

***Федеральное агентство по рыболовству***

***Федеральное государственное бюджетное образовательное***

***учреждение высшего образования***

***«Астраханский государственный технический университет»***

**Система менеджмента качества в области образования, воспитания, науки и инноваций сертифицирована**

**ООО «ДКС РУС» по международному стандарту ISO 9001:2015**

|  |  |
| --- | --- |
| Институт | Информационных технологий и коммуникаций |
| Направление | 09.03.01 Информатика и вычислительная техника |
| Профиль | «Автоматизированные системы обработки информации и управления» |
| Кафедра | «Автоматизированные системы обработки информации и управления» |

**Лабораторная работа № 10**

**«PostgreSQL. Основы языка определения данных.**

**Значения по умолчанию и ограничения целостности. Часть 2.**»

по дисциплине «СУБД PostgreSQL»

|  |  |  |
| --- | --- | --- |
|  |  | Работа выполнена студентом группы ДИНРБ-31  Кузургалиев Р.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Фамилия И.О.) подпись |
|  |  | Проверил работу:  ст. преподаватель Мамлеева А.Р,\_\_\_\_\_\_\_\_\_\_\_\_  (ученая степень, ученое звание, Фамилия И.О.) |

Работа защищена

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**АСТРАХАНЬ – 2024**

**СОДЕРЖАНИЕ ОТЧЕТА**

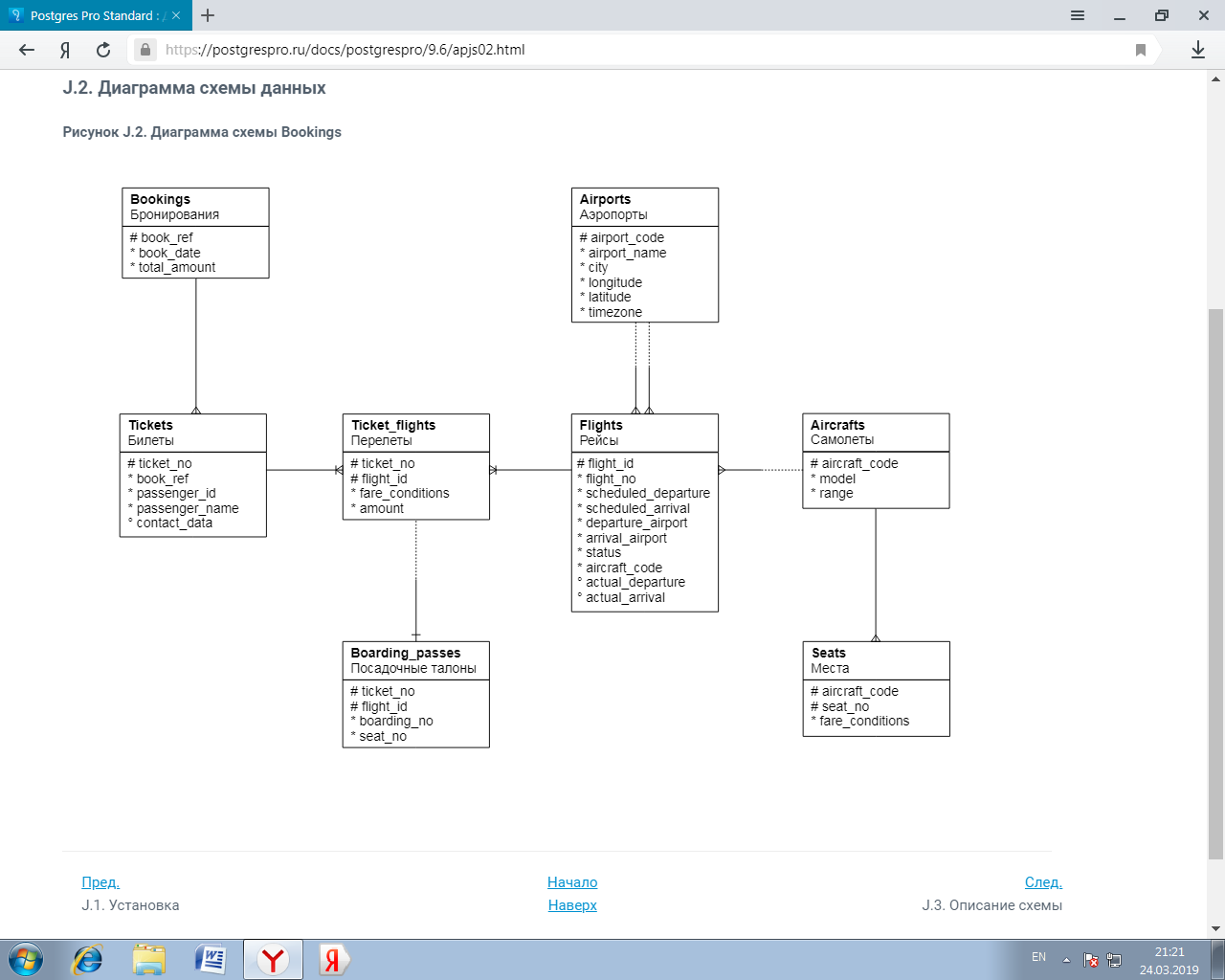
**Цель работы**: Получить практические навыки в использовании команд языка определения данных DDL в СУБД PostgreSQL.

**Литература**: 1. Лекция № 3. Раздел «Значения по умолчанию и ограничения

целостности».

**2. Моргунов Е.П. PostgreSQL. Основы языка SQL: учеб. пособие. – СПб.: БХВ-Петербург, 2018.- 336 с.: ил. (!!!)**

Схема базы данных «**Авиаперевозки**» (**avia**):



**Задание 1.**

Создайте базу данных «Авиаперевозки» (avia) и вставьте в таблицы несколько строк.

CREATE TABLE airports

( airport\_code char(3) NOT NULL, -- Код аэропорта

airport\_name text NOT NULL, -- Название аэропорта

city text NOT NULL, -- Город

longitude float NOT NULL, -- Координаты аэропорта: долгота

latitude float NOT NULL, -- Координаты аэропорта: широта

timezone text NOT NULL, -- Часовой пояс

PRIMARY KEY(airport\_code)

);

CREATE TABLE aircrafts

( aircraft\_code char(3) NOT NULL, -- Код самолета, IATA

model text NOT NULL, -- Модель самолета

range integer NOT NULL, -- Максимальная дальность полета (км)

CHECK(range > 0),

PRIMARY KEY(aircraft\_code)

);

CREATE TABLE seats

( aircraft\_code char(3) NOT NULL, -- код самолета, IATA

seat\_no varchar(4) NOT NULL, -- Номер места

fare\_condition varchar(10) NOT NULL, -- Класс обслуживания

CHECK(fare\_condition IN ('Economy', 'Comfort', 'Business')),

PRIMARY KEY(aircraft\_code, seat\_no),

FOREIGN KEY(aircraft\_code)

REFERENCES aircrafts(aircraft\_code)

ON DELETE CASCADE

);

CREATE TABLE bookings

( book\_ref char(6) NOT NULL, -- Номер бронирования

book\_date timestamptz NOT NULL,

total\_amount numeric(10,2) NOT NULL,

PRIMARY KEY(book\_ref)

);

CREATE TABLE tickets

( ticket\_no char(13) NOT NULL, -- Номер билета

book\_ref char(6) NOT NULL, -- Номер бронирования

passenger\_id varchar(20) NOT NULL, -- ID пассажира

passenger\_name text NOT NULL, -- Имя пассажира

contact\_data jsonb, -- Контактные данные пассажира

PRIMARY KEY(ticket\_no),

FOREIGN KEY(book\_ref)

REFERENCES bookings(book\_ref)

);

CREATE TABLE flights

( flight\_id serial NOT NULL, -- Идентификатор рейса

flight\_no char(6) NOT NULL, -- Номер рейса

scheduled\_departure timestamptz NOT NULL, --Время вылета по расписанию

departure\_airport char(3) NOT NULL, -- Время прилета по расписанию

arrival\_airport char(3) NOT NULL, -- Аэропорт отправления

status varchar(20) NOT NULL, -- Аэропорт прибытия

aircraft\_code char(3) NOT NULL, -- Код самолета, IATA

actual\_departure timestamptz, -- Фактическое время вылета

actual\_arrival timestamptz, -- Фактическое время прилета

CHECK(actual\_arrival > scheduled\_departure),

CHECK(status IN ('On Time', 'Delayed', 'Departed',

'Arrives', 'Scheduled', 'Canceled')),

CHECK(actual\_arrival IS NULL OR

(actual\_departure IS NOT NULL AND

actual\_arrival IS NOT NULL AND

actual\_arrival > actual\_departure)),

PRIMARY KEY(flight\_id),

UNIQUE(flight\_no, scheduled\_departure),

FOREIGN KEY(aircraft\_code) REFERENCES aircrafts(aircraft\_code),

FOREIGN KEY(arrival\_airport) REFERENCES airports(airport\_code),

FOREIGN KEY(departure\_airport) REFERENCES airports(airport\_code)

);

CREATE TABLE ticket\_flight

( ticket\_no char(13) NOT NULL, -- Номер билета

flight\_id integer NOT NULL, -- Идентификатор рейса

fare\_conditions varchar(10) NOT NULL, -- Класс обслуживания

amount numeric(10,2) NOT NULL, -- Стоимость перелета

CHECK (amount >= 0),

CHECK (fare\_conditions IN ('Economy', 'Comfort', 'Business')),

PRIMARY KEY (ticket\_no, flight\_id),

FOREIGN KEY (flight\_id) REFERENCES flights (flight\_id),

FOREIGN KEY (ticket\_no) REFERENCES tickets (ticket\_no)

);

CREATE TABLE boording\_passes

( ticket\_no char(13) NOT NULL, -- Номер билета

flight\_id integer NOT NULL, -- Идентификатор рейса

boarding\_no integer NOT NULL, -- Номер посадочного талона

seat\_no varchar(4) NOT NULL, -- Номер места

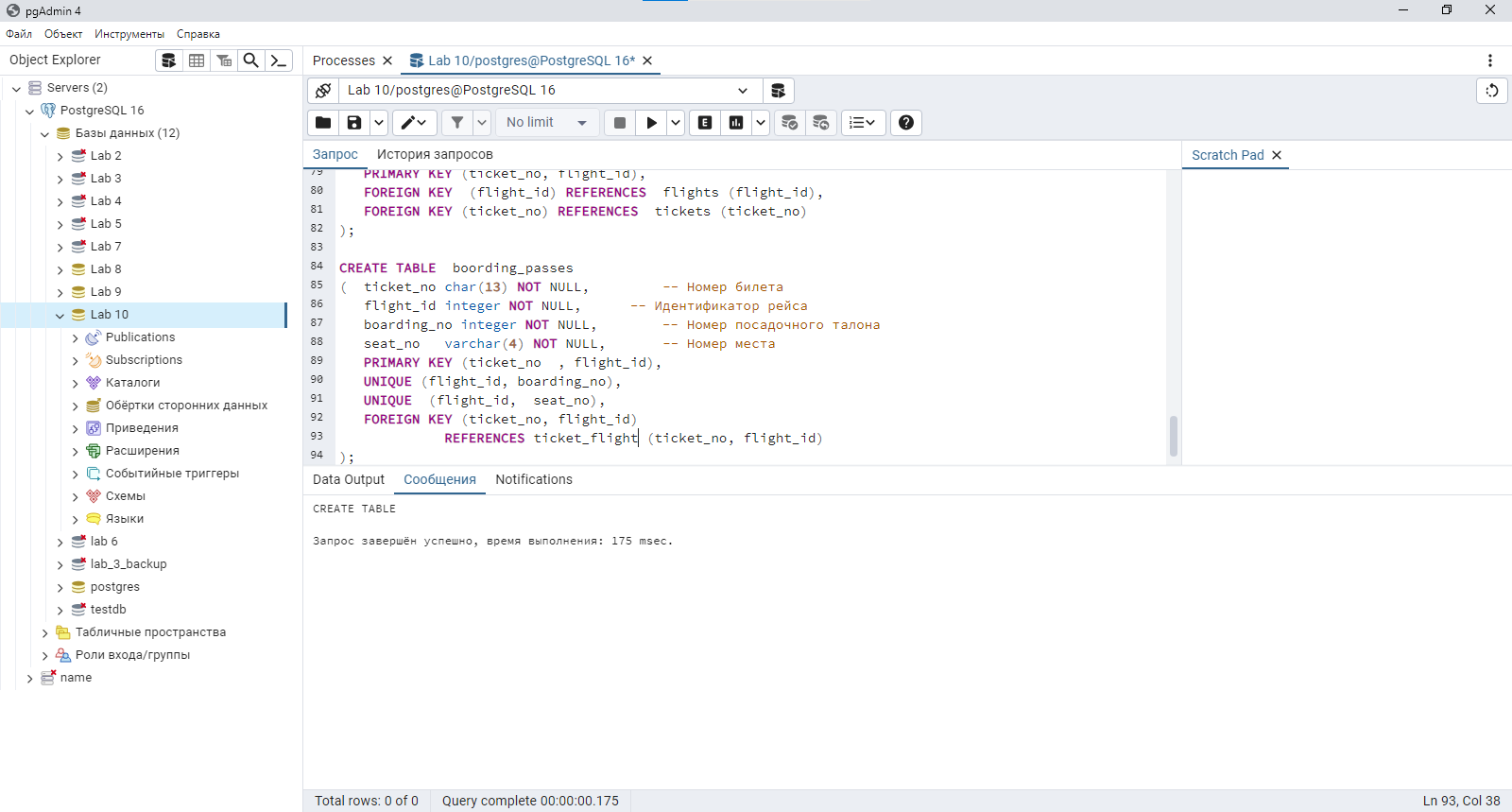
PRIMARY KEY (ticket\_no , flight\_id),

UNIQUE (flight\_id, boarding\_no),

UNIQUE (flight\_id, seat\_no),

FOREIGN KEY (ticket\_no, flight\_id)

REFERENCES ticket\_flight (ticket\_no, flight\_id)

);****

**Задание 2.**

В таблице «Рейсы» (flights) есть ограничение, которое регулирует соотношения значений фактического времени вылета и фактического времени прилета.

Вопрос: не является ли выражение actual\_arrival IS NOT NULL второй частью условного оператора OR избыточным?

CREATE TABLE flights

( …

CHECK(actual\_arruval IS NULL OR

(actual\_departure IS NOT NULL AND

actual\_arrival IS NOT NULL AND

actual\_arrival > actual\_departure)),

…

);

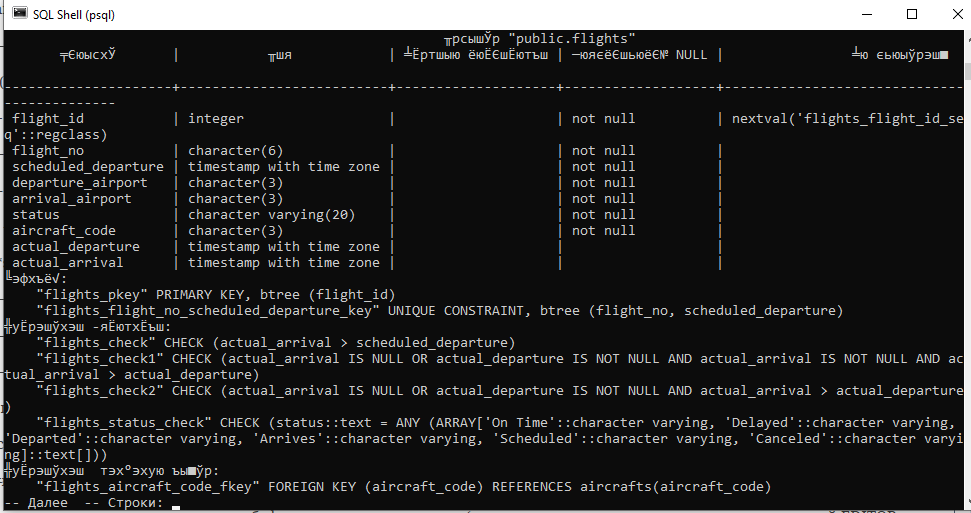
Проверьте Ваши предположения на практике.

Для этого сначала удалите существующее ограничение с помощью команды:

ALTER TABLE flights DROP CONSTRAINT имя\_ограничения;

Как определить имя этого ограничения?

С помощью команды **/d flights** получите описание таблицы **flights**, а нем есть названия всех ограничений.



Затем создайте это же ограничение, но в модифицируемом виде:

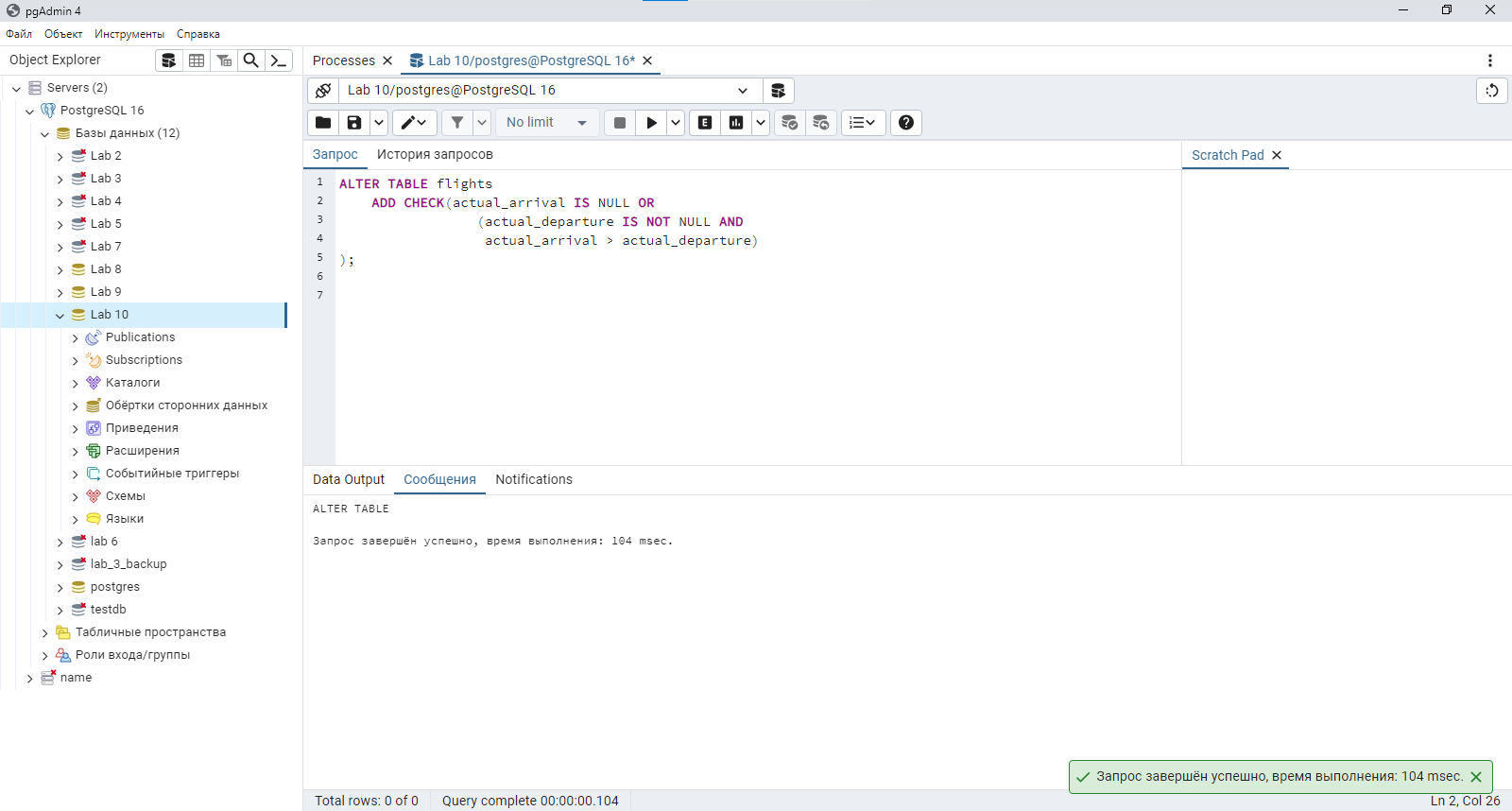
ALTER TABLE flights

ADD CHECK(actual\_arruval IS NULL OR

(actual\_departure IS NOT NULL AND

actual\_arrival > actual\_departure)

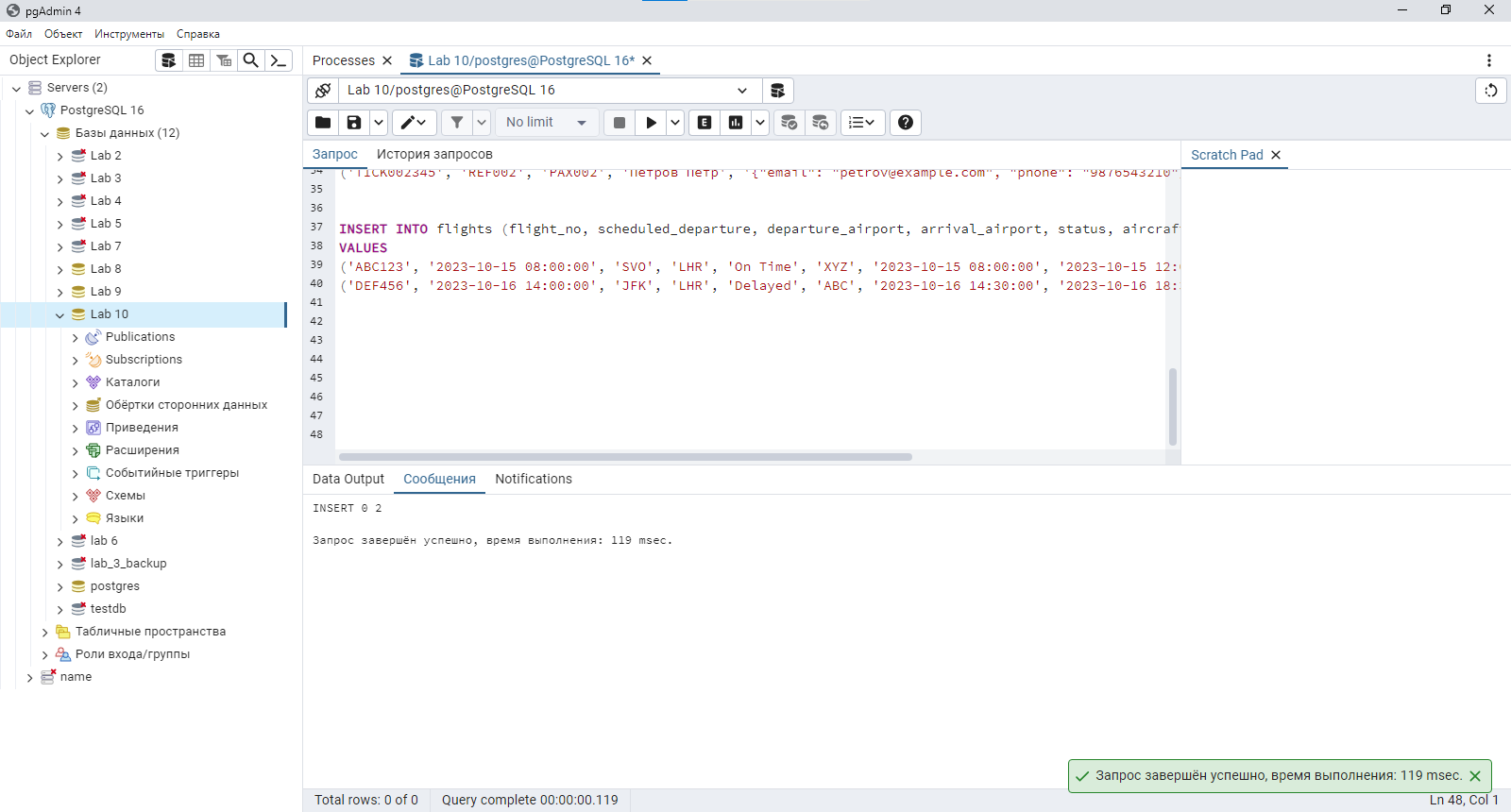
);



Добавьте в таблицу **flights** две-три строки, подбирая такие значения атрибутов **actual\_departure** и **actual\_arrival**, чтобы проверить все возможные исходы этих проверок.

Конечно, вместо добавления новых строк можно модифицировать одну и ту же строку с помощью команды **UPDATE**.

*Примечание: таблица* ***flights*** *связана внешними ключами с таблицами* ***airport****s и* ***aircrafts****. Добавьте в эти таблицы несколько строк до момента добавления новых строк в таблицу* ***flights****.*

****

**Задание 3.**

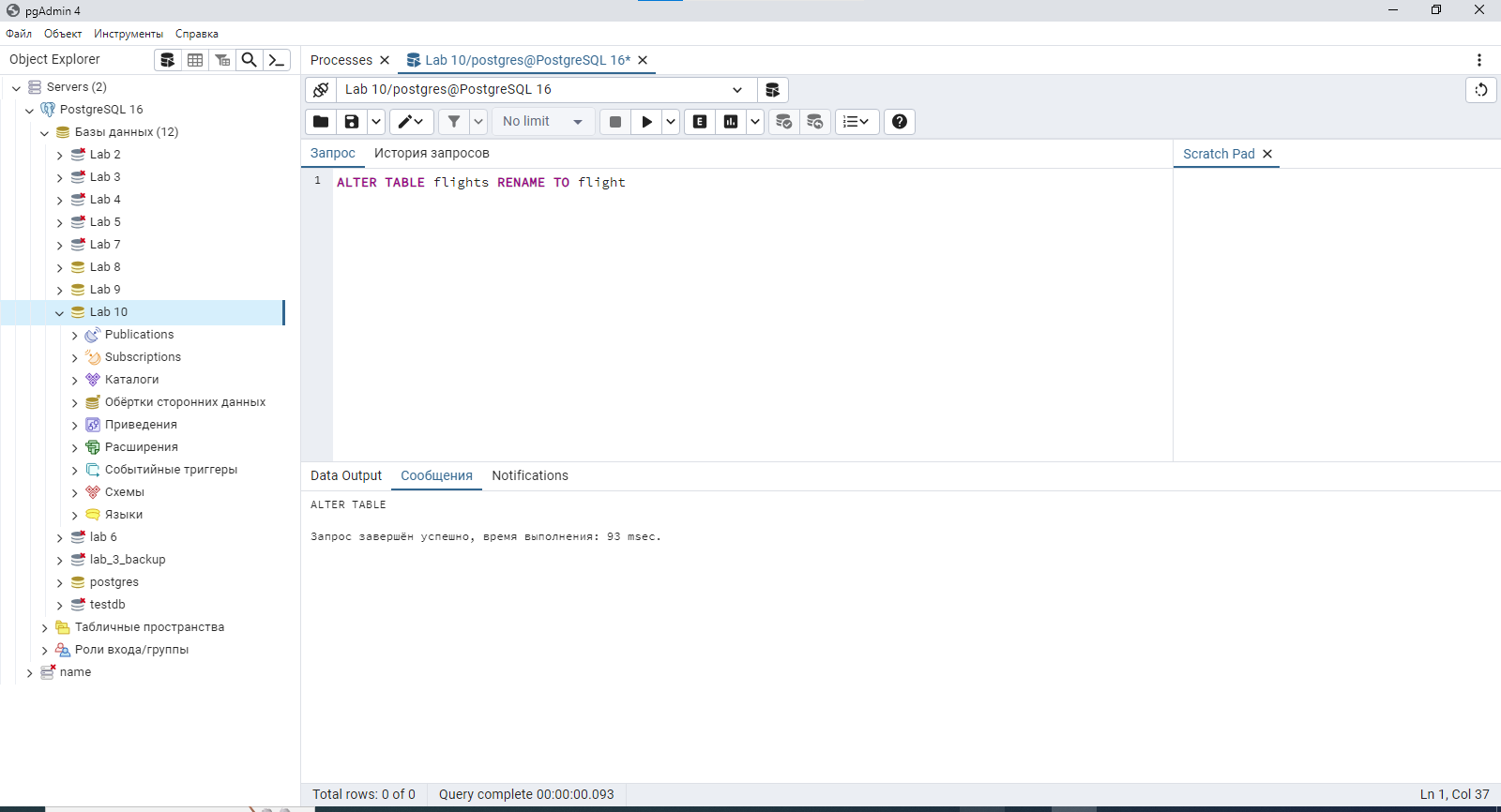
Команда ALTER TABLE позволяет переименовать таблицу. Например:

ALTER TABLE table\_name RENAME TO new\_table\_name;

Поскольку в командах создания таблиц базы данных «Авиаперевозки» не указаны имена ограничителей для первичных и внешних ключей, то их имена были сформированы автоматически самой СУБД.

***Как Вы считаете, получили ли эти ограничения новые имена после переименования таблицы?***

Проверьте Ваши предположения, выполнив операцию с одной из таблиц базы данных «Авиаперевозки», имеющих внешние ключи.

****

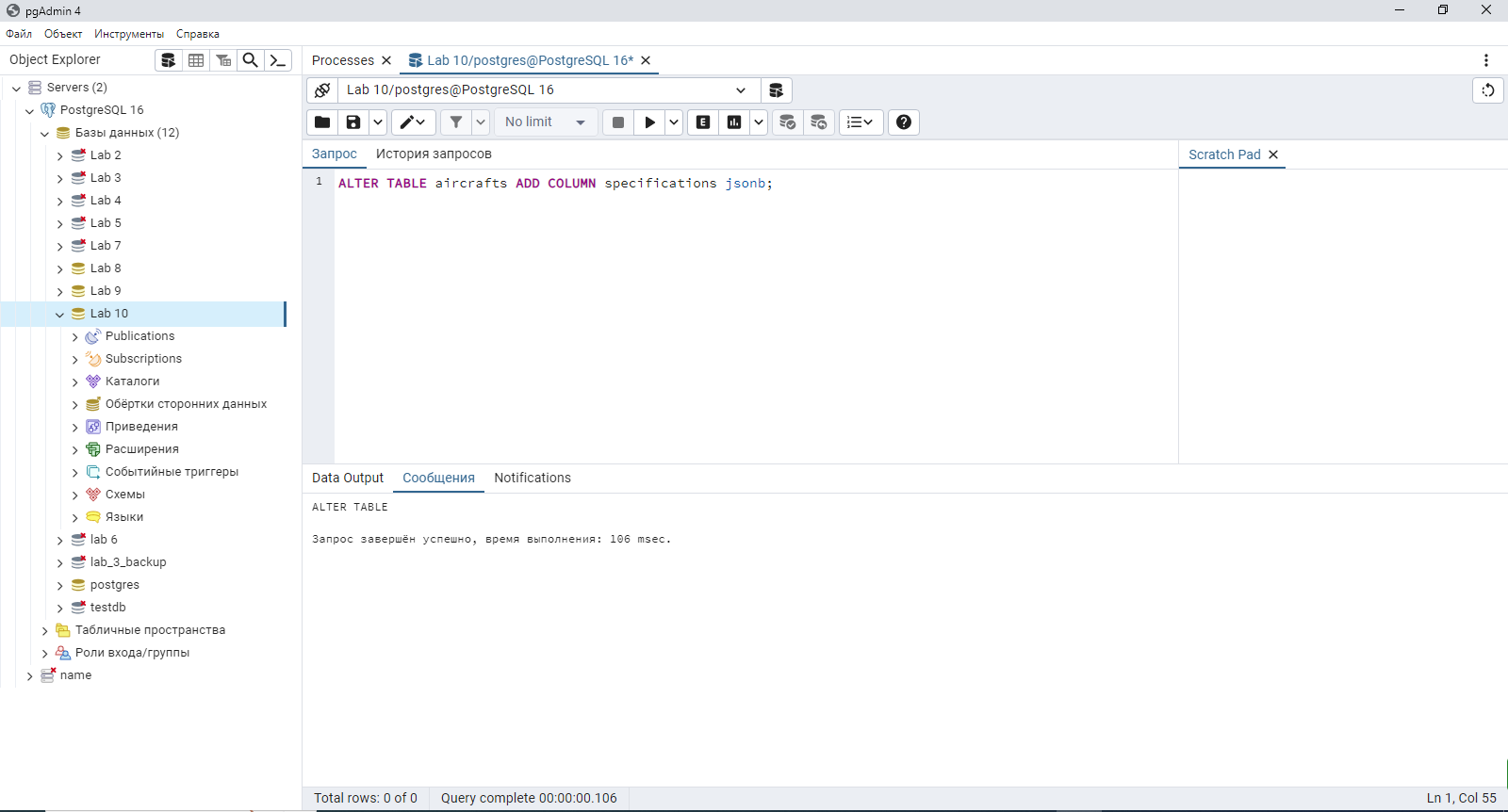
**Задание 4.**

Предположим, что нам понадобилось иметь в базе данных сведения о технических характеристиках самолетов, эксплуатируемых в авиакомпании. Пусть это будут такие сведения, как число членов экипажа (пилоты), тип двигателя и их количество.

Следовательно, необходимо добавить новый столбец в таблицу «**Самолеты**» (**aircrafts**).

Дадим ему имя **specifications**, а в качестве типа данных выберем **jsonb**. Если впоследствии потребуется добавить и другие характеристики, то мы сможем это сделать, не модифицируя определение таблицы.

ALTER TABLE aircrafts ADD COLUMN specifications jsonb;



Добавим сведения для модели самолета, например, Airbus A320-200:

UPDATE aircrafts

SET specifications =

‘{“crew”: 2,

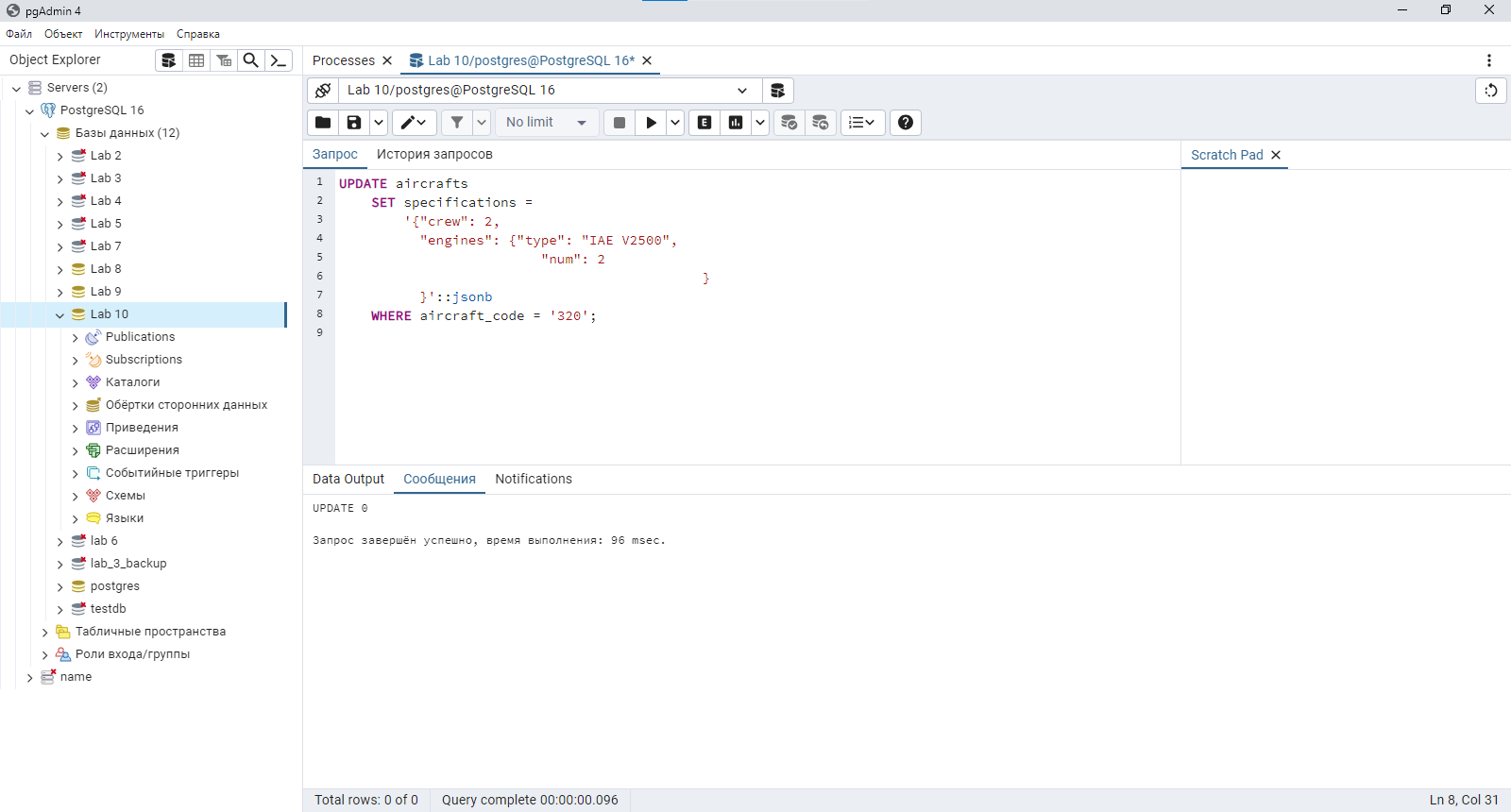
“engines”: {“type”: “IAE V2500”,

“num”: 2

}

}’::jsonb

WHERE aircraft\_code = ‘320’;

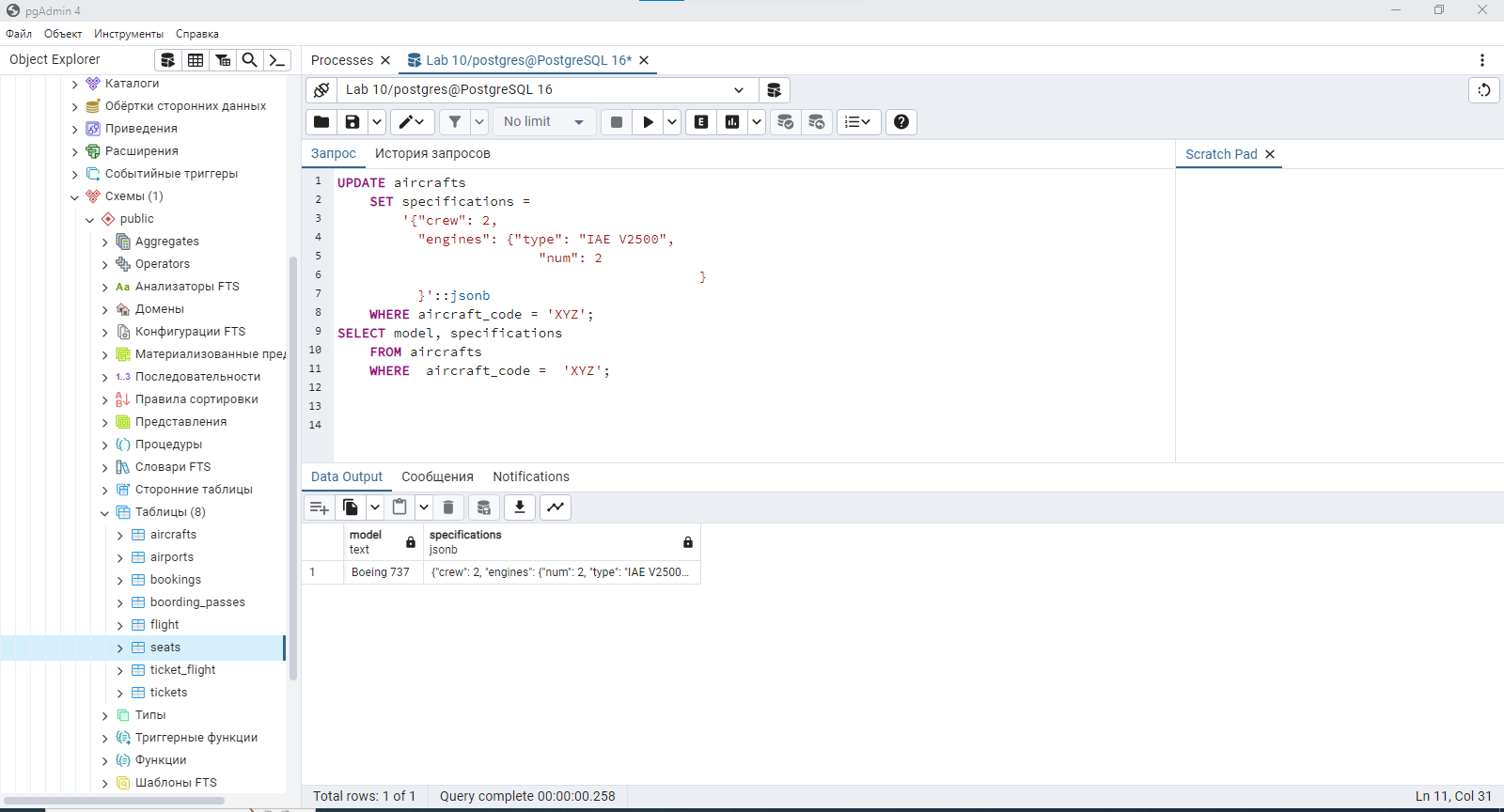


Посмотрите, что получились:

SELECT model, specifications

FROM aircrafts

WHERE aircraft\_code = ‘320’;

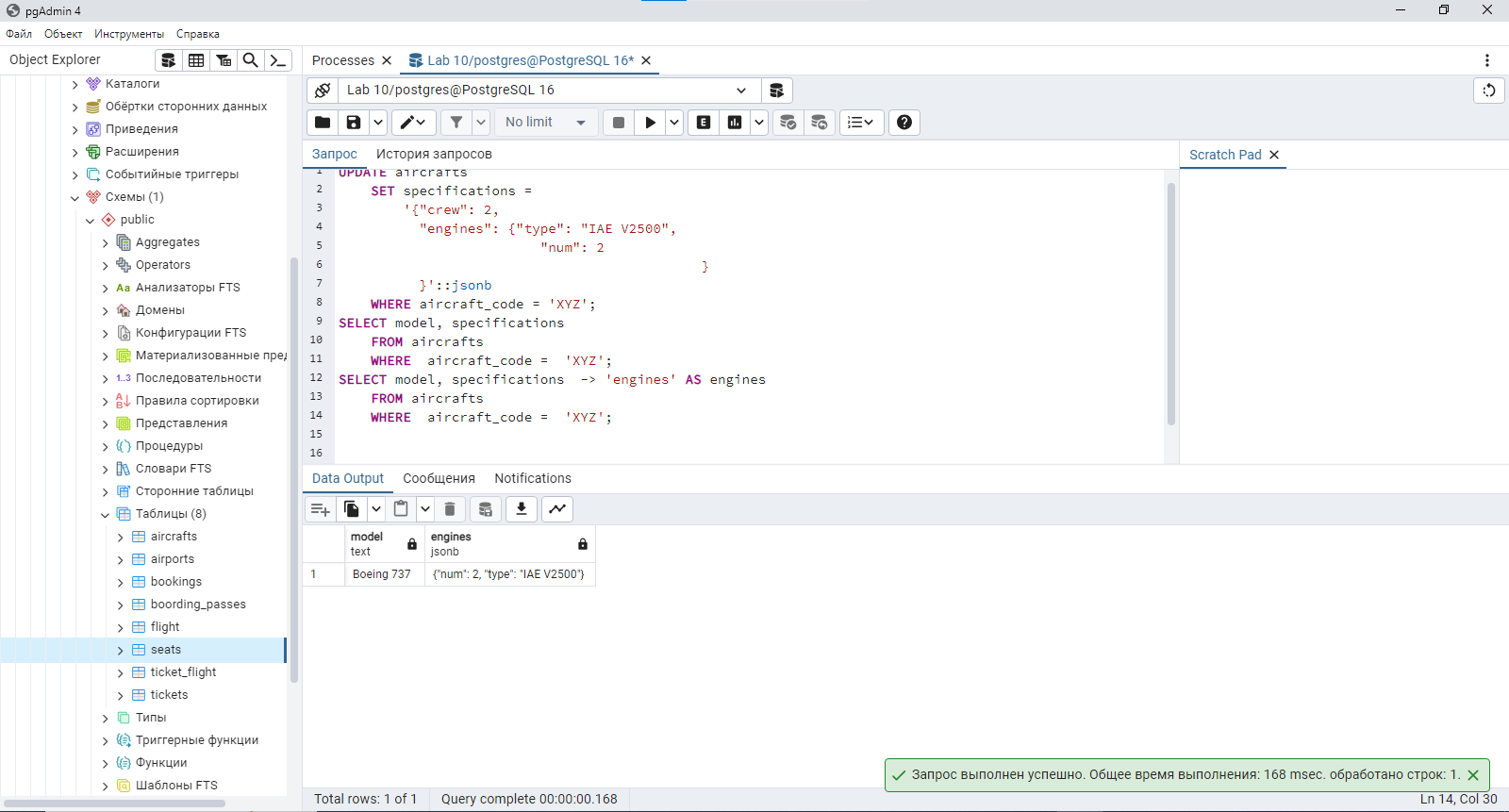


Можно получить только сведения о двигателе:

SELECT model, specifications -> 'engines' AS engines

FROM aircrafts

WHERE aircraft\_code = ‘320’;



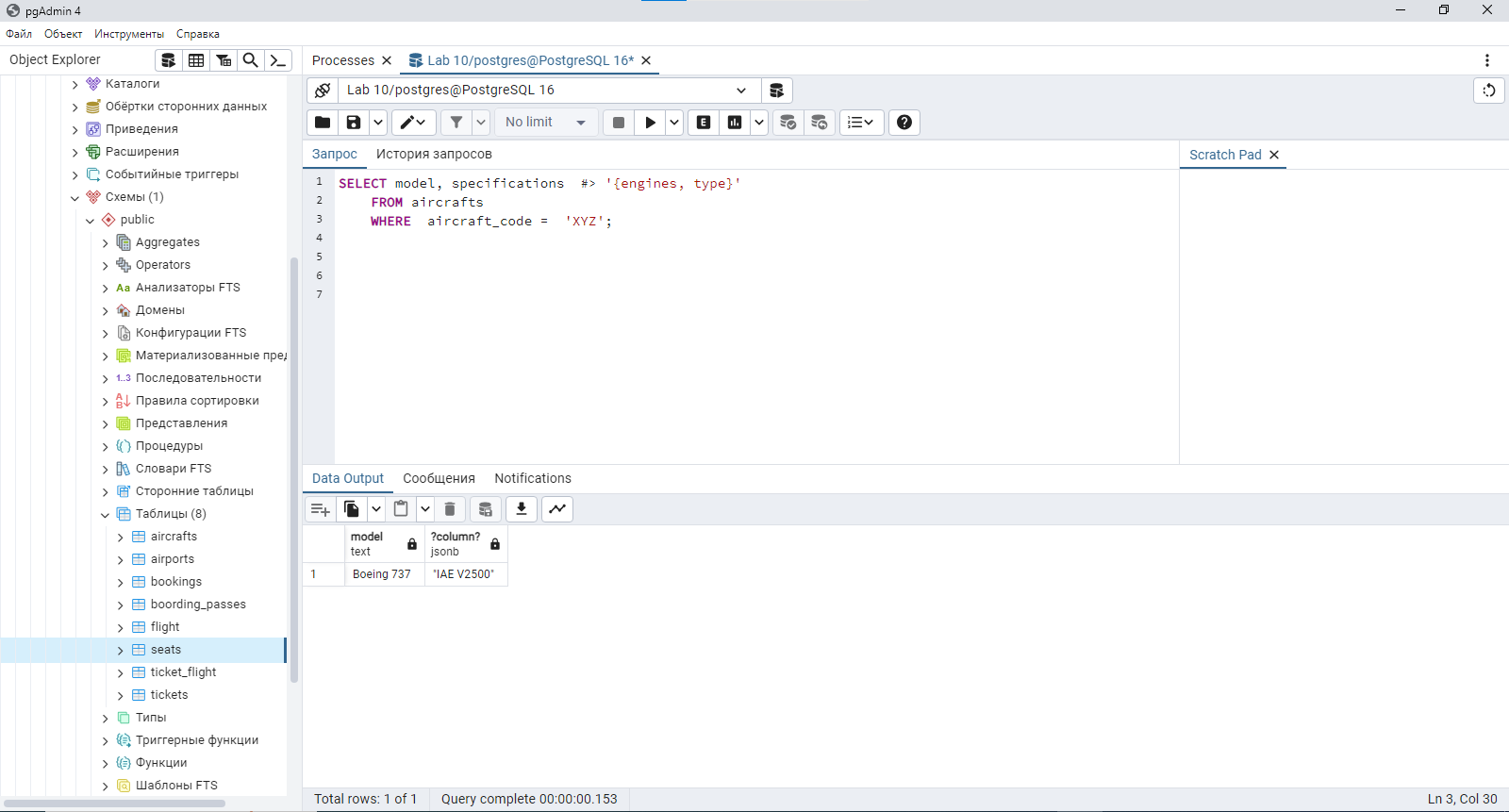
**Задание 5.**

Чтобы получить еще более детальные сведения, например, о типе двигателей, нужно учитывать, что созданный JSON-объект имеет сложную структуру: он содержит вложенный JSON-объект. Поэтому нужно использовать оператор **#>** для указания пути доступа к ключу второго уровня.

SELECT model, specifications #> '{engines, type}'

FROM aircrafts

WHERE aircraft\_code = ‘320’;

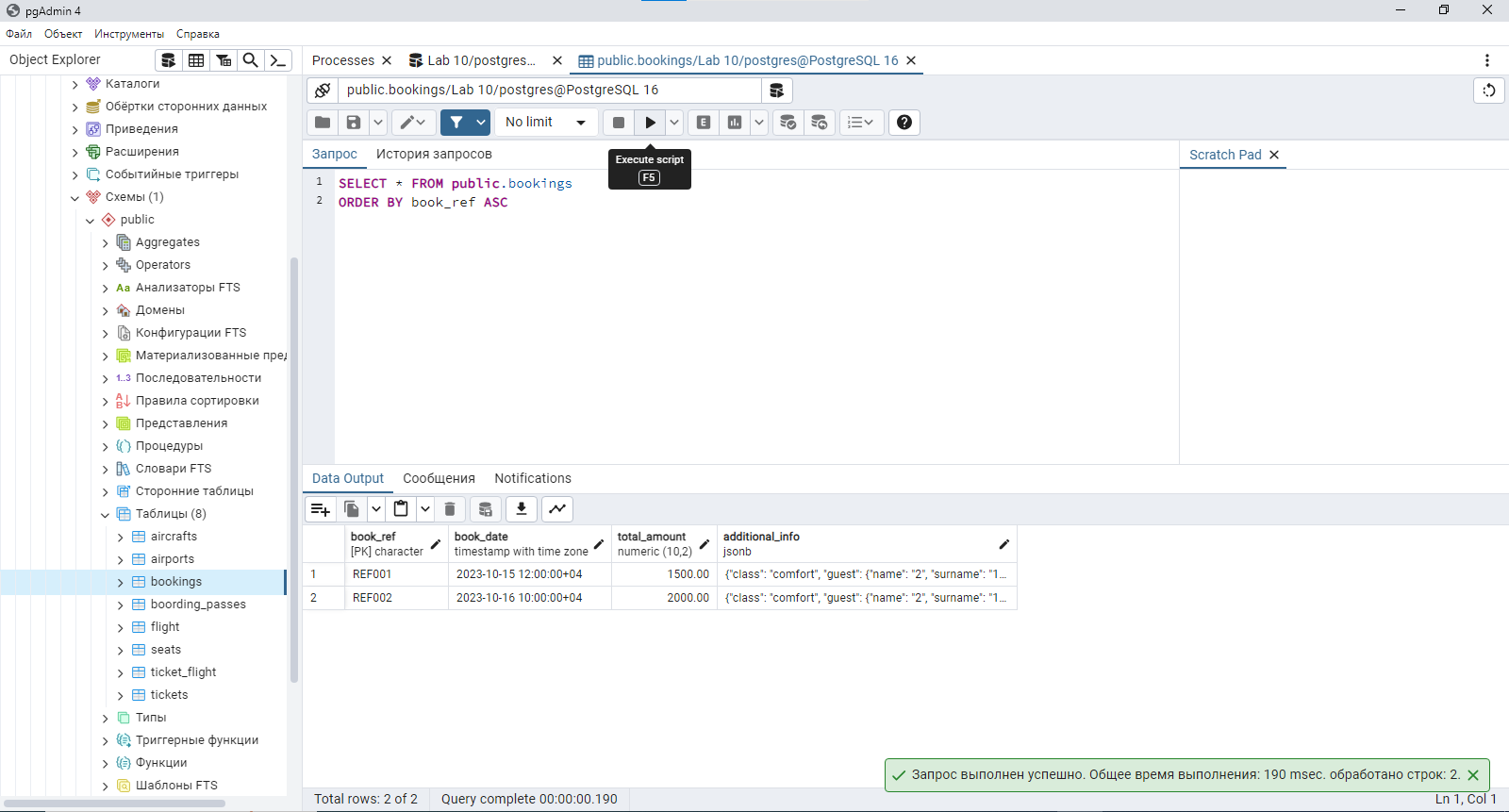
****

**Задание 6 (выполнить самостоятельно!).**

Подумайте, какие еще таблицы было бы целесообразно дополнить столбцами типа **json/jsonb**.

Вспомните, что, например, в таблице «**Билеты**» (**tickets**) уже есть столбец такого типа – contact\_data.

Выполните модификацию таблиц и измените в них одну-две строки для проверки правильности Ваших решений.

****

**UPDATE bookings**

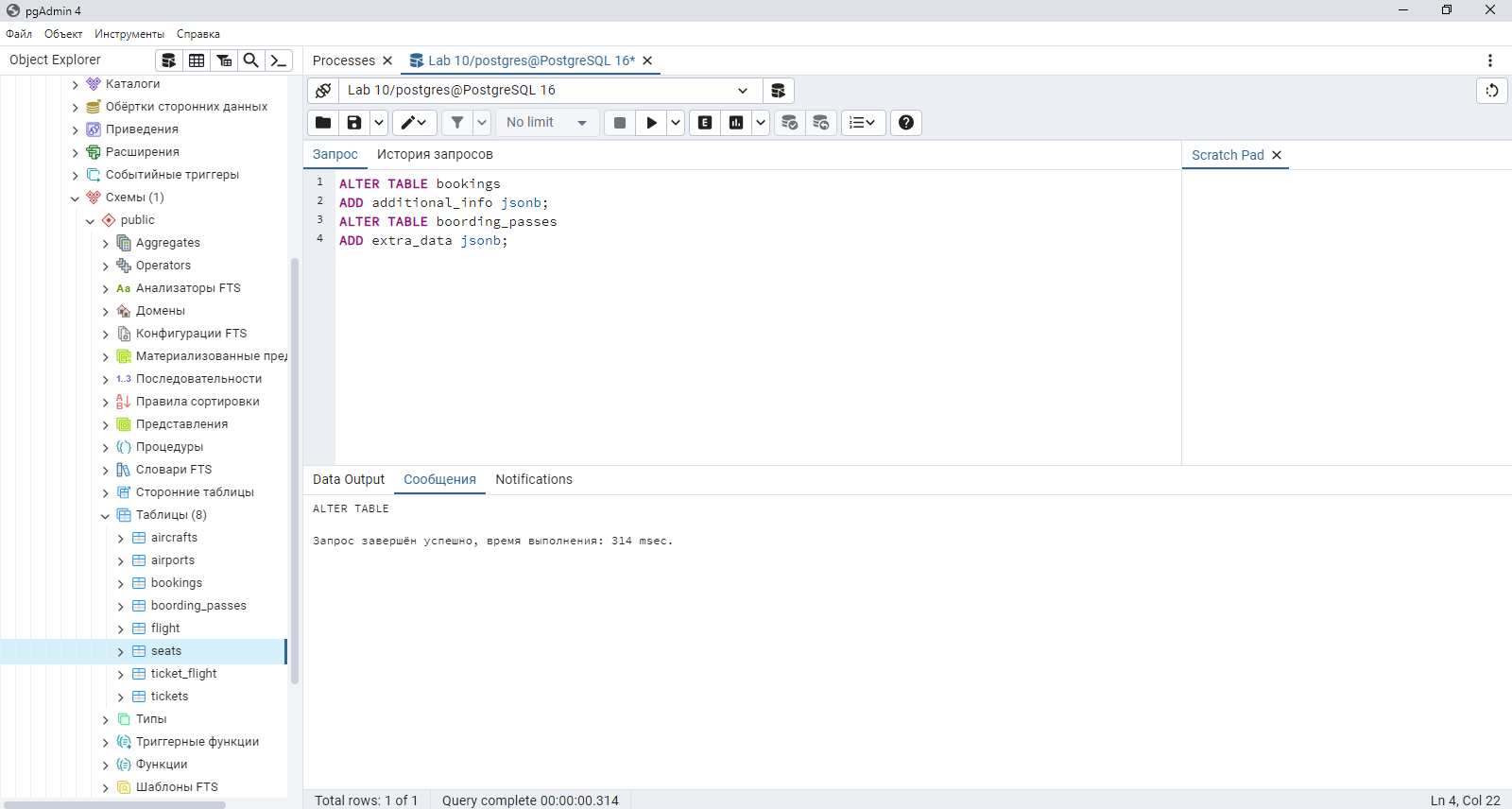
**SET additional\_info =**

**'{"class": "comfort",**

**"guest": {"name": "2",**

**"surname": "1"**

**}**

**}'::jsonb**

**Выводы:** в ходе выполнения лабораторной работы я получил практические навыки в использовании команд языка определения данных DDL в СУБД PostgreSQL.