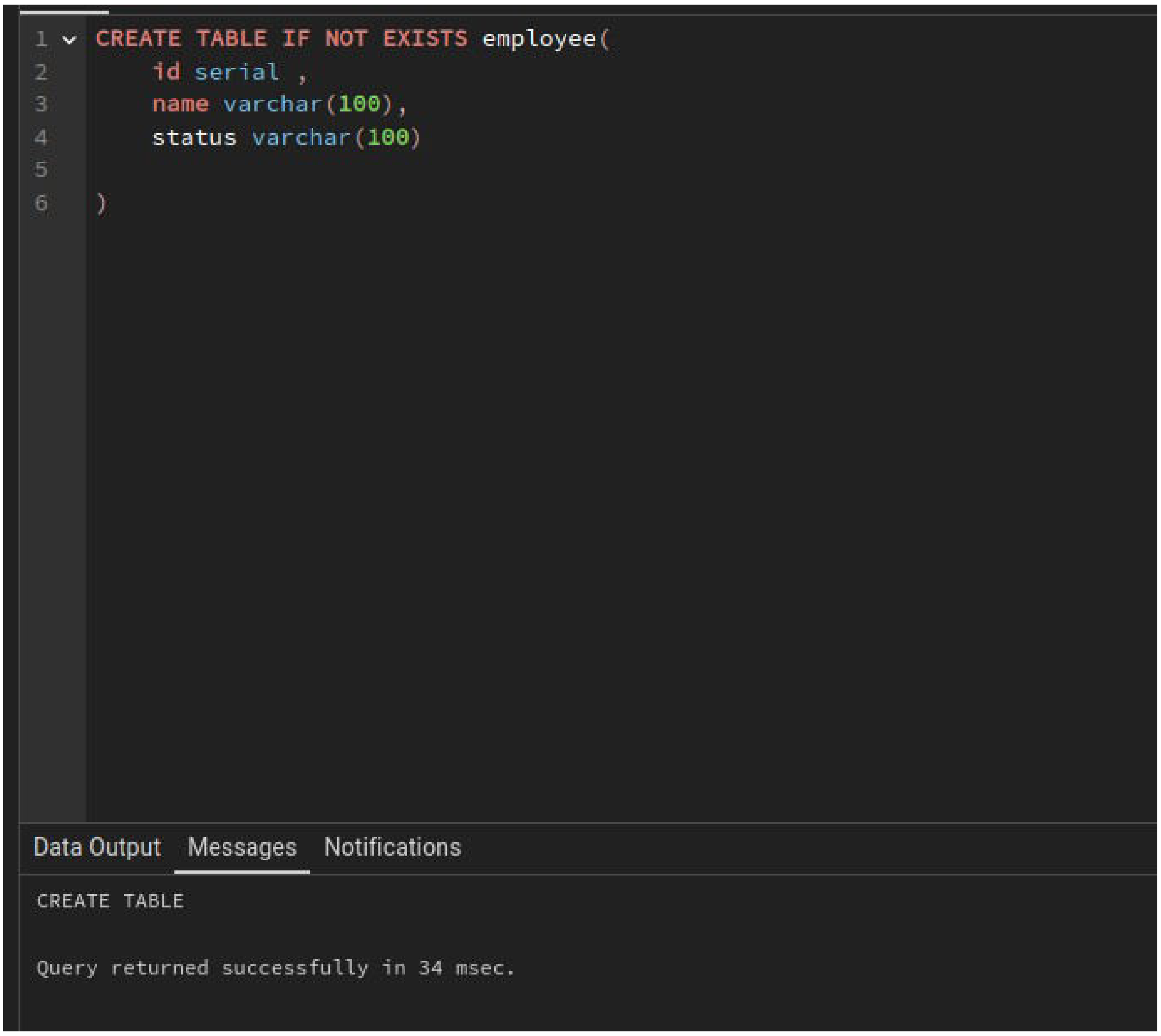
PG_DWH TASK 2

CONTENTS

1.

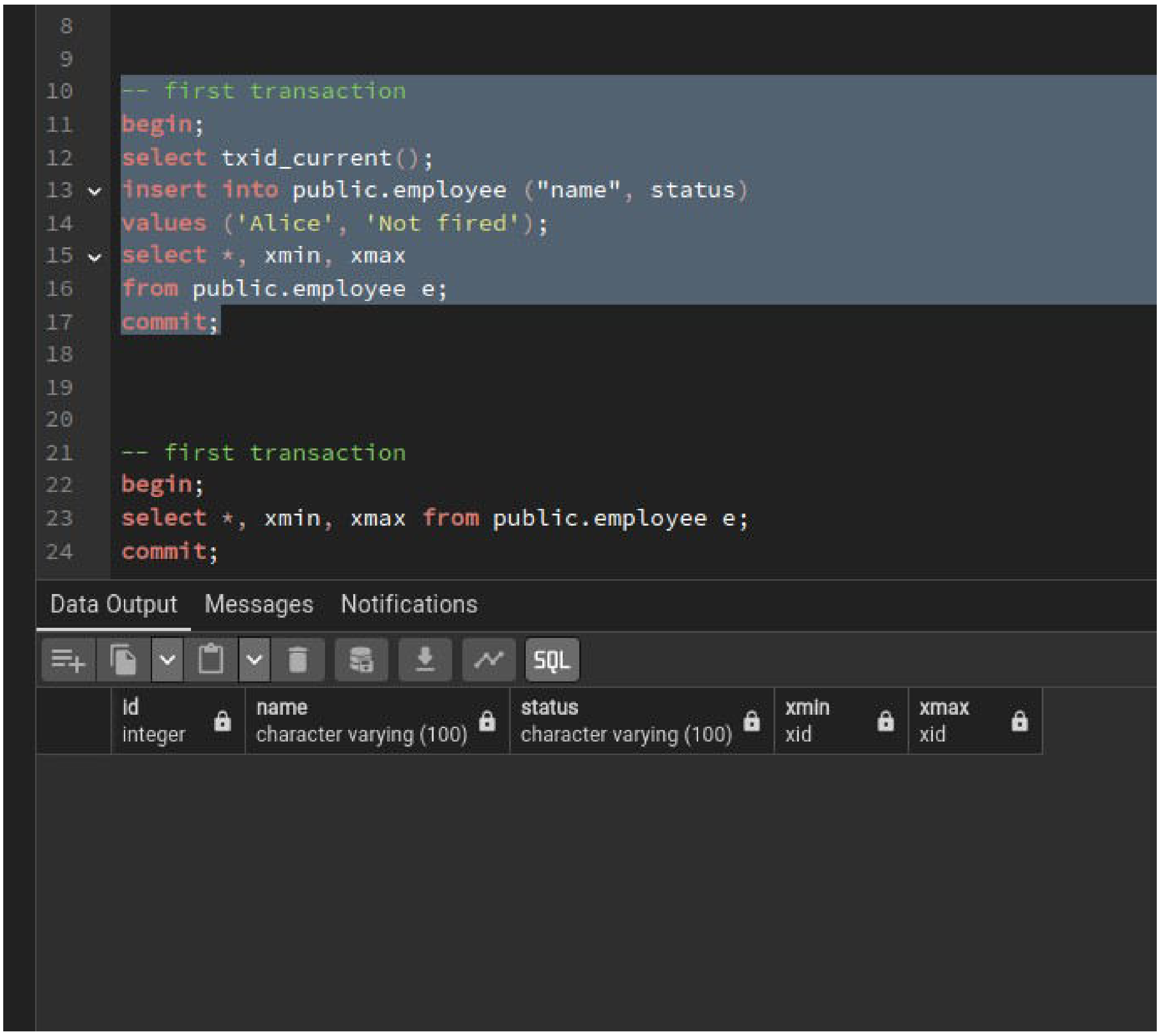


```sql
CREATE TABLE IF NOT EXISTS employee(
    id serial ,
    name varchar(100),
    status varchar(100)

)
```

```
CREATE TABLE

Query returned successfully in 34 msec.
```

Table created.

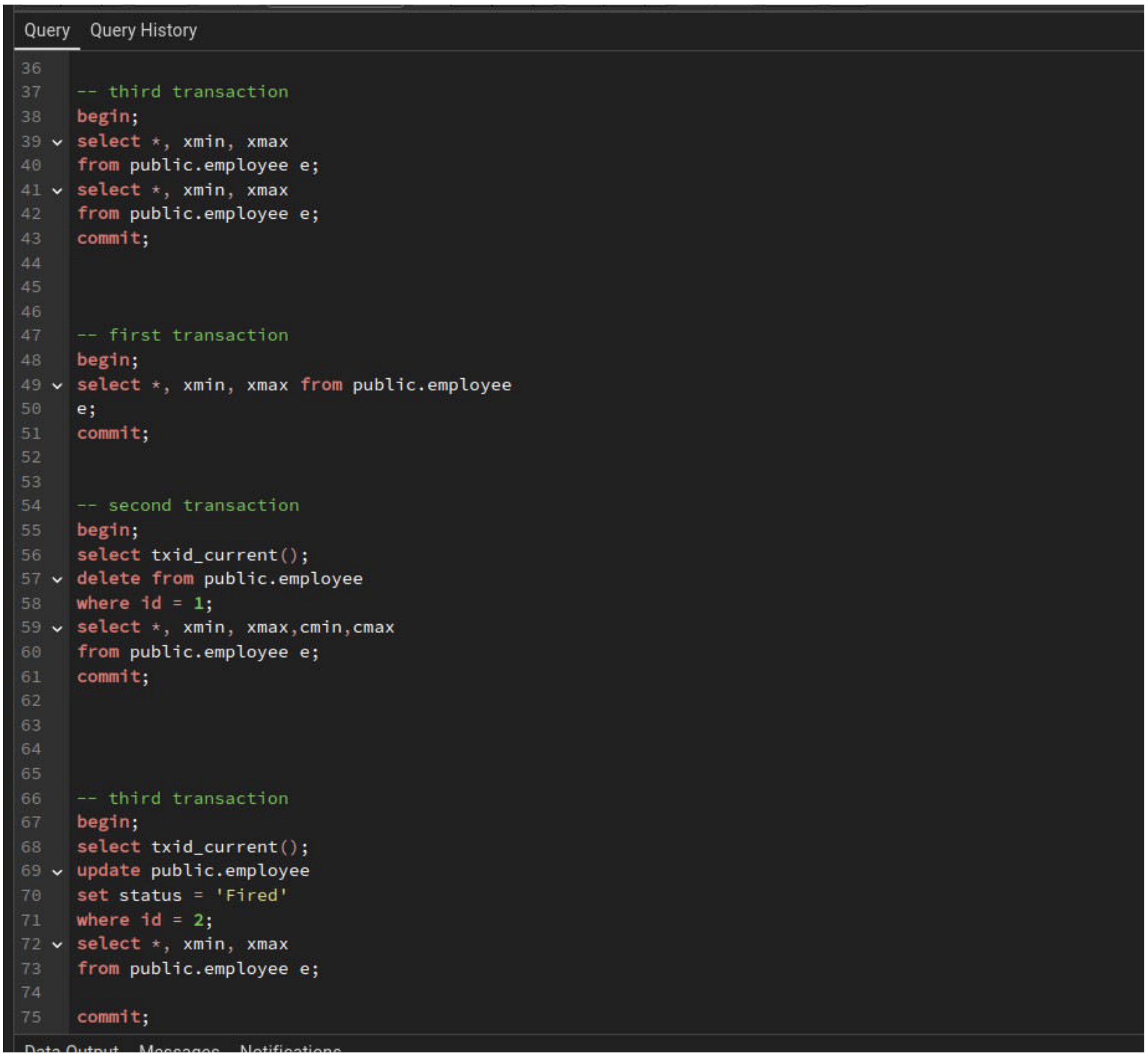
2.



```
-- first transaction
begin;
select txid_current();
insert into public.employee ("name", status)
values ('Alice', 'Not fired');
select *, xmin, xmax
from public.employee e;
commit;



-- first transaction
begin;
select *, xmin, xmax from public.employee e;
commit;
```

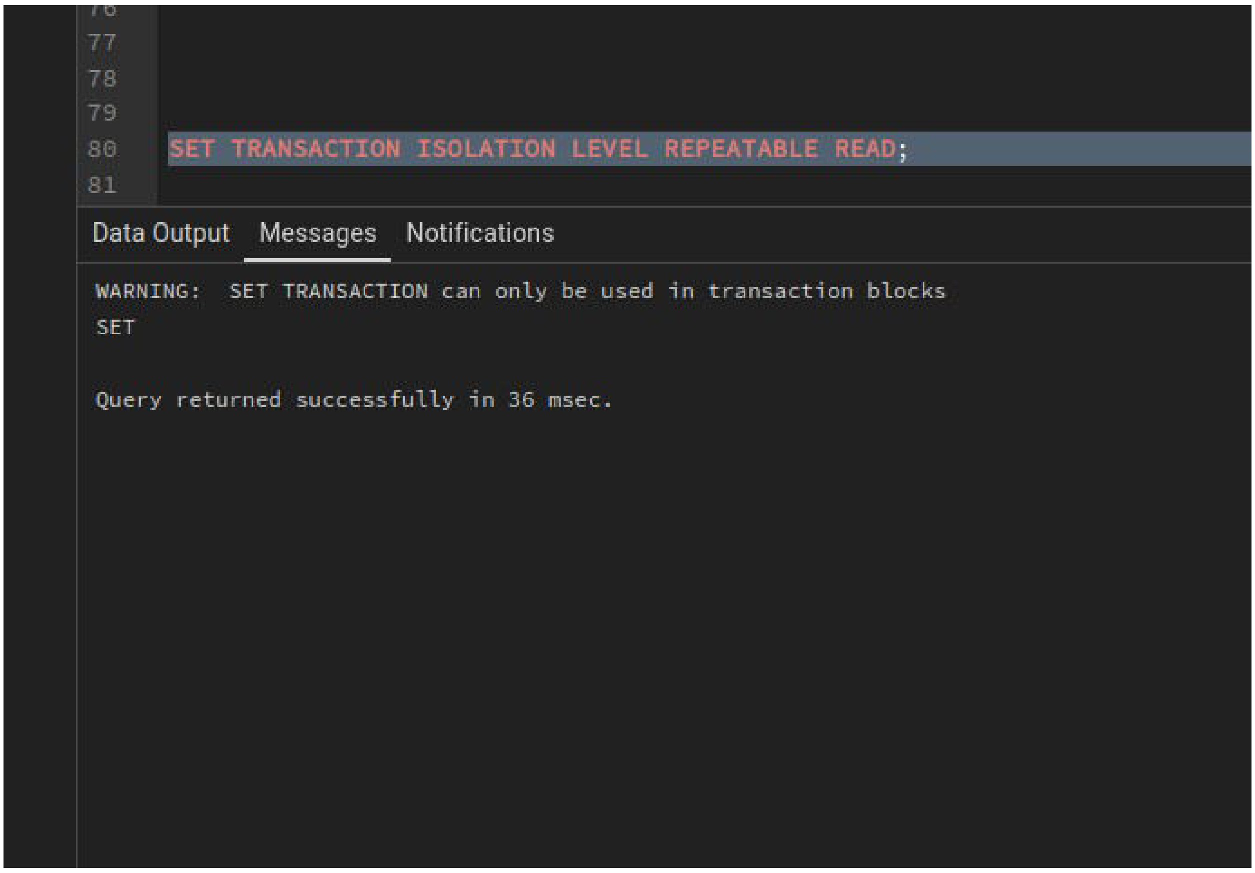
id integer	name character varying (100)	status character varying (100)	xmin xid	xmax xid

All queries are executed as in lecture .



```
-- third transaction
begin;
select *, xmin, xmax
from public.employee e;
select *, xmin, xmax
from public.employee e;
commit;



-- first transaction
begin;
select *, xmin, xmax from public.employee
e;
commit;


-- second transaction
begin;
select txid_current();
delete from public.employee
where id = 1;
select *, xmin, xmax,cmin,cmax
from public.employee e;
commit;




-- third transaction
begin;
select txid_current();
update public.employee
set status = 'Fired'
where id = 2;
select *, xmin, xmax
from public.employee e;

commit;
```

3.

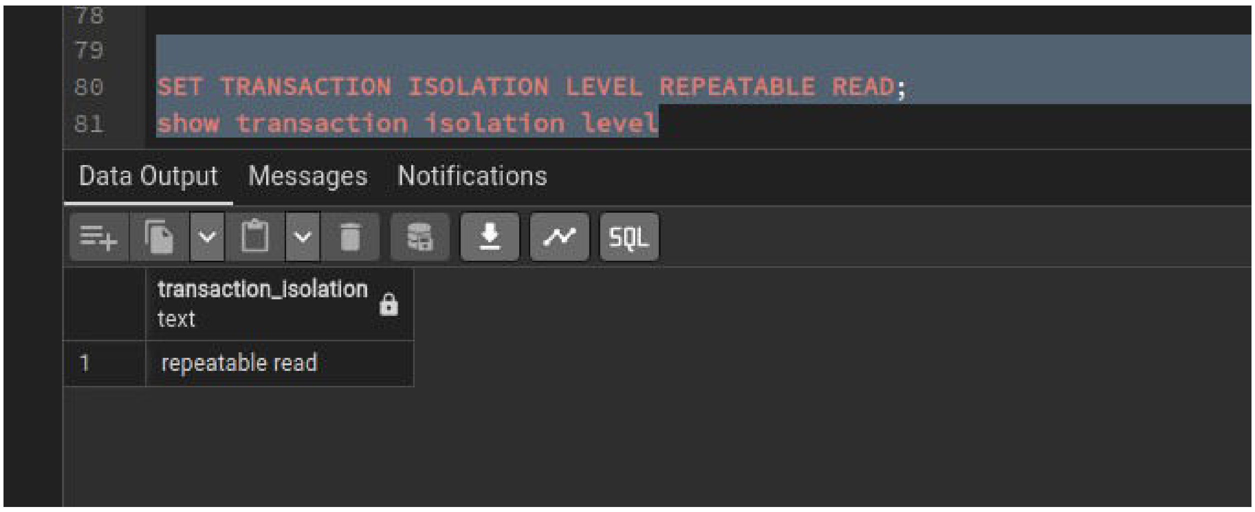


```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
```

```
WARNING:  SET TRANSACTION can only be used in transaction blocks
SET

Query returned successfully in 36 msec.
```

4.

Set Transaction isolation level to repeatable read.

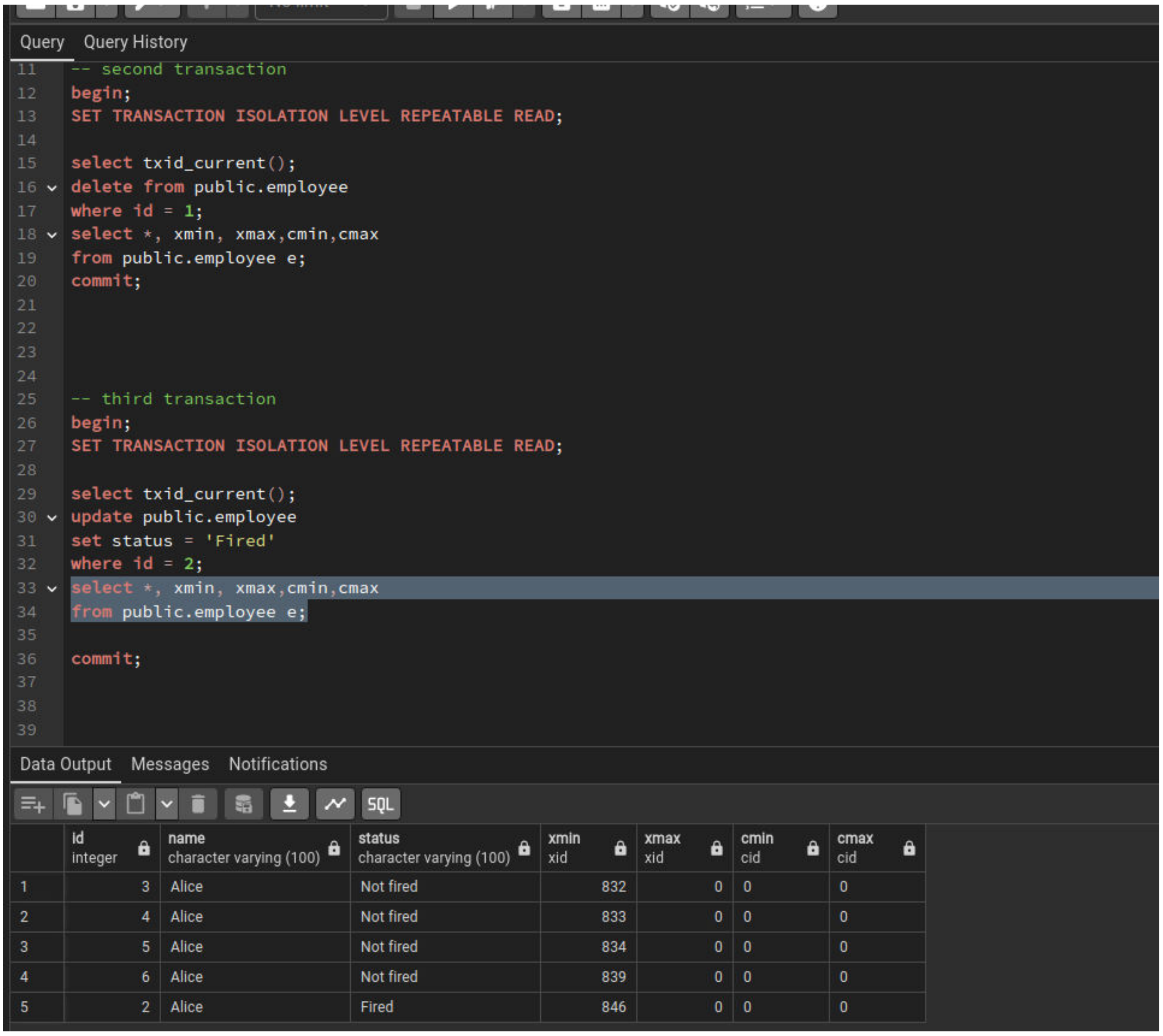


```
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
show transaction isolation level
```

	transaction_isolation text
1	repeatable read

Transaction isolation level is repeatable read.

5.



Query Query History

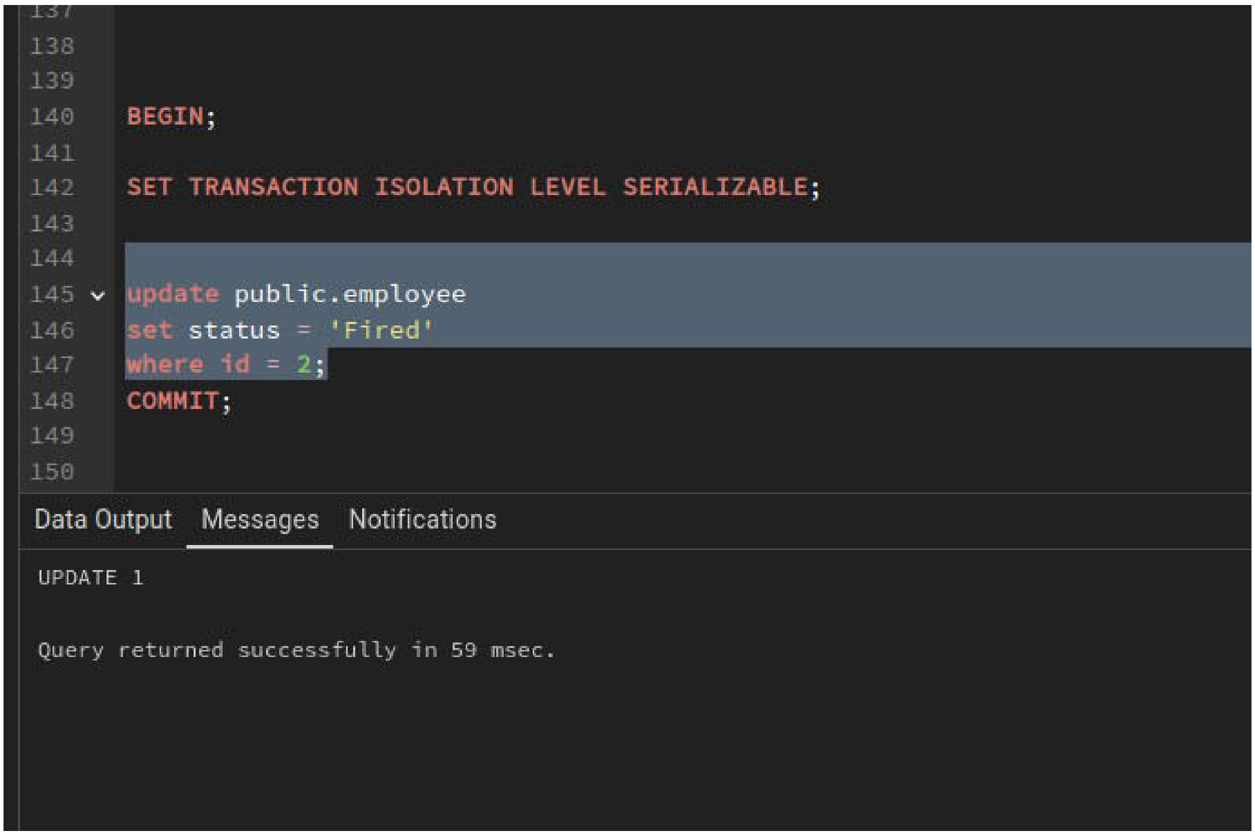
```
-- second transaction
begin;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

select txid_current();
delete from public.employee
where id = 1;
select *, xmin, xmax,cmin,cmax
from public.employee e;
commit;




-- third transaction
begin;
SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

select txid_current();
update public.employee
set status = 'Fired'
where id = 2;
select *, xmin, xmax,cmin,cmax
from public.employee e;

commit;
```

Data Output Messages Notifications

	id integer	name character varying (100)	status character varying (100)	xmin xid	xmax xid	cmin cid	cmax cid
1	3	Alice	Not fired	832	0	0	0
2	4	Alice	Not fired	833	0	0	0
3	5	Alice	Not fired	834	0	0	0
4	6	Alice	Not fired	839	0	0	0
5	2	Alice	Fired	846	0	0	0

All queries are executed with isolation level repeatable read.

6.

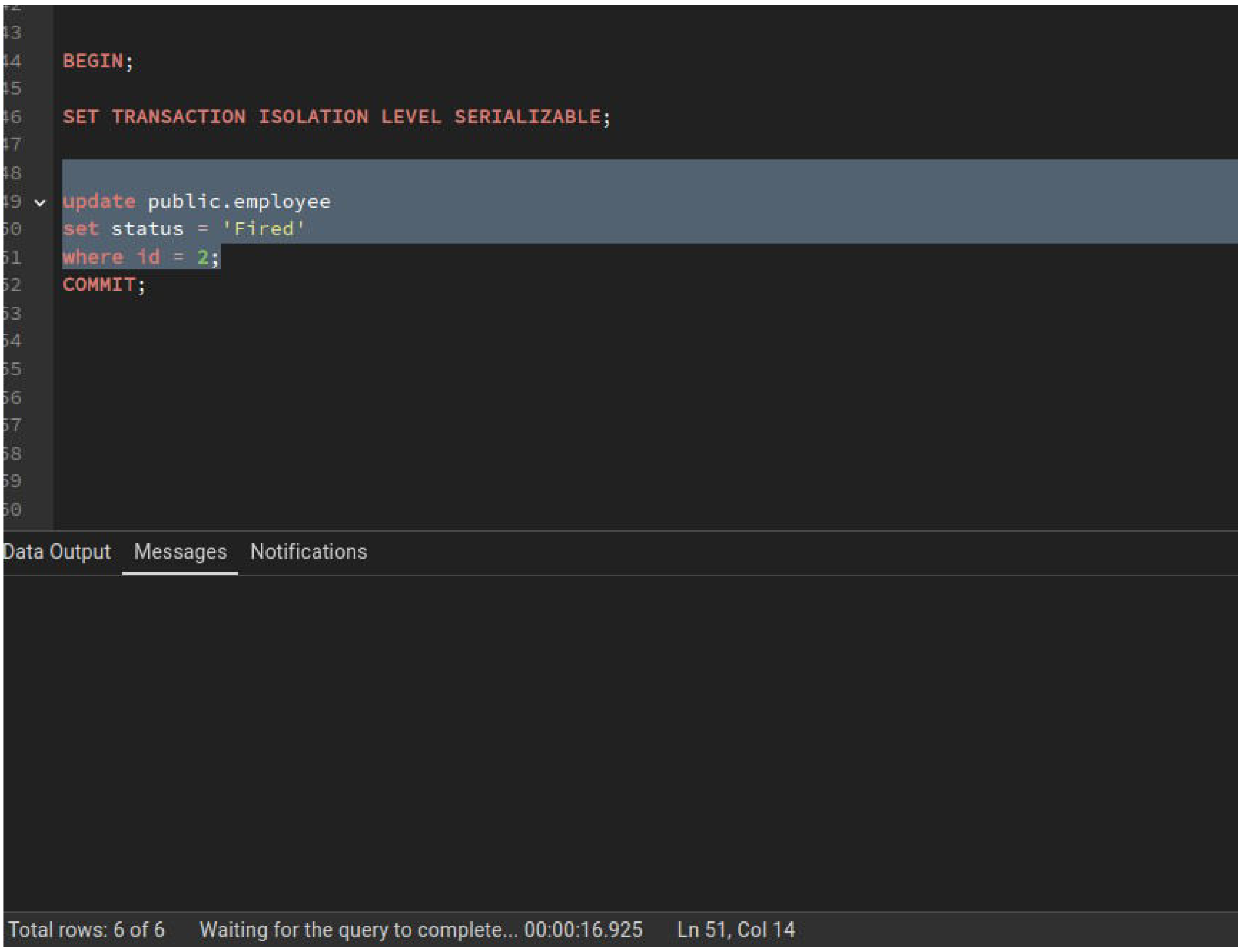


```
BEGIN;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;


update public.employee
set status = 'Fired'
where id = 2;
COMMIT;
```

Data Output Messages Notifications

```
UPDATE 1

Query returned successfully in 59 msec.
```

First transaction works well as expected ,

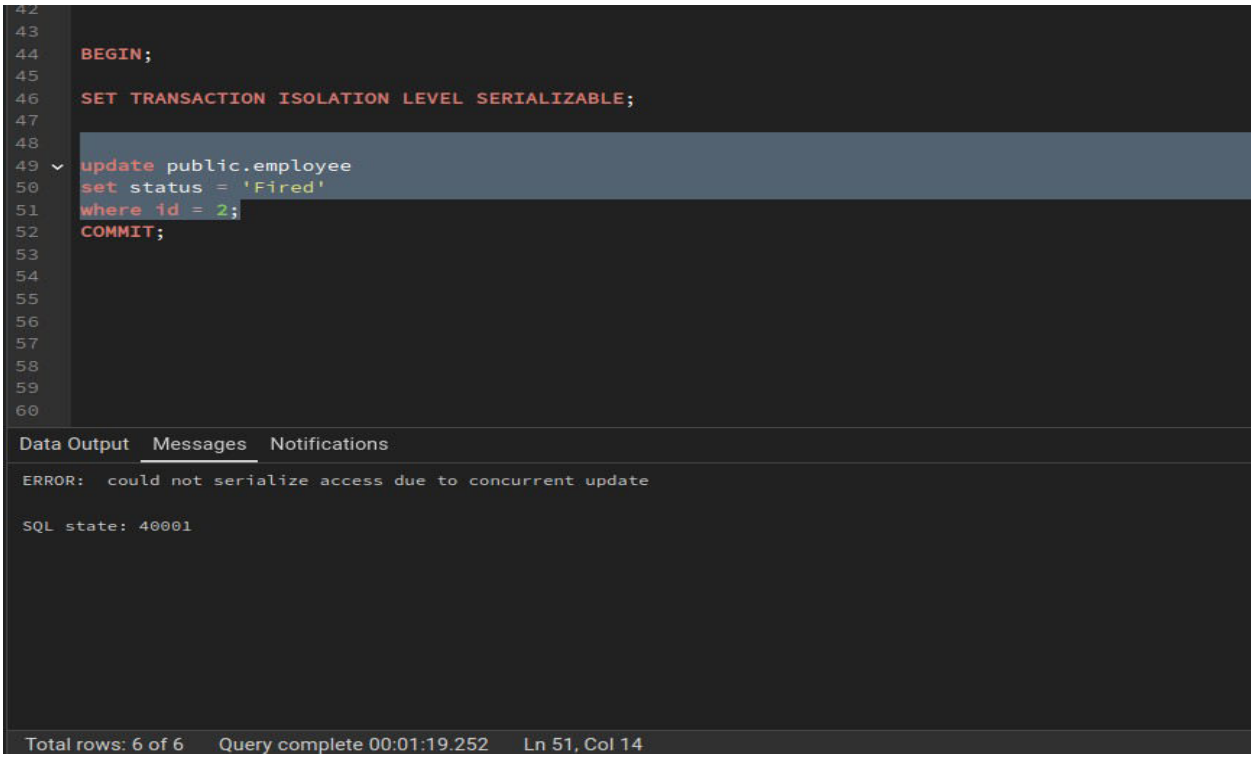


```
BEGIN;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;


update public.employee
set status = 'Fired'
where id = 2;
COMMIT;
```

Data Output Messages Notifications

Total rows: 6 of 6 Waiting for the query to complete... 00:00:16.925 Ln 51, Col 14

Second transaction waiting for the first to be complete then it will be executed.When I commit first one I got this error.



```
BEGIN;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;


update public.employee
set status = 'Fired'
where id = 2;
COMMIT;
```

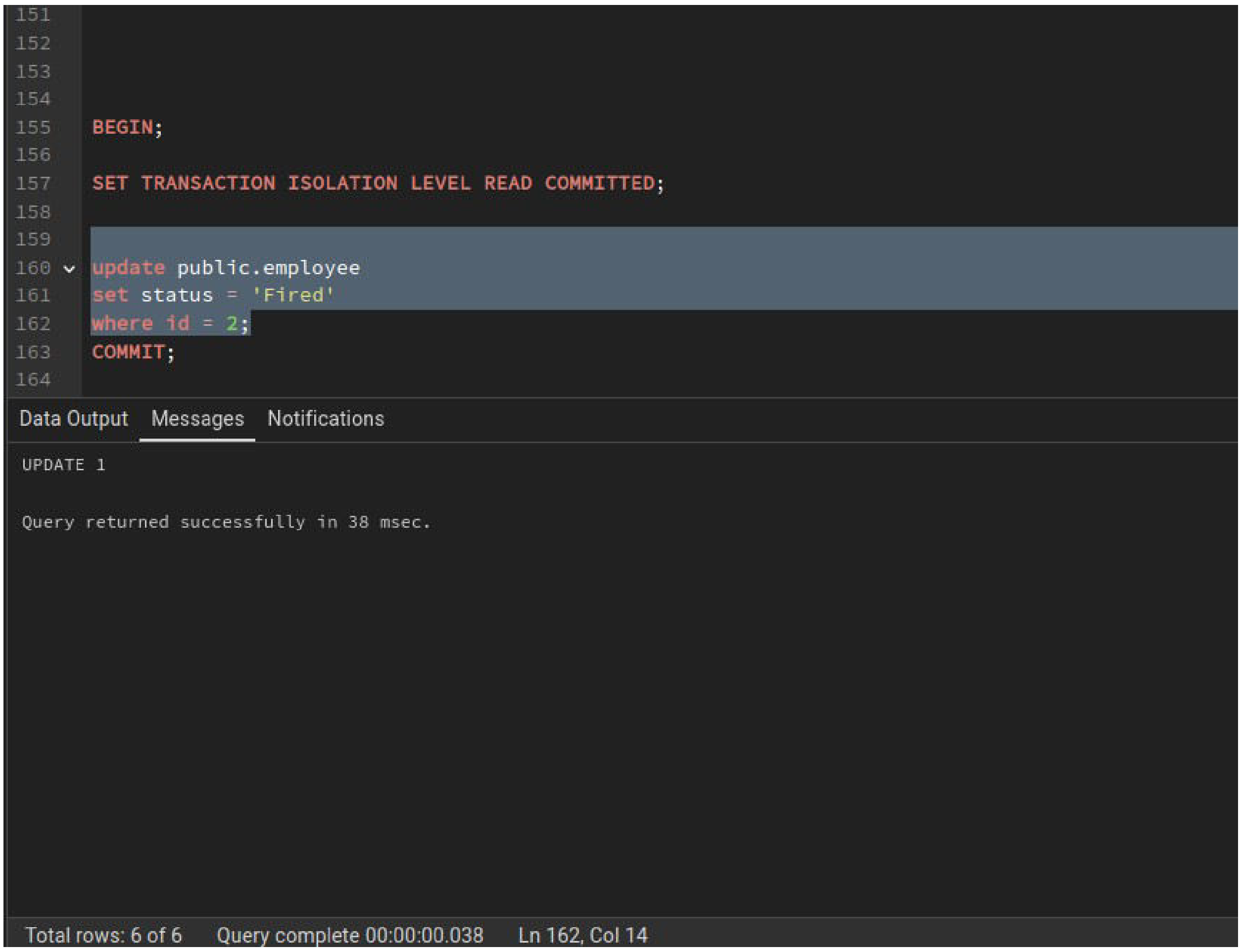
Data Output Messages Notifications

```
ERROR:  could not serialize access due to concurrent update

SQL state: 40001
```

Total rows: 6 of 6 Query complete 00:01:19.252 Ln 51, Col 14

This means Postgres locked access due to concturent update.

7.



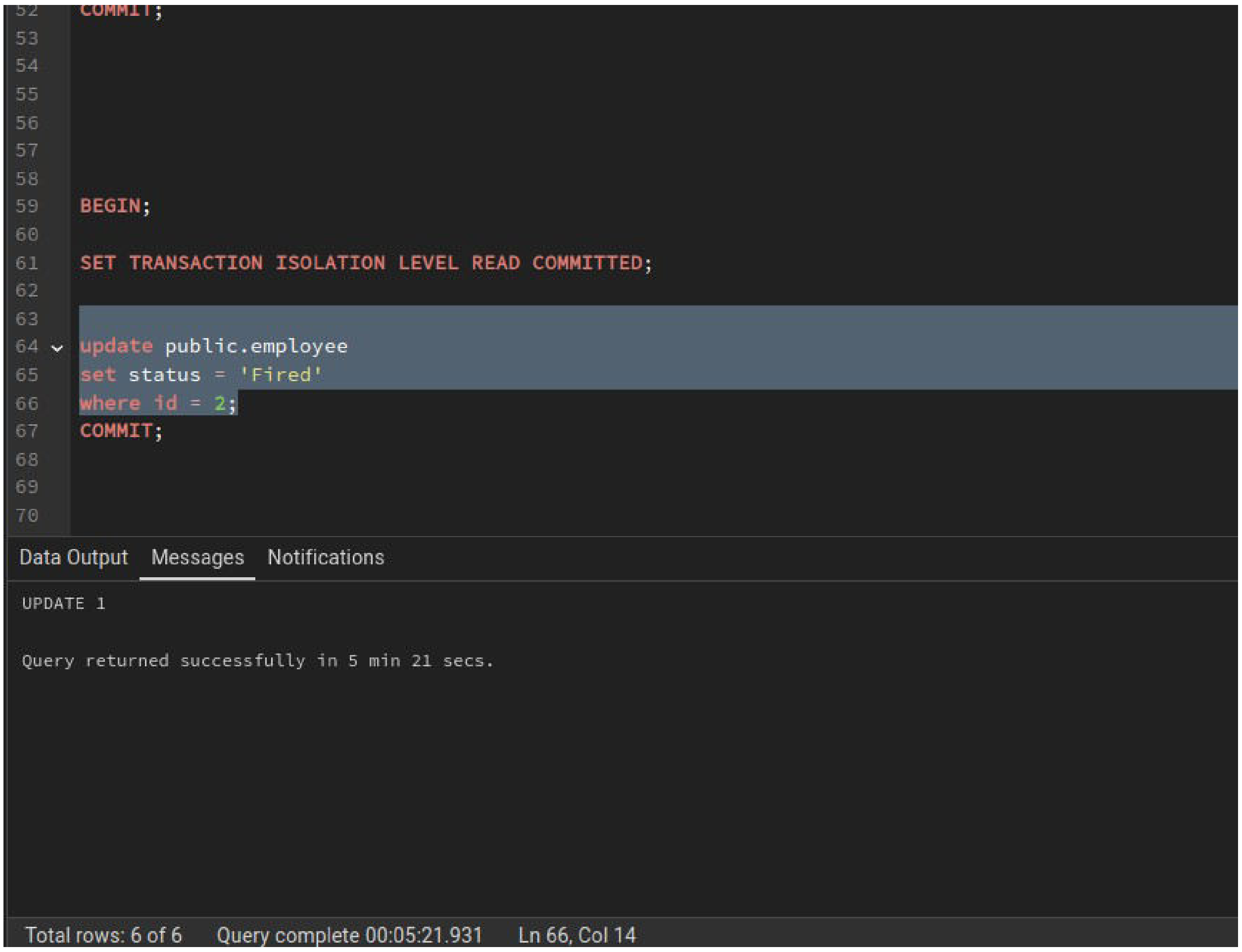
```
BEGIN;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;


update public.employee
set status = 'Fired'
where id = 2;
COMMIT;
```

Data Output   Messages   Notifications

```
UPDATE 1

Query returned successfully in 38 msec.
```

Total rows: 6 of 6   Query complete 00:00:00.038   Ln 162, Col 14

We successfully update our table in first transaction and when I run second one it’s also waiting for the first one to be commited.



```
COMMIT;






BEGIN;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;


update public.employee
set status = 'Fired'
where id = 2;
COMMIT;
```

Data Output | Messages | Notifications

```
UPDATE 1

Query returned successfully in 5 min 21 secs.
```

Total rows: 6 of 6   Query complete 00:05:21.931   Ln 66, Col 14

And after I commit first one , second one also executed successfully and updated row.