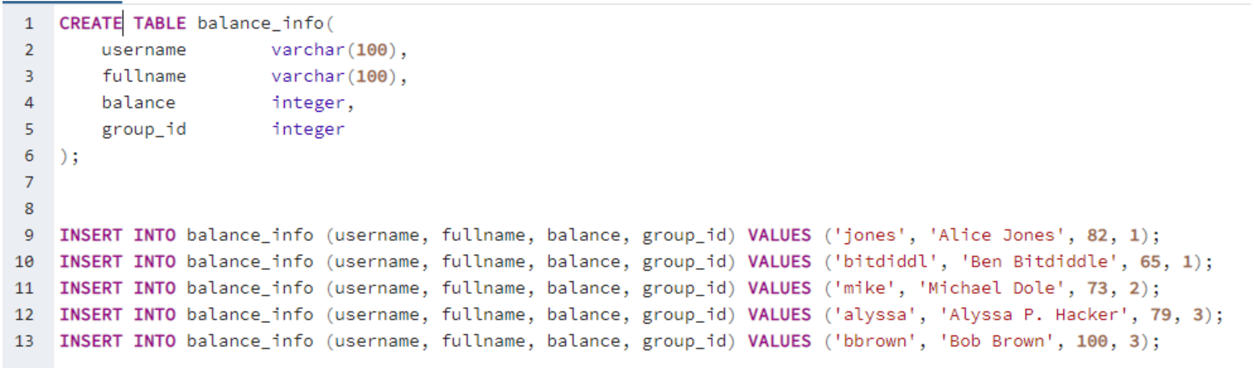
# Exercise 2

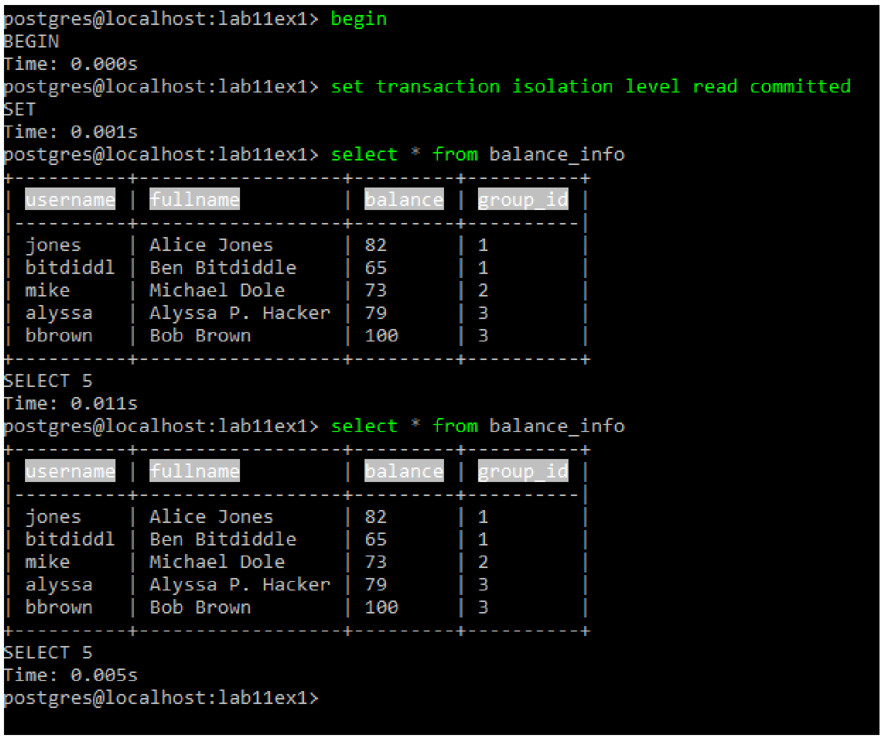
## Create and fill table



```
CREATE TABLE balance_info(
    username        varchar(100),
    fullname        varchar(100),
    balance         integer,
    group_id        integer
);


INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('jones', 'Alice Jones', 82, 1);
INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('bitdiddl', 'Ben Bitdiddle', 65, 1);
INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('mike', 'Michael Dole', 73, 2);
INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('alyssa', 'Alyssa P. Hacker', 79, 3);
INSERT INTO balance_info (username, fullname, balance, group_id) VALUES ('bbrown', 'Bob Brown', 100, 3);
```

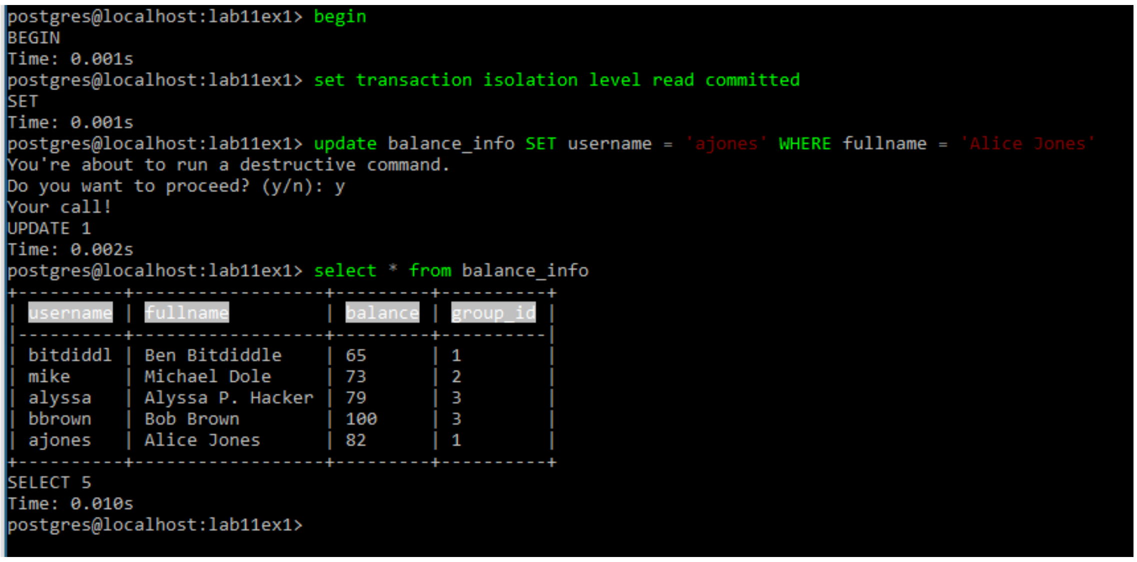
## First part with **read committed**:

1)



```
postgres@localhost:lab11ex1> begin
BEGIN
Time: 0.000s
postgres@localhost:lab11ex1> set transaction isolation level read committed
SET
Time: 0.001s
postgres@localhost:lab11ex1> select * from balance_info
+----------+------------------+---------+----------+
| username | fullname         | balance | group_id |
|----------+------------------+---------+----------|
| jones    | Alice Jones      | 82      | 1        |
| bitdiddl | Ben Bitdiddle    | 65      | 1        |
| mike     | Michael Dole     | 73      | 2        |
| alyssa   | Alyssa P. Hacker | 79      | 3        |
| bbrown   | Bob Brown        | 100     | 3        |
+----------+------------------+---------+----------+
SELECT 5
Time: 0.011s
postgres@localhost:lab11ex1> select * from balance_info
+----------+------------------+---------+----------+
| username | fullname         | balance | group_id |
|----------+------------------+---------+----------|
| jones    | Alice Jones      | 82      | 1        |
| bitdiddl | Ben Bitdiddle    | 65      | 1        |
| mike     | Michael Dole     | 73      | 2        |
| alyssa   | Alyssa P. Hacker | 79      | 3        |
| bbrown   | Bob Brown        | 100     | 3        |
+----------+------------------+---------+----------+
SELECT 5
Time: 0.005s
postgres@localhost:lab11ex1>
```

2)



```
postgres@localhost:lab11ex1> begin
BEGIN
Time: 0.001s
postgres@localhost:lab11ex1> set transaction isolation level read committed
SET
Time: 0.001s
postgres@localhost:lab11ex1> update balance_info SET username = 'ajones' WHERE fullname = 'Alice Jones'
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
UPDATE 1
Time: 0.002s
postgres@localhost:lab11ex1> select * from balance_info
+----------+------------------+---------+----------+
| username | fullname         | balance | group_id |
|----------+------------------+---------+----------|
| bitdiddl | Ben Bitdiddle    | 65      | 1        |
| mike     | Michael Dole     | 73      | 2        |
| alyssa   | Alyssa P. Hacker | 79      | 3        |
| bbrown   | Bob Brown        | 100     | 3        |
| ajones   | Alice Jones      | 82      | 1        |
+----------+------------------+---------+----------+
SELECT 5
Time: 0.010s
postgres@localhost:lab11ex1>
```

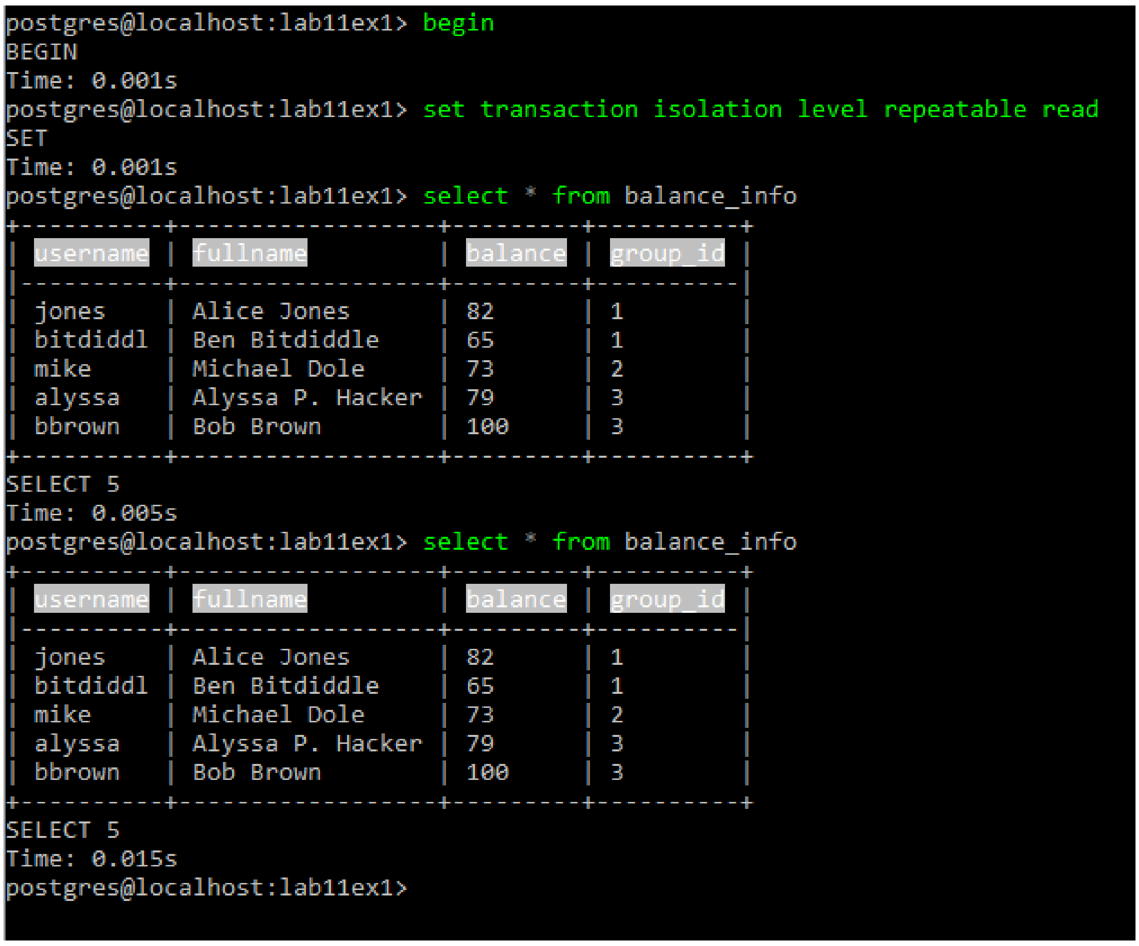
After 4 steps, we can see that the tables are different because the committed read isolation level is free from dirty reads, which results in reading uncommitted changes from other transactions.

After 5 step the tables became the same because the committed read isolation level isn’t free from non-repeatable read. A non-repeatable read means that one of the rows you requested at different stages of the transaction cannot be updated by other transactions.

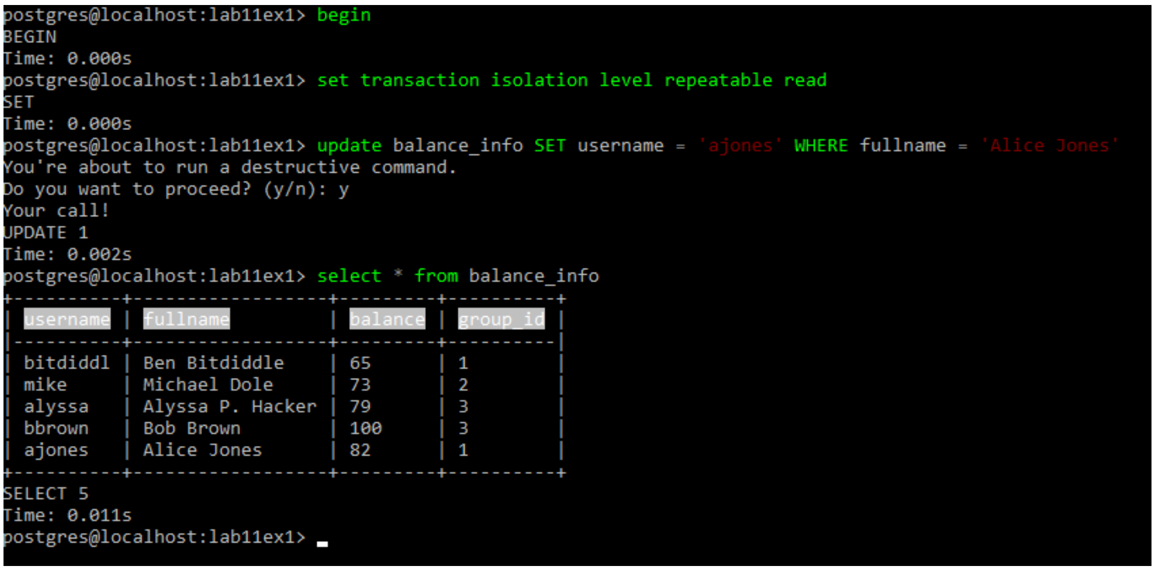
After step 8, Alice's account balance increases by 10 because the committed read isolation level is not free of lost updates. A lost update is when the first transaction reads data into its local memory and then the second transaction changes that data and commits its change. The first transaction then updates the same data based on what was read into memory prior to the second transaction. In this case, the update performed by the second transaction can be considered a lost update.

Into my computer in the second terminal process go to the infinite loop.

First part with **repeatable read**:

1) 

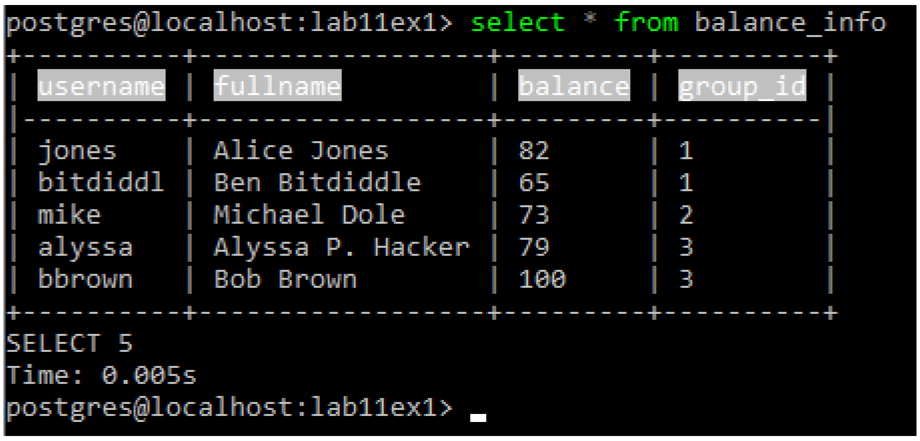
```
postgres@localhost:lab11ex1> begin
BEGIN
Time: 0.001s
postgres@localhost:lab11ex1> set transaction isolation level repeatable read
SET
Time: 0.001s
postgres@localhost:lab11ex1> select * from balance_info
+----------+-----------------+---------+----------+
| username | fullname        | balance | group_id |
|----------+-----------------+---------+----------|
| jones    | Alice Jones     | 82      | 1        |
| bitdiddl | Ben Bitdiddle   | 65      | 1        |
| mike     | Michael Dole    | 73      | 2        |
| alyssa   | Alyssa P. Hacker| 79      | 3        |
| bbrown   | Bob Brown       | 100     | 3        |
+----------+-----------------+---------+----------+
SELECT 5
Time: 0.005s
postgres@localhost:lab11ex1> select * from balance_info
+----------+-----------------+---------+----------+
| username | fullname        | balance | group_id |
|----------+-----------------+---------+----------|
| jones    | Alice Jones     | 82      | 1        |
| bitdiddl | Ben Bitdiddle   | 65      | 1        |
| mike     | Michael Dole    | 73      | 2        |
| alyssa   | Alyssa P. Hacker| 79      | 3        |
| bbrown   | Bob Brown       | 100     | 3        |
+----------+-----------------+---------+----------+
SELECT 5
Time: 0.015s
postgres@localhost:lab11ex1>
```

2) 

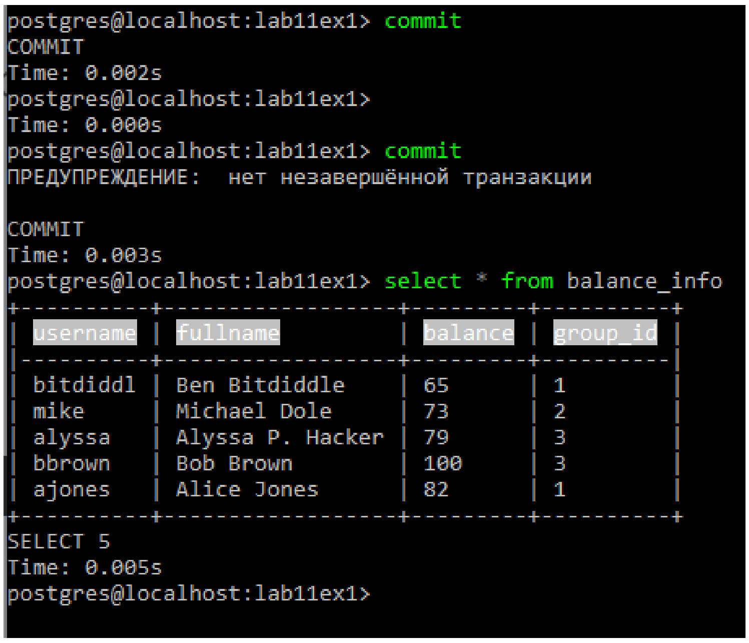
```
postgres@localhost:lab11ex1> begin
BEGIN
Time: 0.000s
postgres@localhost:lab11ex1> set transaction isolation level repeatable read
SET
Time: 0.000s
postgres@localhost:lab11ex1> update balance_info SET username = 'ajones' WHERE fullname = 'Alice Jones'
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
UPDATE 1
Time: 0.002s
postgres@localhost:lab11ex1> select * from balance_info
+----------+-----------------+---------+----------+
| username | fullname        | balance | group_id |
|----------+-----------------+---------+----------|
| bitdiddl | Ben Bitdiddle   | 65      | 1        |
| mike     | Michael Dole    | 73      | 2        |
| alyssa   | Alyssa P. Hacker| 79      | 3        |
| bbrown   | Bob Brown       | 100     | 3        |
| ajones   | Alice Jones     | 82      | 1        |
+----------+-----------------+---------+----------+
SELECT 5
Time: 0.011s
postgres@localhost:lab11ex1> _
```

After 4 steps, we can see that the tables are different because the repeatable read isolation level is free from dirty reads, which results in reading uncommitted changes from other transactions.

After step 5

1) 

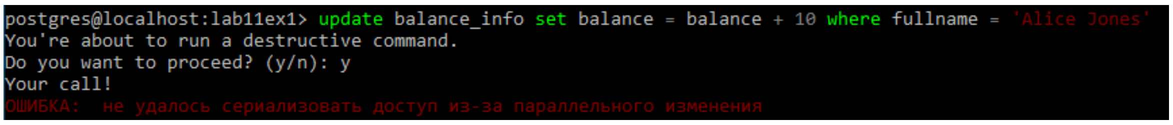
```
postgres@localhost:lab11ex1> select * from balance_info
+----------+------------------+---------+----------+
| username | fullname         | balance | group_id |
|----------+------------------+---------+----------|
| jones    | Alice Jones      | 82      | 1        |
| bitdiddl | Ben Bitdiddle    | 65      | 1        |
| mike     | Michael Dole     | 73      | 2        |
| alyssa   | Alyssa P. Hacker | 79      | 3        |
| bbrown   | Bob Brown        | 100     | 3        |
+----------+------------------+---------+----------+
SELECT 5
Time: 0.005s
postgres@localhost:lab11ex1>
```

2) 

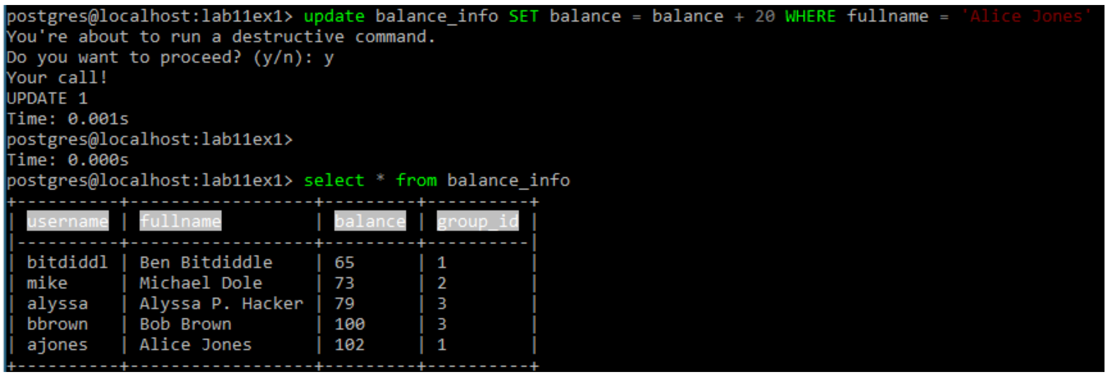
```
postgres@localhost:lab11ex1> commit
COMMIT
Time: 0.002s
postgres@localhost:lab11ex1>
Time: 0.000s
postgres@localhost:lab11ex1> commit
ПРЕДУПРЕЖДЕНИЕ:  нет незавершённой транзакции

COMMIT
Time: 0.003s
postgres@localhost:lab11ex1> select * from balance_info
+----------+------------------+---------+----------+
| username | fullname         | balance | group_id |
|----------+------------------+---------+----------|
| bitdiddl | Ben Bitdiddle    | 65      | 1        |
| mike     | Michael Dole     | 73      | 2        |
| alyssa   | Alyssa P. Hacker | 79      | 3        |
| bbrown   | Bob Brown        | 100     | 3        |
| ajones   | Alice Jones      | 82      | 1        |
+----------+------------------+---------+----------+
SELECT 5
Time: 0.005s
postgres@localhost:lab11ex1>
```

the tables remain distinct because the committed read isolation level is free from non-repeatable reads. This means that one of the rows requested by you at different stages of the transaction may be updated by other transactions.

After step 8,

1) 

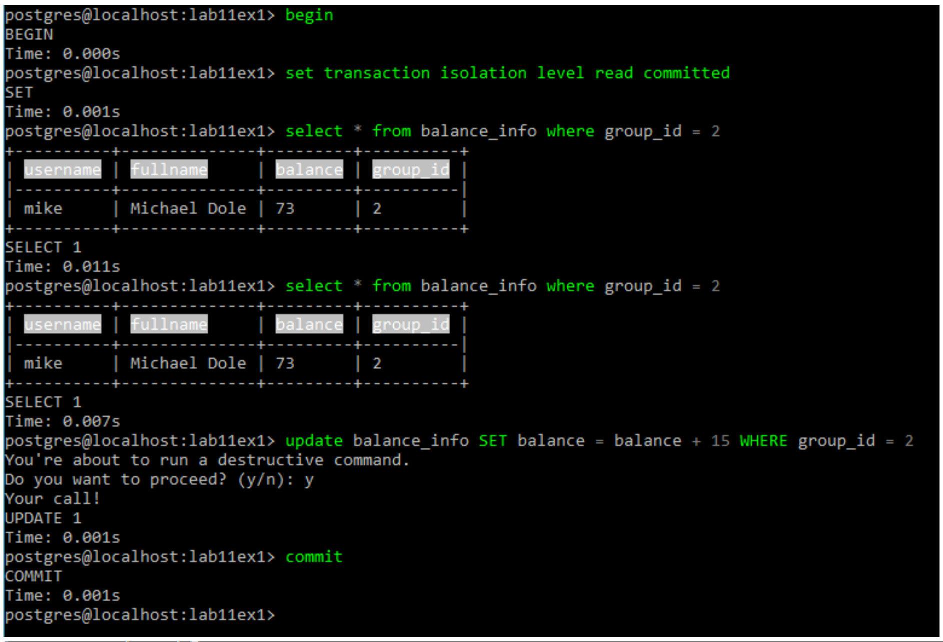
```
postgres@localhost:lab11ex1> update balance_info set balance = balance + 10 where fullname = 'Alice Jones'
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
ОШИБКА:  не удалось сериализовать доступ из-за параллельного изменения
```

2) 

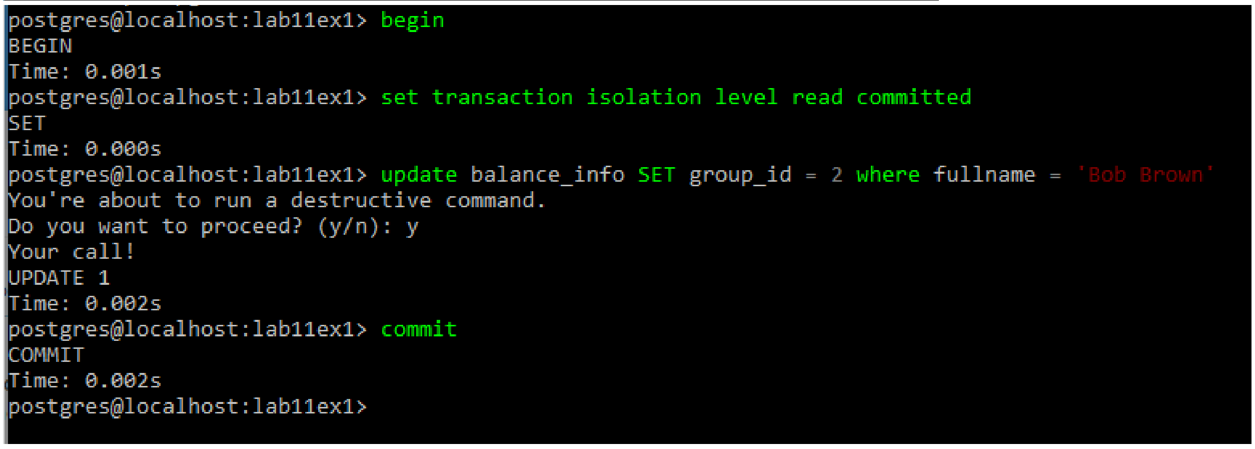
```
postgres@localhost:lab11ex1> update balance_info SET balance = balance + 20 WHERE fullname = 'Alice Jones'
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
UPDATE 1
Time: 0.001s
postgres@localhost:lab11ex1>
Time: 0.000s
postgres@localhost:lab11ex1> select * from balance_info
+----------+------------------+---------+----------+
| username | fullname         | balance | group_id |
|----------+------------------+---------+----------|
| bitdiddl | Ben Bitdiddle    | 65      | 1        |
| mike     | Michael Dole     | 73      | 2        |
| alyssa   | Alyssa P. Hacker | 79      | 3        |
| bbrown   | Bob Brown        | 100     | 3        |
| ajones   | Alice Jones      | 102     | 1        |
+----------+------------------+---------+----------+
```

In the second terminal Alice's account balance increases by 20 and the first terminal get the error because the committed read isolation level is free of lost updates.

Second part with **read committed**:

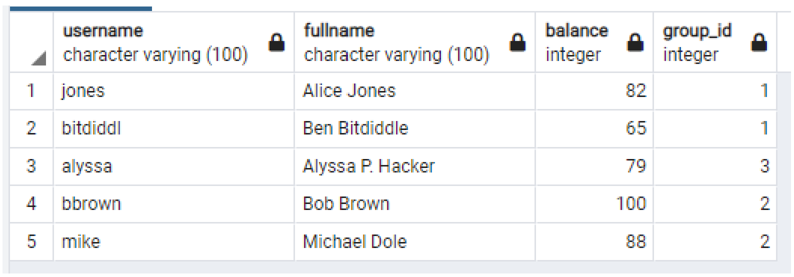
1) 

```
postgres@localhost:lab11ex1> begin
BEGIN
Time: 0.000s
postgres@localhost:lab11ex1> set transaction isolation level read committed
SET
Time: 0.001s
postgres@localhost:lab11ex1> select * from balance_info where group_id = 2
+----------+--------------+---------+----------+
| username | fullname     | balance | group_id |
|----------+--------------+---------+----------|
| mike     | Michael Dole | 73      | 2        |
+----------+--------------+---------+----------+
SELECT 1
Time: 0.011s
postgres@localhost:lab11ex1> select * from balance_info where group_id = 2
+----------+--------------+---------+----------+
| username | fullname     | balance | group_id |
|----------+--------------+---------+----------|
| mike     | Michael Dole | 73      | 2        |
+----------+--------------+---------+----------+
SELECT 1
Time: 0.007s
postgres@localhost:lab11ex1> update balance_info SET balance = balance + 15 WHERE group_id = 2
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
UPDATE 1
Time: 0.001s
postgres@localhost:lab11ex1> commit
COMMIT
Time: 0.001s
postgres@localhost:lab11ex1>
```

2) 

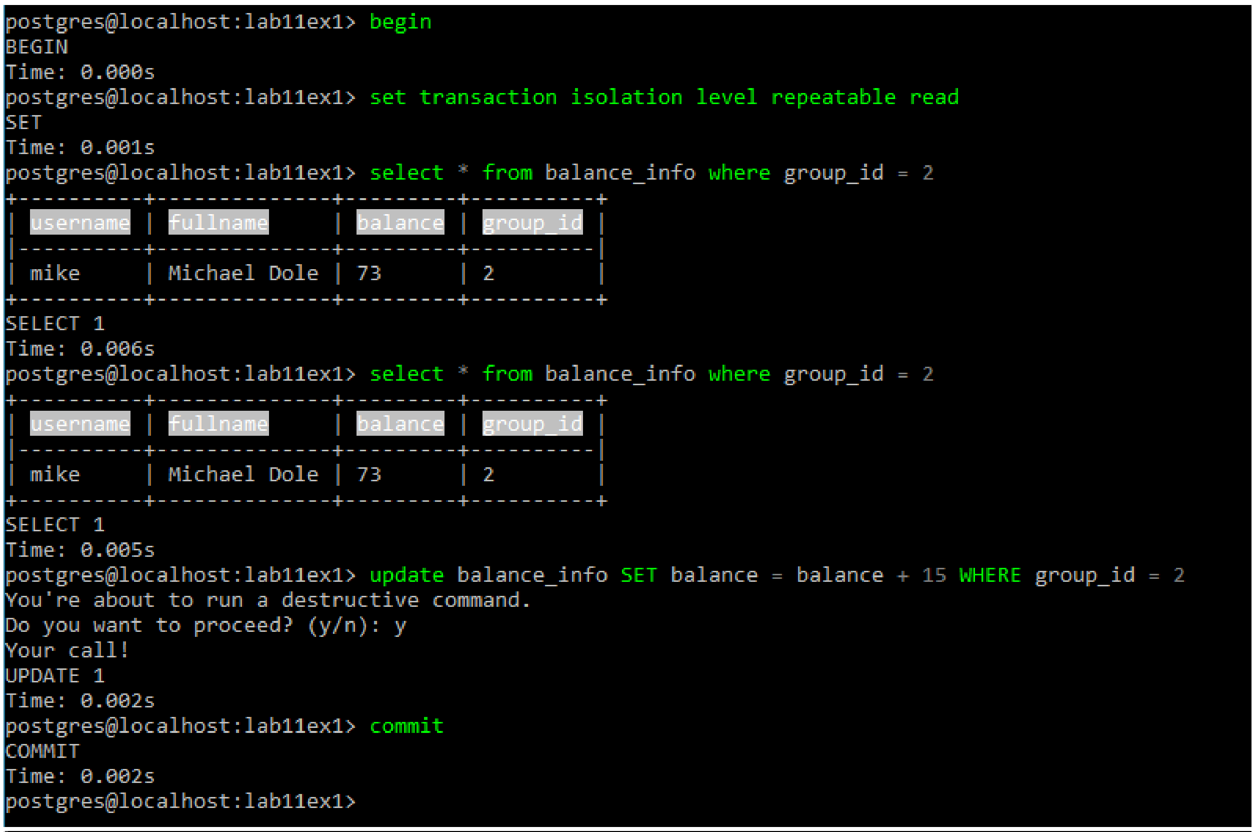
```
postgres@localhost:lab11ex1> begin
BEGIN
Time: 0.001s
postgres@localhost:lab11ex1> set transaction isolation level read committed
SET
Time: 0.000s
postgres@localhost:lab11ex1> update balance_info SET group_id = 2 where fullname = 'Bob Brown'
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
UPDATE 1
Time: 0.002s
postgres@localhost:lab11ex1> commit
COMMIT
Time: 0.002s
postgres@localhost:lab11ex1>
```

The list of accounts with the group_id = 2 in T1 was not updated after in T2 the Bob group_id was changed by 2 because the committed read isolation level is free from dirty reads, which results in reading uncommitted changes from other transactions. Since the balance was increases by 15 only for Michael Dole.

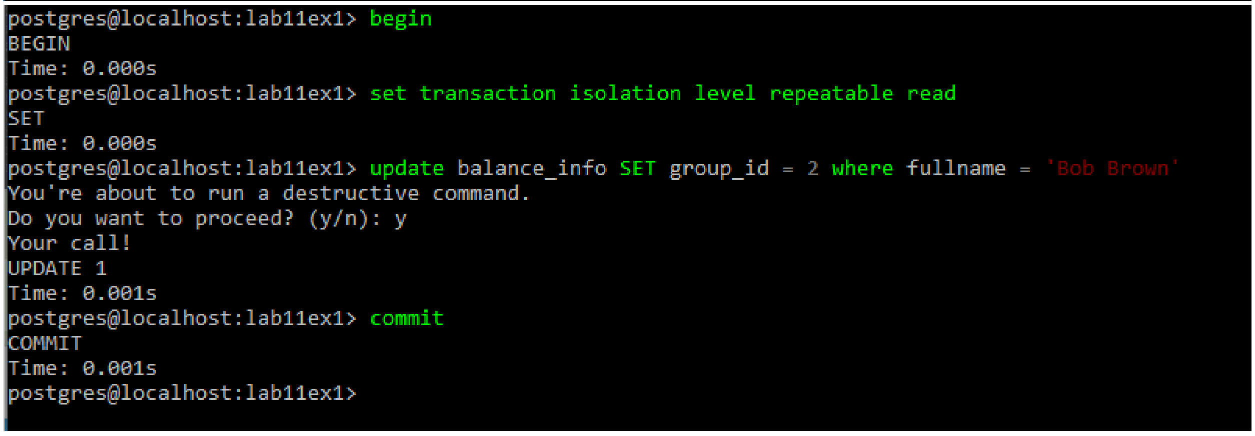


| | username character varying (100) | fullname character varying (100) | balance integer | group_id integer |
|---|---|---|---|---|
| 1 | jones | Alice Jones | 82 | 1 |
| 2 | bitdiddl | Ben Bitdiddle | 65 | 1 |
| 3 | alyssa | Alyssa P. Hacker | 79 | 3 |
| 4 | bbrown | Bob Brown | 100 | 2 |
| 5 | mike | Michael Dole | 88 | 2 |

Second part with **repeatable read**:

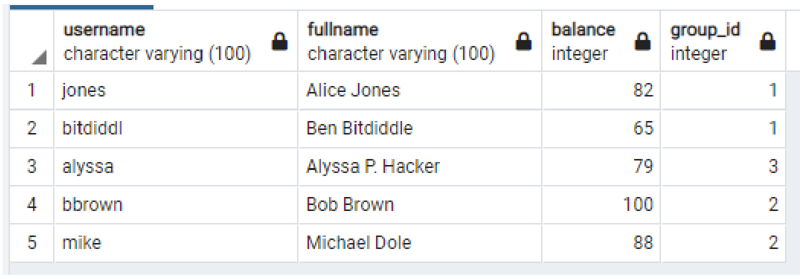
1)


```
postgres@localhost:lab11ex1> begin
BEGIN
Time: 0.000s
postgres@localhost:lab11ex1> set transaction isolation level repeatable read
SET
Time: 0.001s
postgres@localhost:lab11ex1> select * from balance_info where group_id = 2
+----------+--------------+---------+----------+
| username | fullname     | balance | group_id |
|----------+--------------+---------+----------|
| mike     | Michael Dole | 73      | 2        |
+----------+--------------+---------+----------+
SELECT 1
Time: 0.006s
postgres@localhost:lab11ex1> select * from balance_info where group_id = 2
+----------+--------------+---------+----------+
| username | fullname     | balance | group_id |
|----------+--------------+---------+----------|
| mike     | Michael Dole | 73      | 2        |
+----------+--------------+---------+----------+
SELECT 1
Time: 0.005s
postgres@localhost:lab11ex1> update balance_info SET balance = balance + 15 WHERE group_id = 2
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
UPDATE 1
Time: 0.002s
postgres@localhost:lab11ex1> commit
COMMIT
Time: 0.002s
postgres@localhost:lab11ex1>
```

2)


```
postgres@localhost:lab11ex1> begin
BEGIN
Time: 0.000s
postgres@localhost:lab11ex1> set transaction isolation level repeatable read
SET
Time: 0.000s
postgres@localhost:lab11ex1> update balance_info SET group_id = 2 where fullname = 'Bob Brown'
You're about to run a destructive command.
Do you want to proceed? (y/n): y
Your call!
UPDATE 1
Time: 0.001s
postgres@localhost:lab11ex1> commit
COMMIT
Time: 0.001s
postgres@localhost:lab11ex1>
```

The results the same with the read committed isolation level since both isolation levels is free from dirty reads.



| | username<br>character varying (100) | fullname<br>character varying (100) | balance<br>integer | group_id<br>integer |
|---|---|---|---|---|
| 1 | jones | Alice Jones | 82 | 1 |
| 2 | bitdiddl | Ben Bitdiddle | 65 | 1 |
| 3 | alyssa | Alyssa P. Hacker | 79 | 3 |
| 4 | bbrown | Bob Brown | 100 | 2 |
| 5 | mike | Michael Dole | 88 | 2 |