

1 1 ✓

```
CREATE TABLE accounts (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    balance DECIMAL(10, 2) DEFAULT 0.00
);

CREATE TABLE products (
    id SERIAL PRIMARY KEY,
    shop VARCHAR(100) NOT NULL,
    product VARCHAR(100) NOT NULL,
    price DECIMAL(10, 2) NOT NULL
);

INSERT INTO accounts (name, balance) VALUES
    (name 'Alice', balance 1000.00),
    (name 'Bob' balance 500.00)
```

```
s
[2025-12-01 21:41:48] lab10.public> CREATE TABLE accounts (
                                id SERIAL PRIMARY KEY,
                                name VARCHAR(100) NOT NULL,
                                balance DECIMAL(10, 2) DEFAULT 0.00
)
[2025-12-01 21:41:48] completed in 38 ms
[2025-12-01 21:41:48] lab10.public> CREATE TABLE products (
                                id SERIAL PRIMARY KEY,
                                shop VARCHAR(100) NOT NULL,
                                product VARCHAR(100) NOT NULL,
                                price DECIMAL(10, 2) NOT NULL
)
[2025-12-01 21:41:48] completed in 6 ms
```

s

① >

- Database
- postgres
- lab8_j
- a [ses]
- b [ses]
- lab9_j
- lab8_i
- lab10_

```
INSERT INTO accounts (name, balance) VALUES
    ( name 'Alice', balance 1000.00),
    ( name 'Bob', balance 500.00),
    ( name 'Wally', balance 750.00);

INSERT INTO products (shop, product, price) VALUES
    ( shop 'Joe''s Shop', product 'Coke', price 2.50),
    ( shop 'Joe''s Shop', product 'Pepsi', price 3.00);
```



```
-- Task 1: Basic Transaction with COMMIT
```

```
BEGIN;
UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';
UPDATE accounts SET balance = balance + 100
```

```
es
```

```
+ ⚡ > 01 21:41:48] completed in 6 ms
01 21:45:01] lab10.public> INSERT INTO accounts (name, balance) VALUES
    ('Alice', 1000.00),
    ('Bob', 500.00),
    ('Wally', 750.00)
01 21:45:01] 3 rows affected in 35 ms
⌚ b [ses 01 21:45:01] lab10.public> INSERT INTO products (shop, product, price) VALUES
    ('Joe''s Shop', 'Co
    ('Joe''s Shop', 'Pe
01 21:45:01] 2 rows affected in 4 ms
```

```
-- Task 1: Basic Transaction with COMMIT  
BEGIN;  
UPDATE accounts SET balance = balance - 100.00  
    WHERE name = 'Alice';  
UPDATE accounts SET balance = balance + 100.00  
    WHERE name = 'Bob';  
COMMIT;
```

```
SELECT name, balance FROM accounts WHERE name IN ('Alice','Bob');
```



- a) Alice: 900.00, Bob: 600.00.
 - b) atomicity – both updates succeed together or not at all.
 - c) Without a transaction, partial update could persist leaving inconsistent state
- */

```
-- Task 2: Using ROLLBACK
```

```
s
```

Output lab10.public.accounts

Database

	name	balance
1	Alice	900.00
2	Bob	600.00

postgres
lab8_i
a [ses
b [ses
lab9_i
lab8_i
lab10

```
-- Task 2: Using ROLLBACK  
BEGIN;  
UPDATE accounts SET balance = balance - 500.00  
    WHERE name = 'Alice';  
SELECT * FROM accounts WHERE name = 'Alice';  
  
ROLLBACK;  
  
SELECT * FROM accounts WHERE name = 'Alice';
```



- a) After UPDATE before ROLLBACK you see decreased balance in this session
 - b) After ROLLBACK balance returns to original
 - c) Use ROLLBACK when an error occurs or validation fails
- */

ces

+

Output lab10.public.accounts lab10.public.accounts 2

Database

postgres

	id	name	balance
a [ses	1	Alice	900.00
b [ses			
lab9_i			
lab8_i			
lab10			

1 ✓ 1 ^

```
-- Task 3: Working with SAVEPOINTS
BEGIN;
UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';

SAVEPOINT my_savepoint;

UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Bob';

ROLLBACK TO my_savepoint;

UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Wally';

COMMIT;

/*
a) Alice: decreased by 100 Bob: unchanged Wally: increased by 100
b) Bob was not credited in final state because we rolled back the step that credited Bob before commit.
c) SAVEPOINT allows partial undo inside a long transaction without aborting everything.
*/
```

ices

```
+ Ⓜ > Output lab10.public.accounts lab10.public.accounts 2
Database
  postgres 01 21:50:27] 1 row affected in 2 ms
    lab8_i 01 21:50:29] lab10.public> COMMIT
      a [ses 01 21:50:29] completed in 2 ms
      b [ses
```

```
-- Task 4: Isolation Level Demonstration
BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT * FROM products WHERE shop = 'Joe''s Shop';

```



```
SELECT * FROM products WHERE shop = 'Joe''s Shop';
COMMIT;
```

```
-- Task 4: Isolation Level Demonstration      ✓1 ^  
BEGIN;  
DELETE FROM products WHERE shop = 'Joe''s Shop';  
INSERT INTO products (shop, product, price) VALUES ('  
COMMIT;
```

```
-- Task 4: Isolation Level Demonstration
BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT * FROM products WHERE shop = 'Joe''s Shop';

-- Task 4: Isolation Level Demonstration
BEGIN;
DELETE FROM products WHERE shop = 'Joe''s Shop';
INSERT INTO products (shop, product, price) VALUES ('Joe''s Shop', 'Fanta', 3.50);
COMMIT;
```

Services All Services

Tx Output lab10.public.products

	id	shop	product	price
1	3	Joe's Shop	Fanta	3.50

1 row

```
-- Task 4: Isolation Level Demonstration
BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;
SELECT * FROM products WHERE shop = 'Joe''s Shop';

SELECT * FROM products WHERE shop = 'Joe''s Shop';
COMMIT;

-- Task 4: Isolation Level Demonstration
BEGIN;
DELETE FROM products WHERE shop = 'Joe''s Shop';
INSERT INTO products (shop, product, price) VALUES ('Joe''s Shop', 'Fanta', 3.50);
COMMIT;
```

	id	shop	product	price
1	4	Joe's Shop	Fanta	3.50

```
a x lab10_transactions.sql lab9_inclass.sql | v : b x
Tx: Auto v Playground v lab10.public v Tx: Auto v Playground v lab10.public v

1 -- Task 4: Isolation Level Demonstration ✓
2 BEGIN TRANSACTION ISOLATION LEVEL READ COMMITTED;
3 SELECT * FROM products WHERE shop = 'Joe''s Shop';
4
5
6 SELECT * FROM products WHERE shop = 'Joe''s Shop';
7 COMMIT;
8
9
10
11 BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
12 SELECT * FROM products WHERE shop = 'Joe''s Shop';
13
14 ✓ SELECT * FROM products WHERE shop = 'Joe''s Shop';
15 ✓ COMMIT;
16
```

Services All Services a x b x

Tx Output lab10.public.products x

CSV v ↴ ↵ ↺ ↻

ID	Shop	Product	Price
1	Joe's Shop	Fanta	3.50

1 row v :

```
a x lab10_transactions.sql b x lab9_inclass.sql | v : lab10.public v
D | C P S Tx: Auto v | Playground v | Tx: Auto v

4 SELECT * FROM products WHERE shop = 'Joe''s Shop'; ✓
5 COMMIT;
6 BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
7 SELECT * FROM products WHERE shop = 'Joe''s Shop';
8 SELECT * FROM products WHERE shop = 'Joe''s Shop';
9 COMMIT;

10
11 -- Task 5: Phantom Read Demonstration
12 ✓ BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
13 ✓ SELECT MAX(price), MIN(price) FROM products
14 WHERE shop = 'Joe''s Shop';
15 -- W // ▶ 📁 :
16 SELECT MAX(price), MIN(price) FROM products
17 WHERE shop = 'Joe''s Shop';
18 COMMIT;

19
20
21

1 ✓
2 ✓
3 ✓
4 ✓
5 ✓
6
7
8
9
10
11 -- Task 5: Phantom Re
12
13
14
15
```

Services All Services a x b x

Tx Output Result 7-2 x

Tx Output Result 7-2 x

max ↴ min ↴

1	3.5	3.5
---	-----	-----

The screenshot shows two SQL consoles in the DataGrip IDE. Console 'a' contains the following code:

```
5 COMMIT;  
6 BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
7 SELECT * FROM products WHERE shop = 'Joe''s Shop';  
8 SELECT * FROM products WHERE shop = 'Joe''s Shop';  
9 COMMIT;  
10  
11 -- Task 5: Phantom Read Demonstration  
12 BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;  
13 SELECT MAX(price), MIN(price) FROM products  
    WHERE shop = 'Joe''s Shop';  
14  
15 -- Wait for Terminal 2  
16 ✓ SELECT MAX(price), MIN(price) FROM products  
    WHERE shop = 'Joe''s Shop';  
17  
18 ✓ COMMIT;
```

Console 'b' contains the following code:

```
2 BEGIN;  
3 DELETE FROM products WHERE shop = 'Joe''s Shop';  
4 INSERT INTO products (shop, product, price)  
    VALUES ('Joe''s Shop', 'Sprinkles', 1.99);  
5 COMMIT;  
6  
7  
8  
9  
10  
11  
12 ✓  
13 ✓  
14  
15 ✓  
16 -- Task 5: Phantom Read Demonstration  
17 BEGIN;  
18 INSERT INTO products (shop, product, price)  
    VALUES ('Joe''s Shop', 'Sprinkles', 1.99);  
19 COMMIT;
```

```

a x lab10_transactions.sql lab9_inclass.sql Tx: Auto v Playground v lab10.public v
9 COMMIT; ✓
10
11 -- Task 5: Phantom Read Demonstration
12 BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
13 SELECT MAX(price), MIN(price) FROM products
14 WHERE shop = 'Joe''s Shop';
15 -- Wait for Terminal 2
16 SELECT MAX(price), MIN(price) FROM products
17 WHERE shop = 'Joe''s Shop';
18 COMMIT;
19
20 -- Task 6: Dirty Read Demonstration
21 ✓ BEGIN TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
22 ✓ SELECT * FROM products WHERE shop = 'Joe''s Shop';
23 - [Lightbulb] // ⚙️ ⏪ ⏴ ⏵ ⏹ : to UPDATE but NOT commit
24 SELECT * FROM products WHERE shop = 'Joe''s Shop';
25 -- Wait for Terminal 2 to ROLLBACK
26 SELECT * FROM products WHERE shop = 'Joe''s Shop';
27 COMMIT;
28
29
30

b x COMMIT; ✓ 2 ⌂
6
7
8
9
10
11 -- Task 5: Phantom Read Demonstration
12 ✓ BEGIN;
13 ✓ INSERT INTO products (shop, product, price)
14 VALUES ('Joe''s Shop', 'Sprite', 4.00);
15 ✓ COMMIT;
16
17
18
19
20 -- Task 6: Dirty Read Demonstration
21 BEGIN;
22 UPDATE products SET price = 99.99
23 WHERE product = 'Fanta';
24 -- Wait here (don't commit yet)
25 -- Then:
26 ROLLBACK;

Services All Services a x b x
Tx Output lab10.public.products x
[Table icon] id shop product price
1 5 Joe's Shop Fanta 3.50
2 6 Joe's Shop Sprite 4.00
2 rows v ...

```

Terminal a:

```

9 COMMIT;
10
11 -- Task 5: Phantom Read Demonstration
12 BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
13 SELECT MAX(price), MIN(price) FROM products
14   WHERE shop = 'Joe''s Shop';
15 -- Wait for Terminal 2
16 SELECT MAX(price), MIN(price) FROM products
17   WHERE shop = 'Joe''s Shop';
18 COMMIT;
19
20 -- Task 6: Dirty Read Demonstration
21 BEGIN TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
22 SELECT * FROM products WHERE shop = 'Joe''s Shop';
23 -- Wait for Terminal 2 to UPDATE but NOT commit
24 SELECT * FROM products WHERE shop = 'Joe''s Shop';
25 // ... to ROLLBACK
26 SELECT * FROM products WHERE shop = 'Joe''s Shop';
27 COMMIT;

```

Terminal b:

```

6
7
8
9
10
11 -- Task 5: Phantom Read Demonstration
12 BEGIN;
13 INSERT INTO products (shop, product, price)
14   VALUES ('Joe''s Shop', 'Sprite', 4.00);
15 COMMIT;
16
17
18
19
20 -- Task 6: Dirty Read Demonstration
21 BEGIN;
22 UPDATE products SET price = 99.99
23   WHERE product = 'Fanta!';
24 -- Wait here (don't commit yet)
25 -- Then:
26 ROLLBACK;

```

Services: All Services

Tx: Output lab10.public.products

id	shop	product	price
1	Joe's Shop	Fanta	3.50
2	Joe's Shop	Sprite	4.00

Two terminal windows (a and b) showing SQL scripts for demonstrating transaction isolation levels.

Terminal a:

```

9 COMMIT;
10
11 -- Task 5: Phantom Read Demonstration
12 BEGIN TRANSACTION ISOLATION LEVEL REPEATABLE READ;
13 SELECT MAX(price), MIN(price) FROM products
14 WHERE shop = 'Joe''s Shop';
15 -- Wait for Terminal 2
16 SELECT MAX(price), MIN(price) FROM products
17 WHERE shop = 'Joe''s Shop';
18 COMMIT;
19
20 -- Task 6: Dirty Read Demonstration
21 BEGIN TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
22 SELECT * FROM products WHERE shop = 'Joe''s Shop';
23 -- Wait for Terminal 2 to UPDATE but NOT commit
24 UPDATE products SET price = 99.99
25 WHERE product = 'Fanta';
26 -- Wait for Terminal 2 to ROLLBACK
27 SELECT * FROM products WHERE shop = 'Joe''s Shop';
28 COMMIT;
29
30

```

Terminal b:

```

6
7
8
9
10
11 -- Task 5: Phantom Read Demonstration
12 BEGIN;
13 INSERT INTO products (shop, product, price)
14 VALUES ('Joe''s Shop', 'Sprite', 2.00);
15 COMMIT;
16
17
18
19
20 -- Task 6: Dirty Read Demonstration
21 BEGIN;
22 UPDATE products SET price = 99.99
23 WHERE product = 'Fanta';
24 -- Wait here (don't commit yet)
25 -- Then:
26 ROLLBACK;
27
28
29
30

```

Output Tab:

id	shop	product	price
1	Joe's Shop	Fanta	3.50
2	Joe's Shop	Sprite	4.00

2 rows



```
70  
71 -- Exercise 2  
72 ✓ BEGIN;  
73 ✓ INSERT INTO products (shop, product, price) VALUES ( shop 'TestShop', product 'TempProduct', price 10.00);  
74 ✓ SAVEPOINT sp1;  
75 ✓ UPDATE products SET price = 12.00 WHERE shop='TestShop' AND product='TempProduct';  
76 ✓ SAVEPOINT sp2;  
77 ✓ DELETE FROM products WHERE shop='TestShop' AND product='TempProduct';  
78  
79 ✓ ROLLBACK TO sp1;  
80 ✓ COMMIT;  
81  
82 ✓ SELECT * FROM products WHERE shop='TestShop';
```

Services	All Services	a	b	
Tx	+ ↻ ⌂ >	Output	lab10.public.accounts	lab10.public.products
lab8_inc	‑			
lab9_inc	‑			
lab8_inc	‑			
b 70 ms	1	7	TestShop	TempProduct
				10.00
lab10_tr				