**Домашнее задание №7**

a.

1)

CREATE TABLE manager

(

id bigint PRIMARY KEY,

student\_id bigint not null,

manager\_id INTEGER,

constraint fk\_manager\_student\_id foreign key (student\_id) references student(id),

CONSTRAINT manager\_fk FOREIGN KEY (manager\_id) REFERENCES manager (id)

);

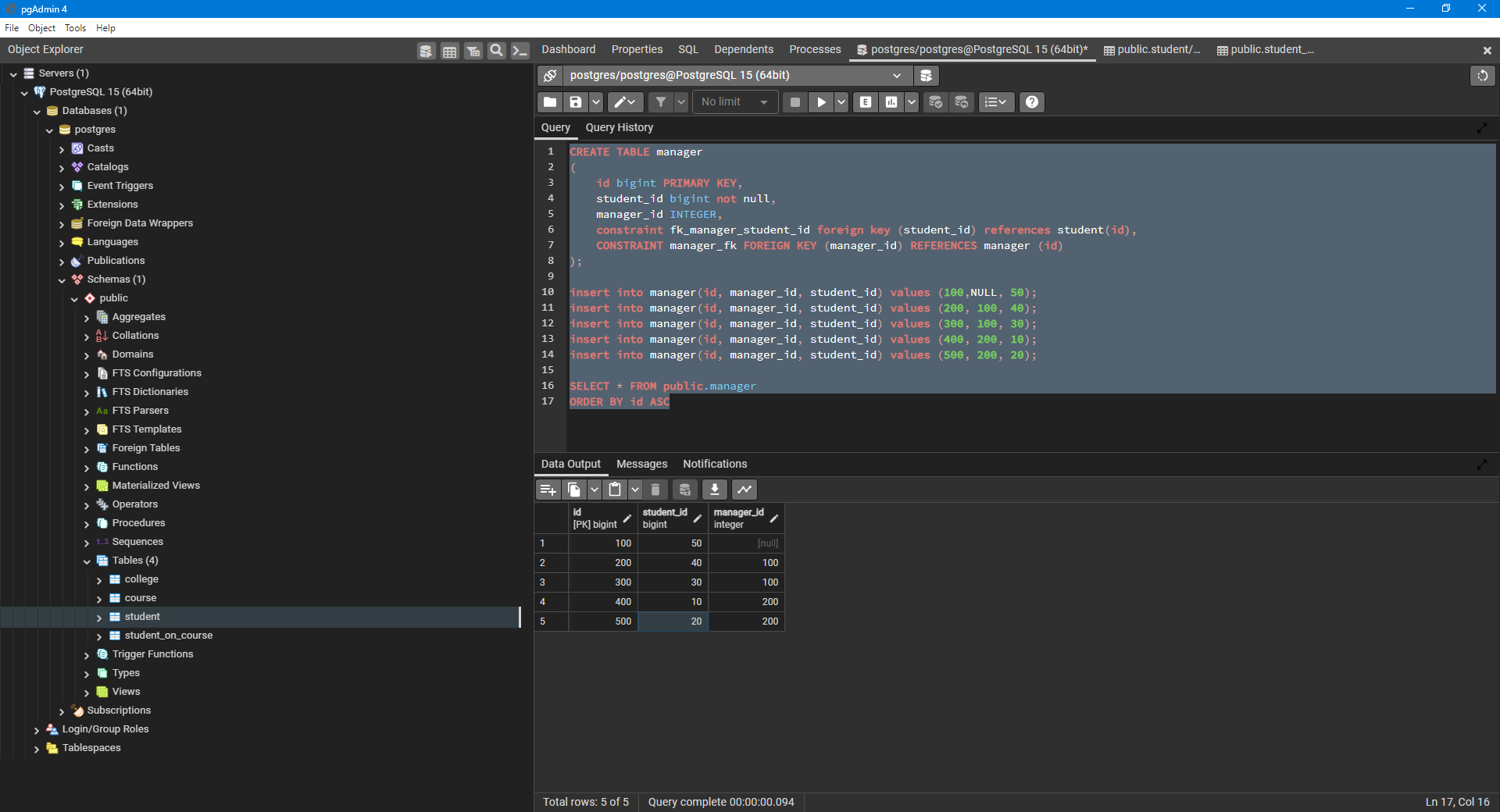
insert into manager(id, manager\_id, student\_id) values (100,NULL, 50);

insert into manager(id, manager\_id, student\_id) values (200, 100, 40);

insert into manager(id, manager\_id, student\_id) values (300, 100, 30);

insert into manager(id, manager\_id, student\_id) values (400, 200, 10);

insert into manager(id, manager\_id, student\_id) values (500, 200, 20);



2)

CREATE TABLE student\_role

(

id bigint PRIMARY KEY,

student\_id bigint not null,

role varchar not null,

svyaz\_manager\_id bigint,

constraint fk\_manager\_student\_id foreign key (student\_id) references student(id),

constraint fk\_svyaz\_manager\_id foreign key (svyaz\_manager\_id) references manager(id)

);

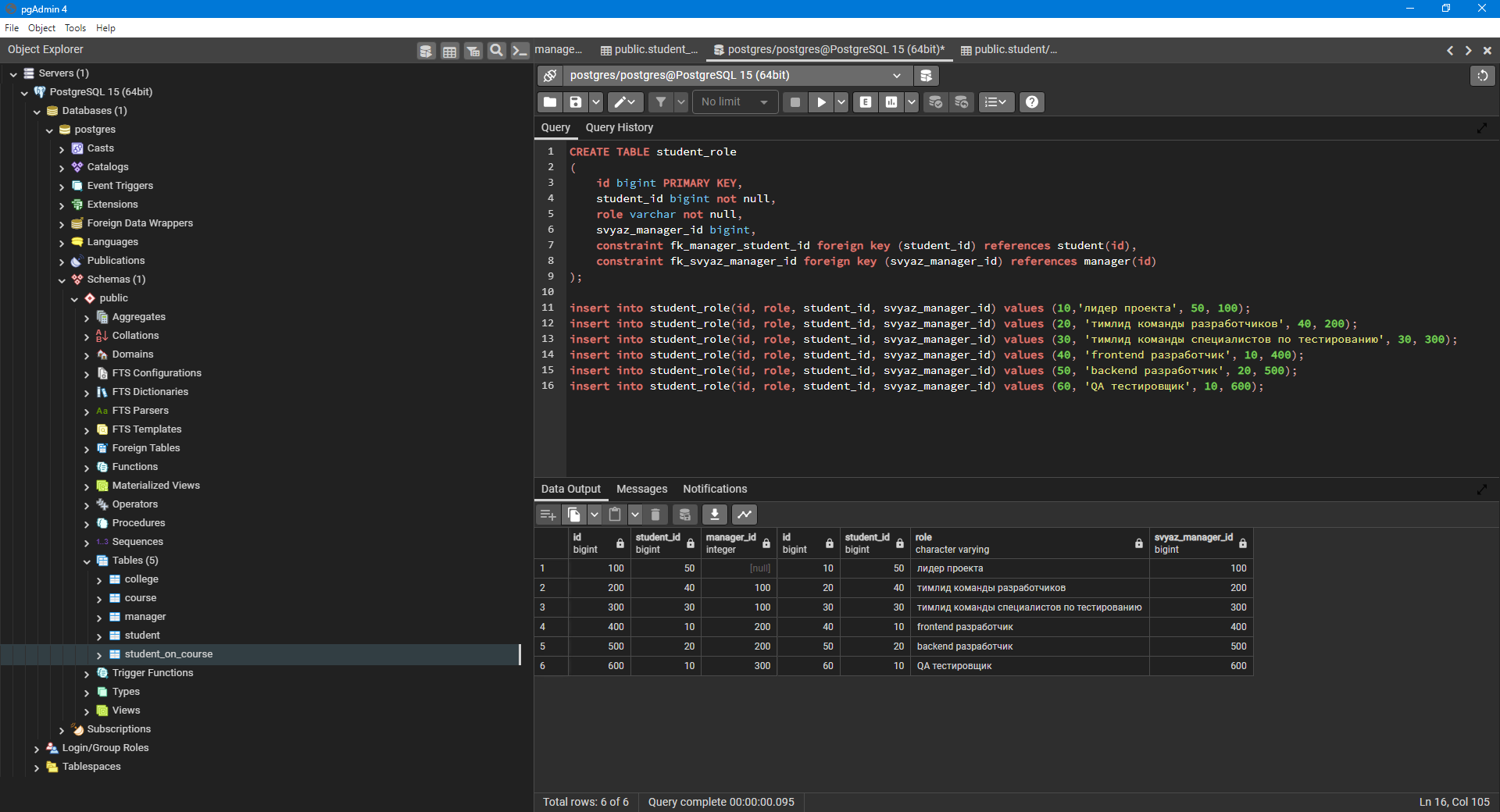
insert into student\_role(id, role, student\_id, svyaz\_manager\_id) values (10,'лидер проекта', 50, 100);

insert into student\_role(id, role, student\_id, svyaz\_manager\_id) values (20, 'тимлид команды разработчиков', 40, 200);

insert into student\_role(id, role, student\_id, svyaz\_manager\_id) values (30, 'тимлид команды специалистов по тестированию', 30, 300);

insert into student\_role(id, role, student\_id, svyaz\_manager\_id) values (40, 'frontend разработчик', 10, 400);

insert into student\_role(id, role, student\_id, svyaz\_manager\_id) values (50, 'backend разработчик', 20, 500);



b.

WITH RECURSIVE \_manager AS

(

SELECT manager.id,

manager\_id,

student.name,

1 AS level,

array [student.name] AS path,

student\_id

FROM manager

join student on manager.student\_id = student.id

WHERE manager\_id IS NULL

UNION

SELECT m1.id,

m1.manager\_id,

student.name,

level + 1 AS level,

path || student.name AS path,

m1.student\_id

FROM manager AS m1

join student on m1.student\_id = student.id

INNER JOIN \_manager AS m2 ON m1.manager\_id = m2.id

)

SELECT student.name as "студент",

role as "роль",

level as "уровень",

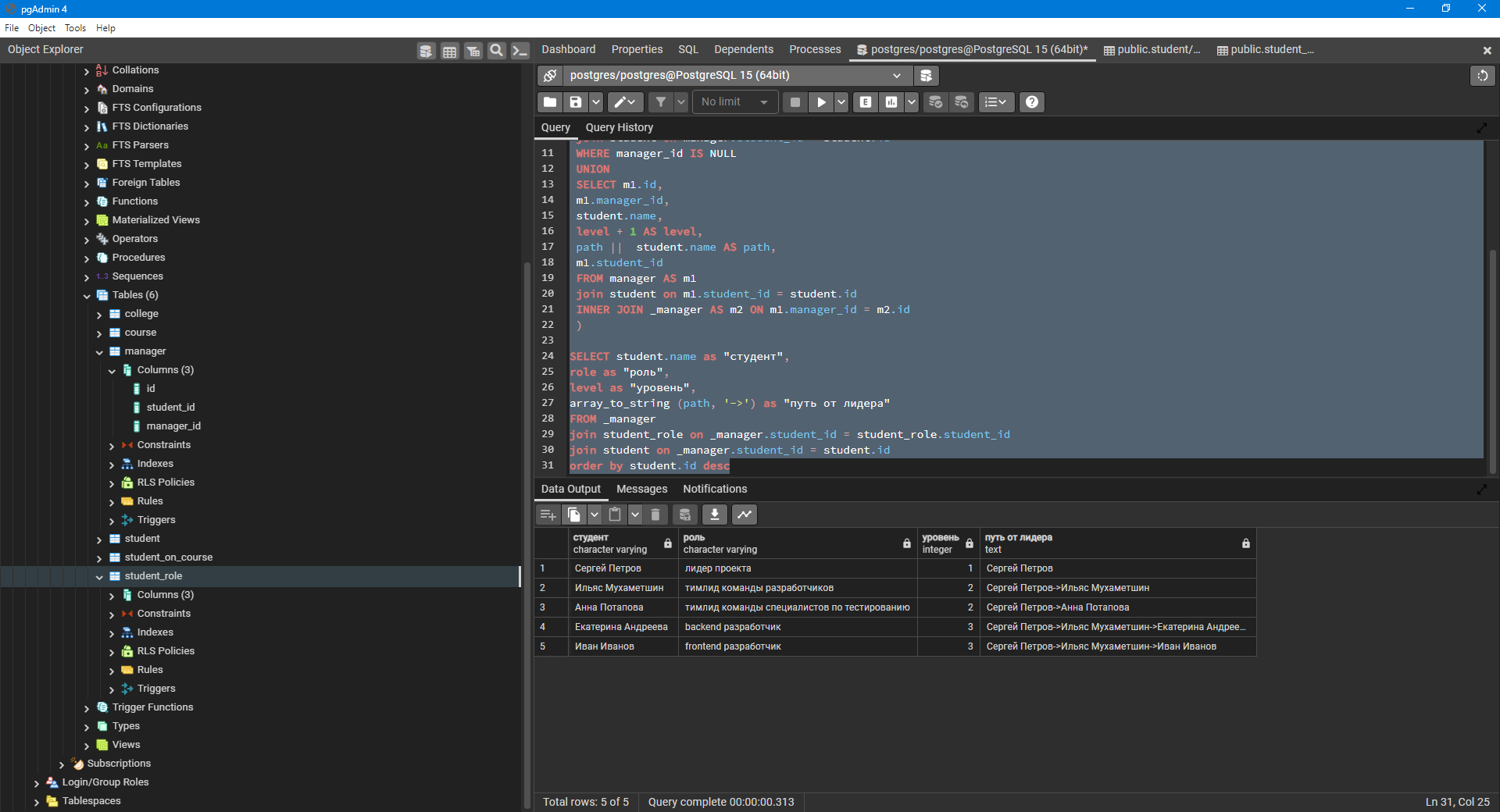
array\_to\_string (path, '->') as "путь от лидера"

FROM \_manager

join student\_role on student\_role.svyaz\_manager\_id = \_manager.id

join student on \_manager.student\_id = student.id

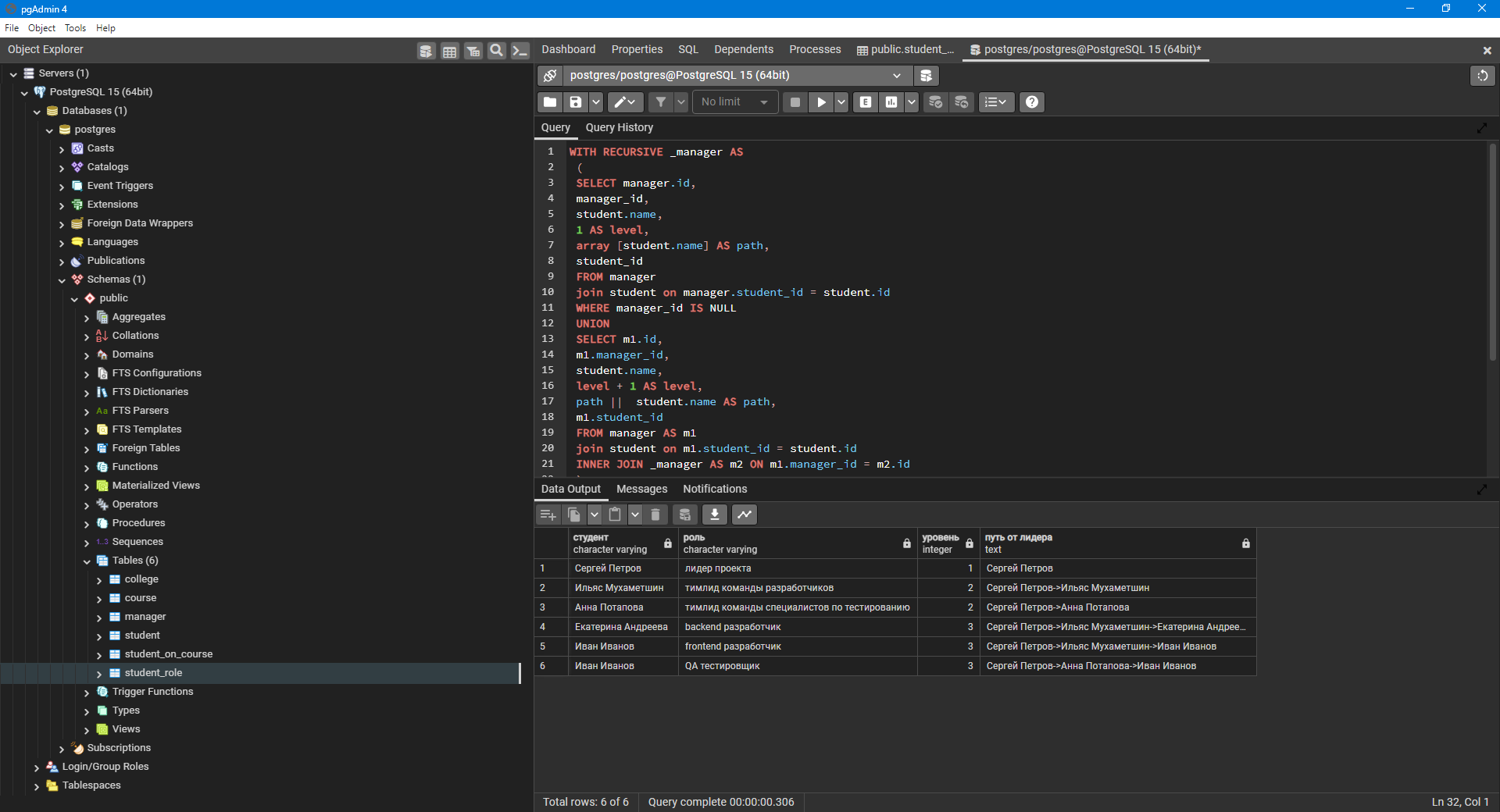
order by student.id desc



c.

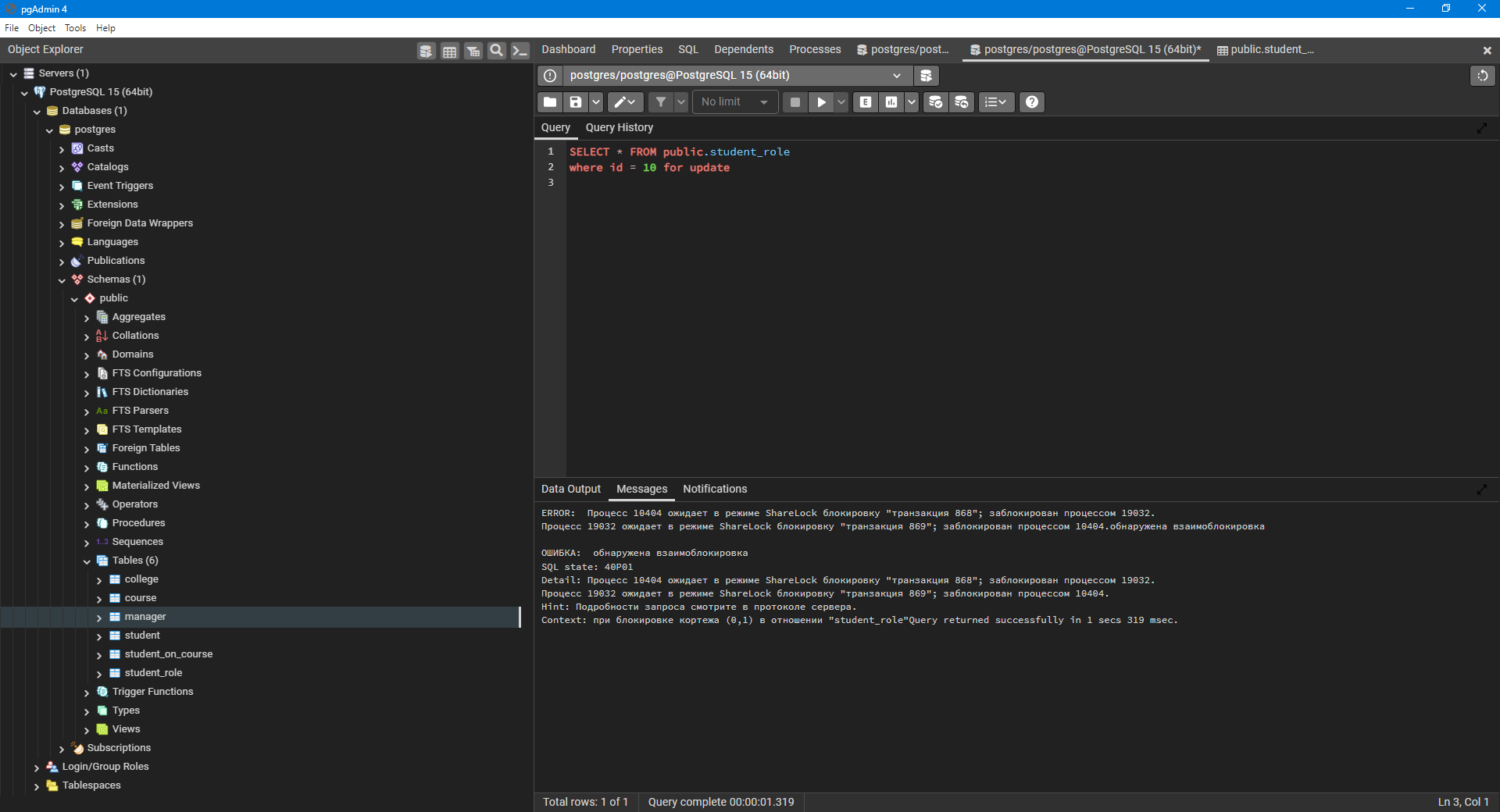
insert into manager(id, manager\_id, student\_id) values (600, 300, 10);

insert into student\_role(id, role, student\_id, svyaz\_manager\_id) values (60, 'QA тестировщик', 10, 600);



d.

Сообщение об ошибке, указывающая на то, что в терминале 2 была выполнена попытка доступа к заблокированной записи, и система обнаружила дэдлок:



e.

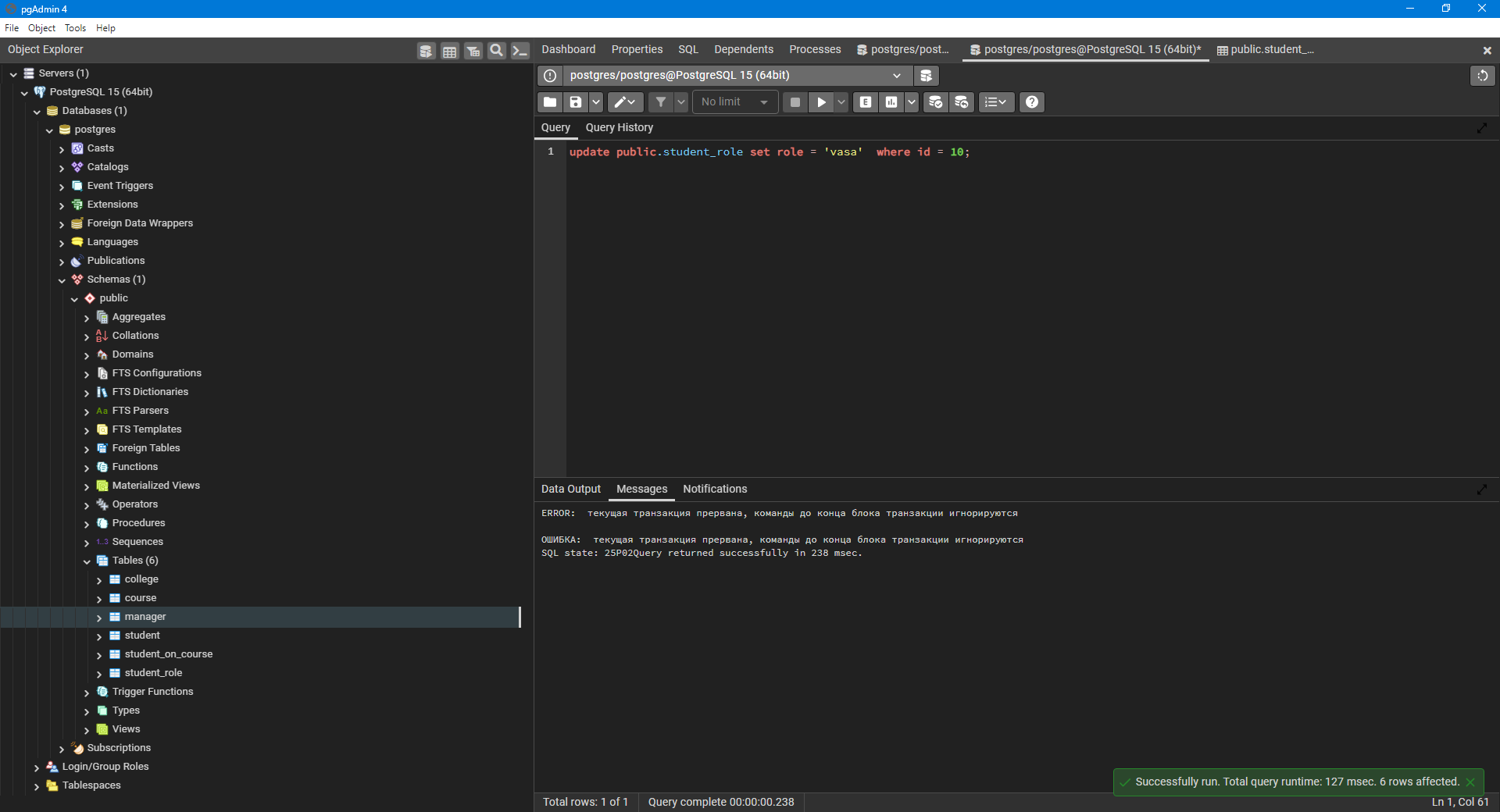
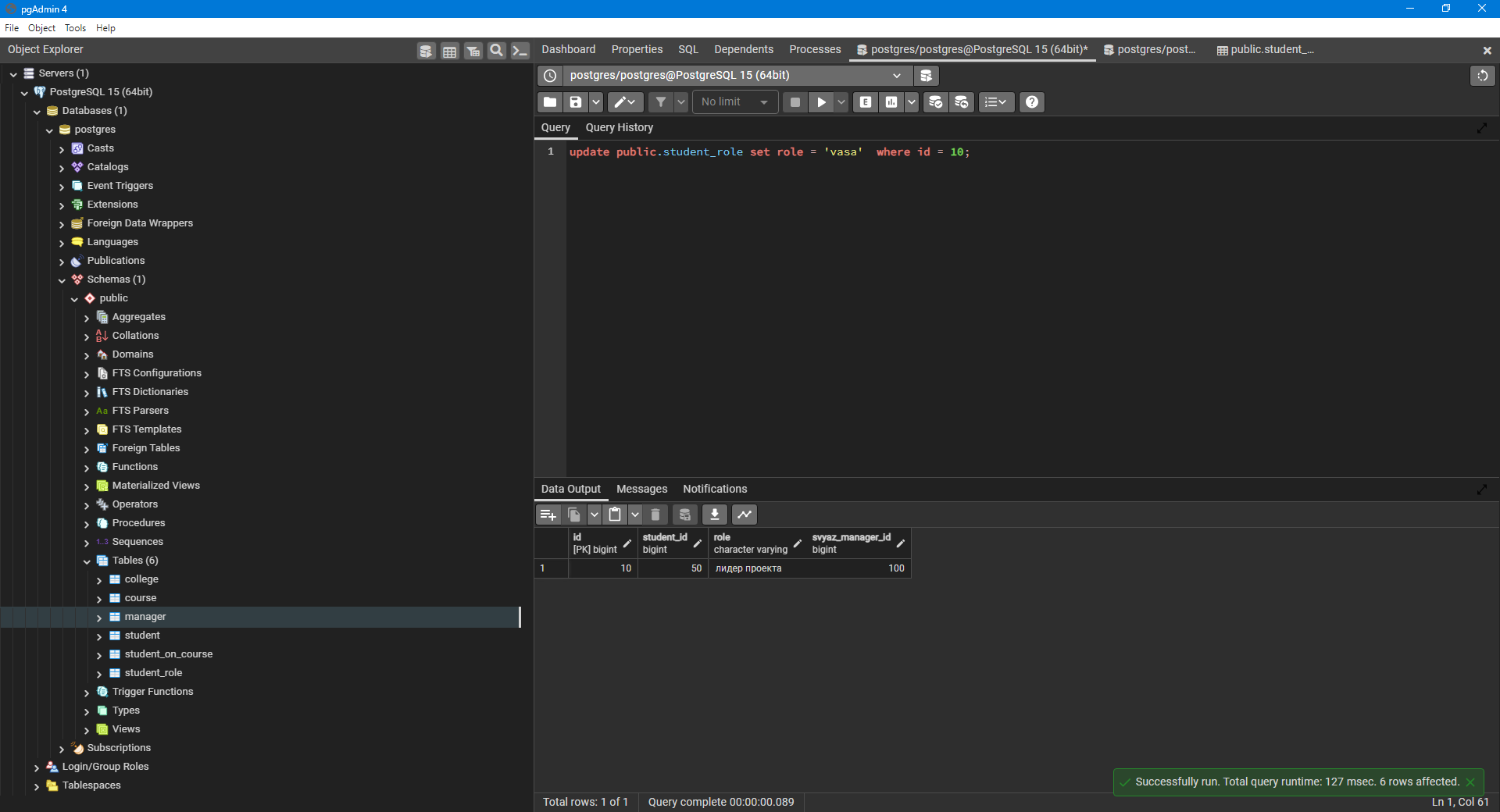
В оба терминала был последовательно введен код ниже:

begin transaction isolation level serializable;

SELECT \* FROM public.student\_role

where id = 10 for update;

Результаты попытки обновления данных в 1 и 2 терминалах:



Во втором терминале пришлось остановить процесс и ввести команду вручную, поскольку 2 окно терминала ожидало блокировки, которая удерживалась первым окном. Вторая транзакция была прервана.