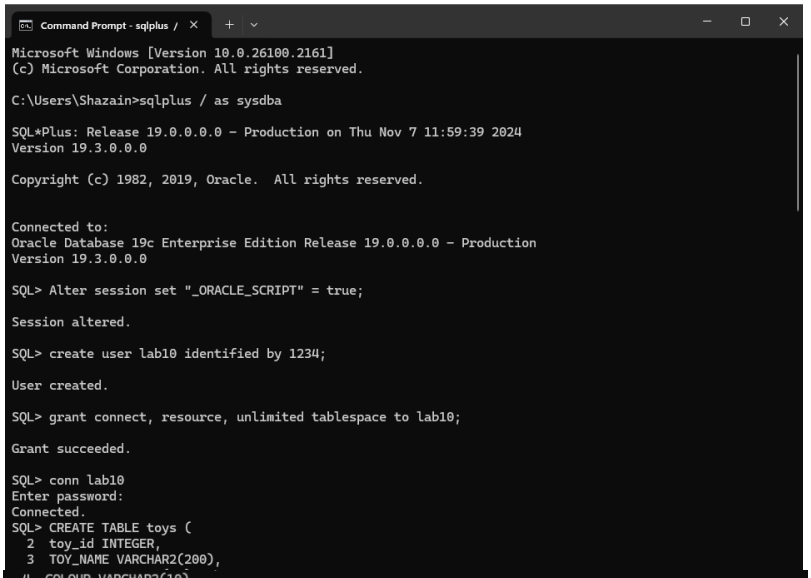
Database Systems
Lab 10
Shazain

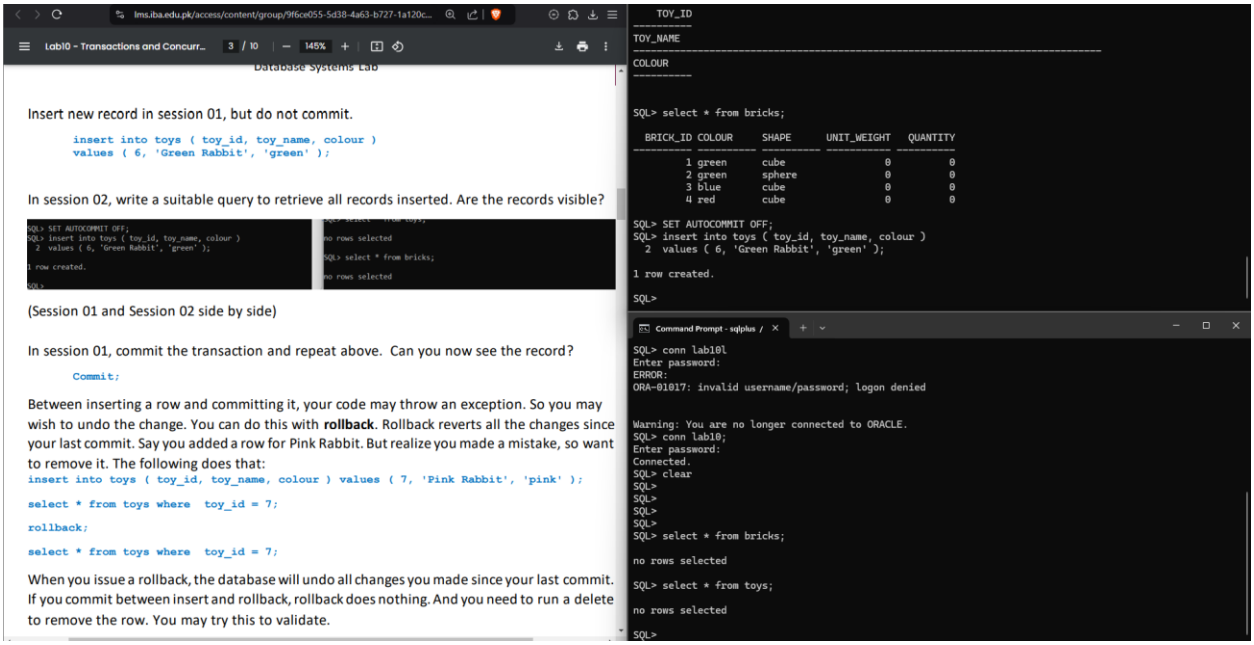
1. **Creating Tables:**



```
Command Prompt - sqlplus /

Microsoft Windows [Version 10.0.26100.2161]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shazain>sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Thu Nov 7 11:59:39 2024
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> Alter session set "_ORACLE_SCRIPT" = true;

Session altered.

SQL> create user lab10 identified by 1234;

User created.

SQL> grant connect, resource, unlimited tablespace to lab10;

Grant succeeded.

SQL> conn lab10
Enter password:
Connected.
SQL> CREATE TABLE toys (
  2  toy_id INTEGER,
  3  TOY_NAME VARCHAR2(200),
  4  COLOUR VARCHAR2(10)
  5  );

Table created.

SQL> create table bricks (
  2   brick_id integer,
  3   colour varchar2(10),
  4   shape varchar2(10),
  5   unit_weight integer DEFAULT 0,
  6   quantity integer DEFAULT 0
  7  );

Table created.

SQL> DESC TOYS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 TOY_ID                                             NUMBER(38)
 TOY_NAME                                           VARCHAR2(200)
 COLOUR                                             VARCHAR2(10)

SQL> DESC BRICKS;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BRICK_ID                                           NUMBER(38)
 COLOUR                                             VARCHAR2(10)
 SHAPE                                              VARCHAR2(10)
 UNIT_WEIGHT                                        NUMBER(38)
 QUANTITY                                           NUMBER(38)

SQL> insert into toys values ( 1, 'Fluffy', 'pink' );

1 row created.

SQL> insert into toys values ( 2, 'Baby Turtle', 'green' );

1 row created.

SQL> insert into bricks (brick_id, colour, shape) values ( 1, 'green', 'cube' );

1 row created.

SQL> insert into bricks (brick_id, colour, shape) values ( 2, 'green', 'sphere' );

1 row created.

SQL> insert into bricks (brick_id, colour, shape) values ( 3, 'blue', 'cube' );

1 row created.
```

2. Committing Changes:

Before committing :



Database Systems Lab

Insert new record in session 01, but do not commit.

```
insert into toys ( toy_id, toy_name, colour )
values ( 6, 'Green Rabbit', 'green' );
```

In session 02, write a suitable query to retrieve all records inserted. Are the records visible?

(Session 01 and Session 02 side by side)

In session 01, commit the transaction and repeat above. Can you now see the record?

```
Commit;
```

Between inserting a row and committing it, your code may throw an exception. So you may wish to undo the change. You can do this with **rollback**. Rollback reverts all the changes since your last commit. Say you added a row for Pink Rabbit. But realize you made a mistake, so want to remove it. The following does that:

```
insert into toys ( toy_id, toy_name, colour ) values ( 7, 'Pink Rabbit', 'pink' );

select * from toys where  toy_id = 7;

rollback;

select * from toys where  toy_id = 7;
```

When you issue a rollback, the database will undo all changes you made since your last commit. If you commit between insert and rollback, rollback does nothing. And you need to run a delete to remove the row. You may try this to validate.

```
SQL> select * from bricks;

  BRICK_ID COLOUR     SHAPE      UNIT_WEIGHT   QUANTITY
---------- ---------- ---------- ----------- ----------
         1 green      cube                 0          0
         2 green      sphere               0          0
         3 blue       cube                 0          0
         4 red        cube                 0          0

SQL> SET AUTOCOMMIT OFF;
SQL> insert into toys ( toy_id, toy_name, colour )
  2  values ( 6, 'Green Rabbit', 'green' );

1 row created.

SQL>
```

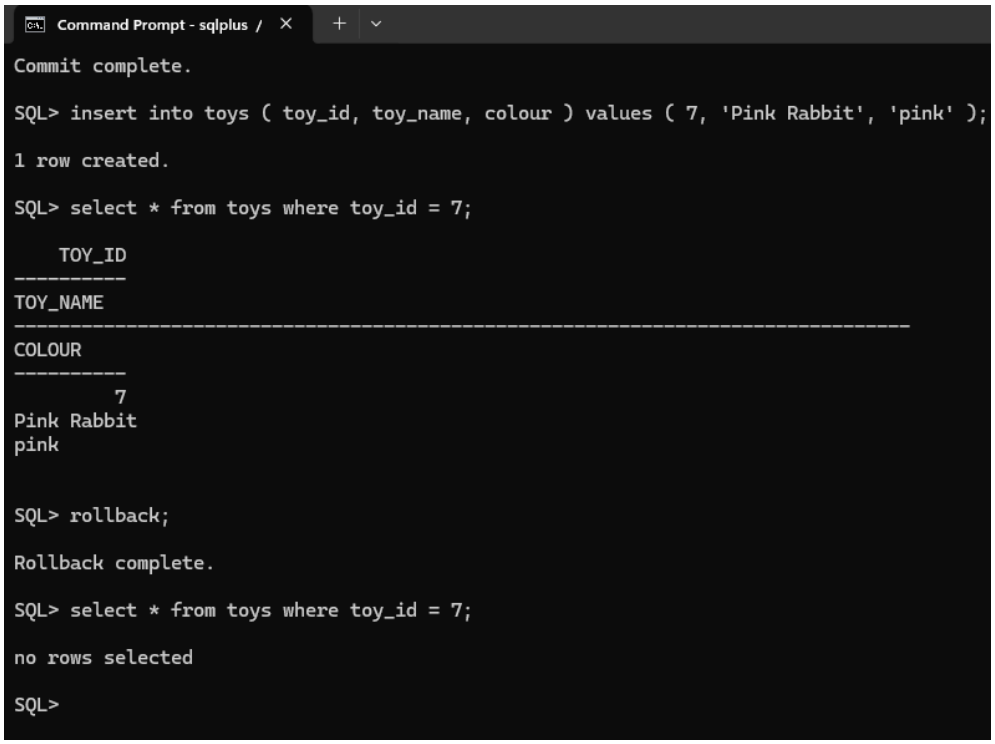
```
Command Prompt - sqlplus /

SQL> conn lab10l
Enter password:
ERROR:
ORA-01017: invalid username/password; logon denied


Warning: You are no longer connected to ORACLE.
SQL> conn lab10;
Enter password:
Connected.
SQL> clear
SQL>
SQL>
SQL>
SQL>
SQL> select * from bricks;

no rows selected

SQL> select * from toys;

no rows selected

SQL>
```

After committing:

The records can be seen from session 2.

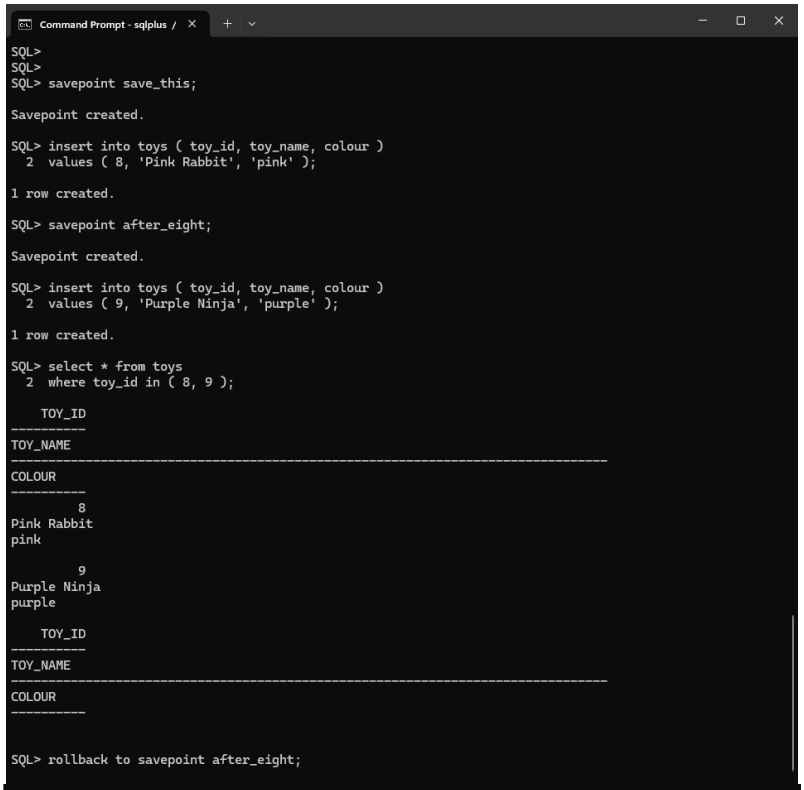
```
Command Prompt - sqlplus /

no rows selected

SQL> select * from bricks;

  BRICK_ID COLOUR     SHAPE      UNIT_WEIGHT   QUANTITY
---------- ---------- ---------- ----------- ----------
         1 green      cube                 0          0
         2 green      sphere               0          0
         3 blue       cube                 0          0
         4 red        cube                 0          0

SQL> select * from toys;

    TOY_ID
----------
TOY_NAME
--------------------------------------------------------------------------------
COLOUR
----------
         1
Fluffy
pink

         2
Baby Turtle
green

    TOY_ID
----------
TOY_NAME
--------------------------------------------------------------------------------
COLOUR
----------

         6
Green Rabbit
green


SQL>
```

Rollback in session 1:



3. Savepoints

A row is added into toys with the id as 8. A savepoint is created and another row is added with id 9. The records are then retrieved. The rollback statement is executed which changes state to the previous savepoint. The records are then retrieved again but nothing returns.



4. **Updating table:**

```
Command Prompt - sqlplus /

SQL>
SQL> update bricks
  2  set quantity = quantity - 10
  3  where colour = 'green'
  4  and shape = 'cube';

1 row updated.

SQL> commit;

Commit complete.

SQL> update bricks
  2  set quantity = quantity + 10
  3  where colour = 'blue'
  4  and shape = 'cube';

1 row updated.

SQL>
SQL> commit;

Commit complete.

SQL> select * from bricks;

  BRICK_ID COLOUR     SHAPE      UNIT_WEIGHT   QUANTITY
---------- ---------- ---------- ----------- ----------
         1 green      cube                 0        -10
         2 green      sphere               0          0
         3 blue       cube                 0         10
         4 red        cube                 0          0
```

5. **Simulating a Deadlock**

The first session throws the error while the second session becomes unresponsive. After committing in the first session, the statement in the second session executes. The statement which set quantity to 1722 did not run.

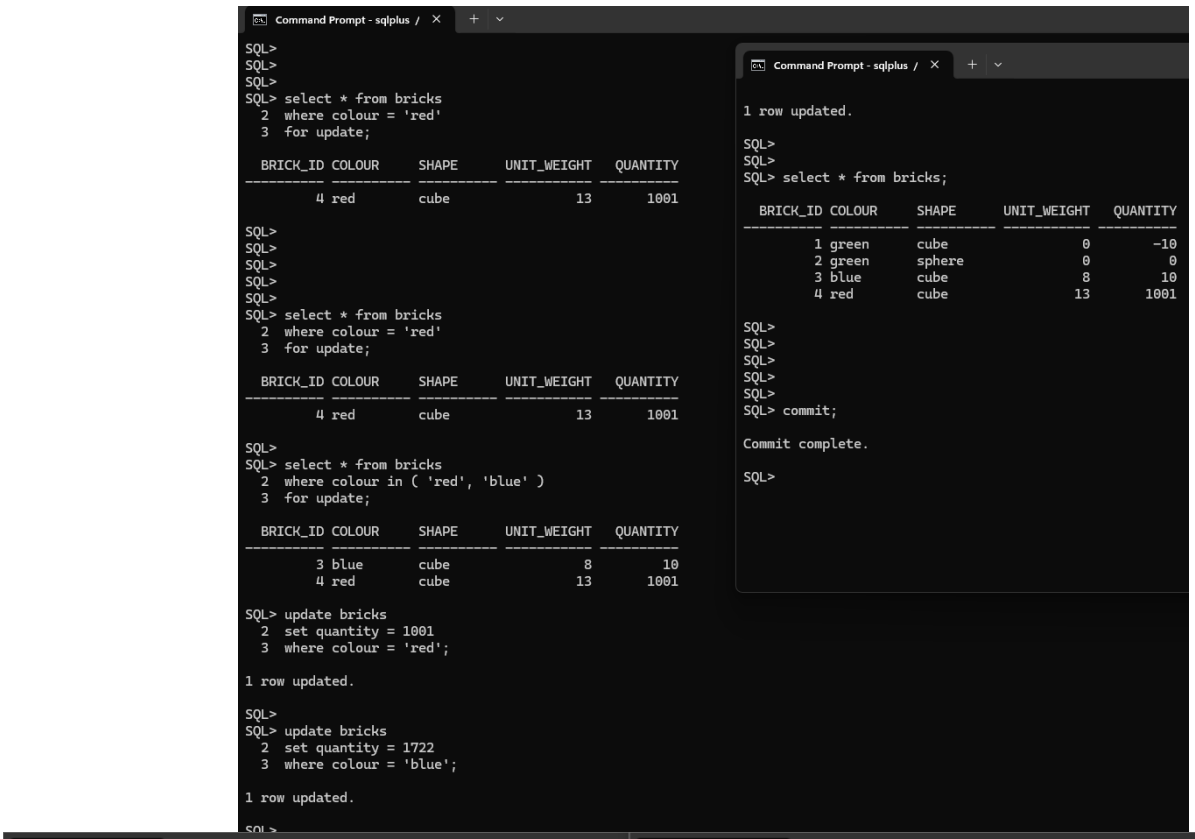
```
Command Prompt - sqlplus /

         3 blue       cube                 0         10
         4 red        cube                 0          0

SQL>
SQL>
SQL> update bricks
  2  set quantity = 1001
  3  where colour = 'red';

1 row updated.

SQL> update bricks
  2  set quantity = 1722
  3  where colour = 'blue';
set quantity = 1722
    *
ERROR at line 2:
ORA-00060: deadlock detected while waiting for resource


SQL>
SQL>
SQL>
SQL>
```

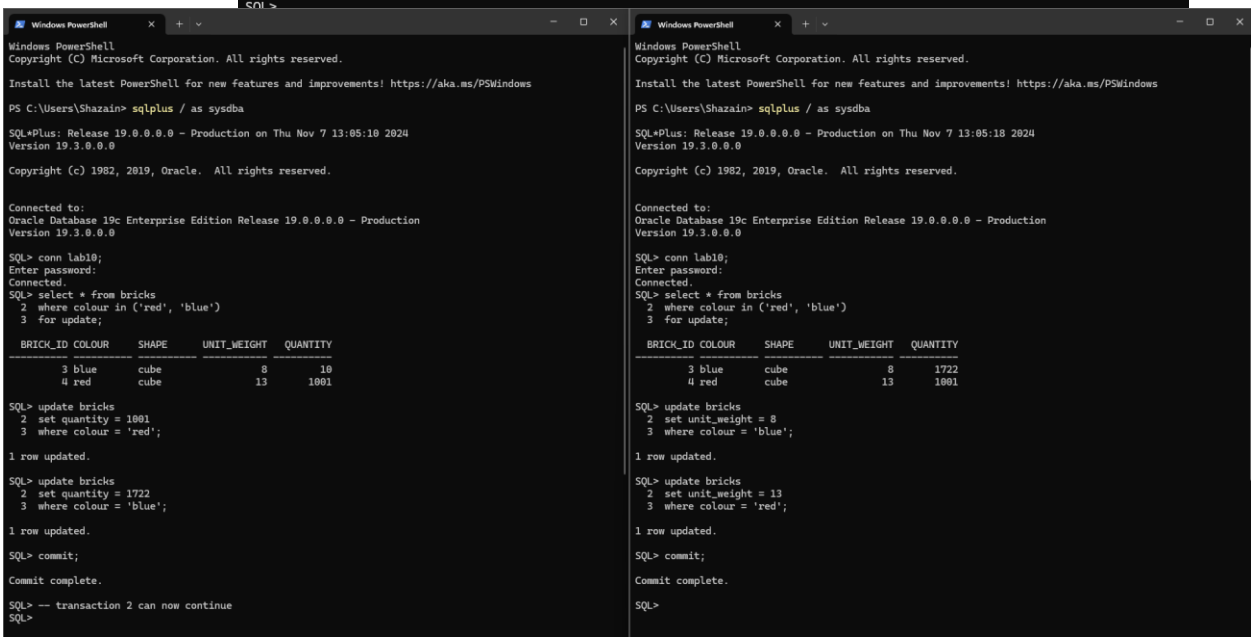
```
Command Prompt - sqlplus /

SQL>
SQL> update bricks
  2  set quantity = 1001
  3  where colour = 'red';

1 row updated.

SQL> update bricks
  2  set quantity = 1722
  3  where colour = 'blue';
set quantity = 1722
    *
ERROR at line 2:
ORA-00060: deadlock detected while waiting for resource


SQL>
SQL>
SQL>
SQL> commit;

Commit complete.

SQL>
```

```
Command Prompt - sqlplus /

SQL> select * from bricks;

  BRICK_ID COLOUR     SHAPE      UNIT_WEIGHT   QUANTITY
---------- ---------- ---------- ----------- ----------
         1 green      cube                 0          0
         2 green      sphere               0          0
         3 blue       cube                 0          0
         4 red        cube                 0          0

SQL> clear
SQL>
SQL>
SQL>
SQL> update bricks
  2  set unit_weight = 8
  3  where colour = 'blue';

1 row updated.

SQL> update bricks
  2  set unit_weight = 13
  3  where colour = 'red';
```

```
Command Prompt - sqlplus /

SQL> update bricks
  2  set unit_weight = 8
  3  where colour = 'blue';

1 row updated.

SQL> update bricks
  2  set unit_weight = 13
  3  where colour = 'red';

1 row updated.

SQL>
SQL>
SQL> select * from bricks;

  BRICK_ID COLOUR     SHAPE      UNIT_WEIGHT   QUANTITY
---------- ---------- ---------- ----------- ----------
         1 green      cube                 0        -10
         2 green      sphere               0          0
         3 blue       cube                 8         10
         4 red        cube                13       1001

SQL>
```

6. Select FOR UPDATE

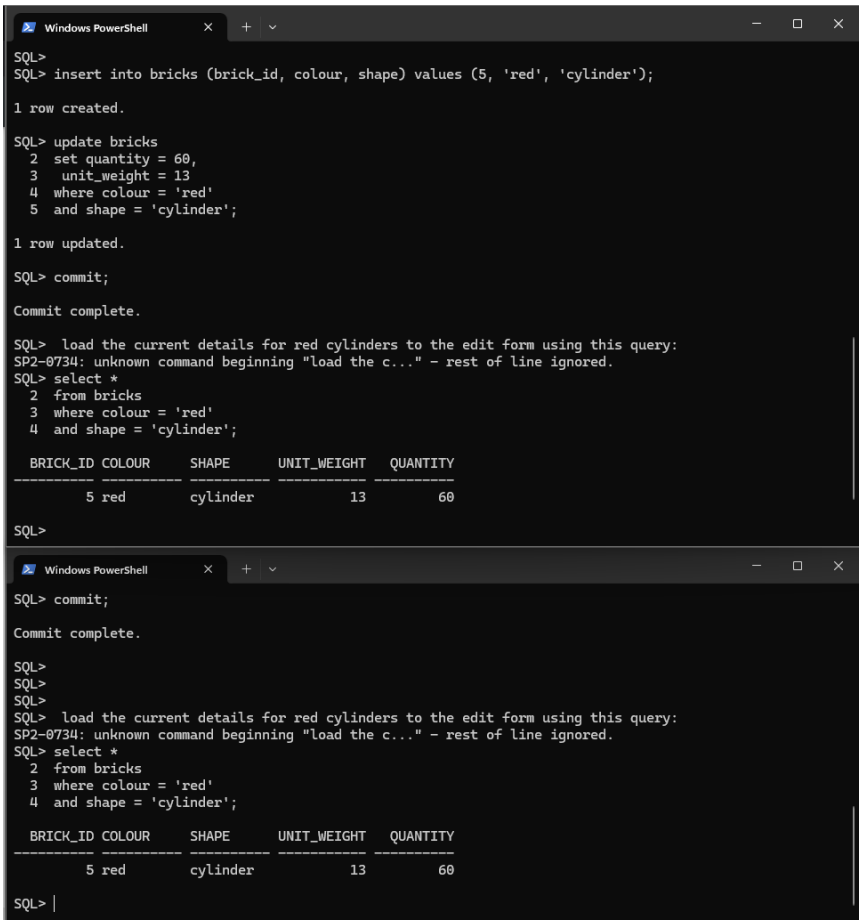




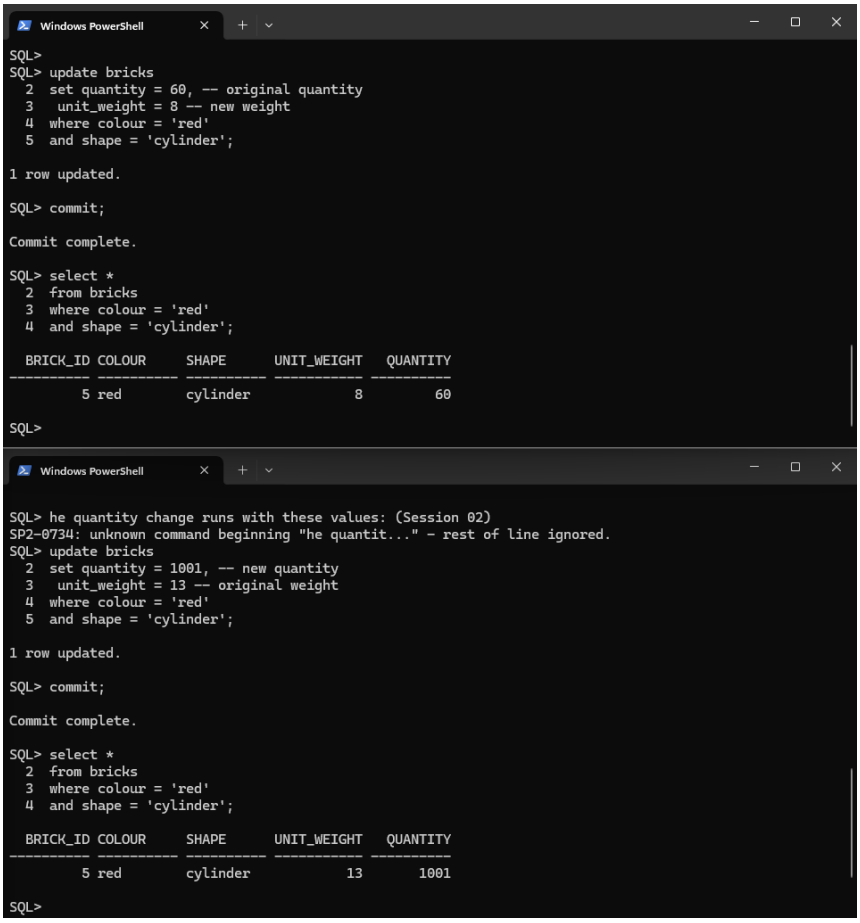
By replicating the instructions in the table, the second session became unresponsive when the second transaction took place.

7. Lost Update

Both of the sessions can see the same table.

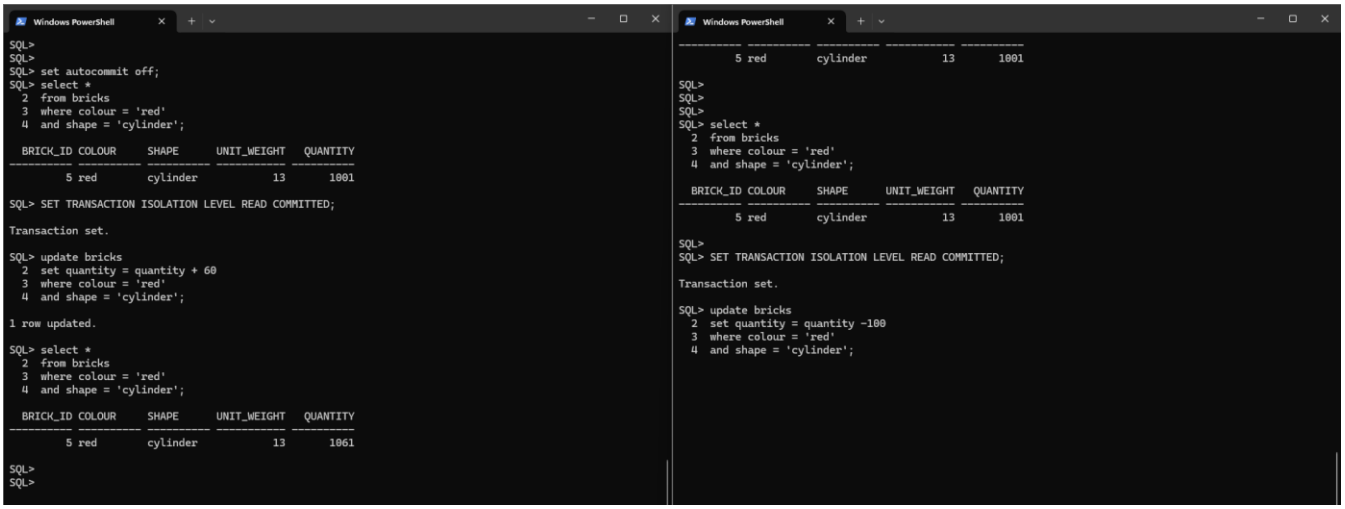


Lost update

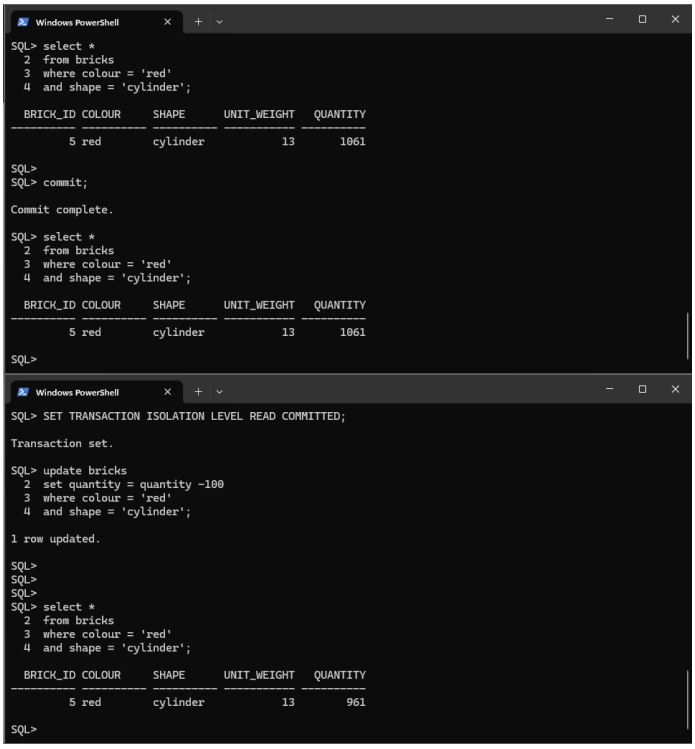


8. Isolation Levels

The second session is blocked.



Session 2 unblocks after commit; in session 1.



A problem still exists. Although Read Committed prevents dirty reads, it does not prevent non-repeatable reads, which could lead to inconsistent data like in this case. Session 2 was blocked from updating before session 1 commits but the values are inconsistent.

A commit is needed in session 2. It will update the quantity and make it visible in other sessions. It session 2 is closed without committing, the non-updated valued will be lost.

60 units were added then 100 units were reduced. The final value is 961.