SQL в свободно-распространяемых СУБД

НИЯУ МИФИ, ИФТЭБ, МНМЦ

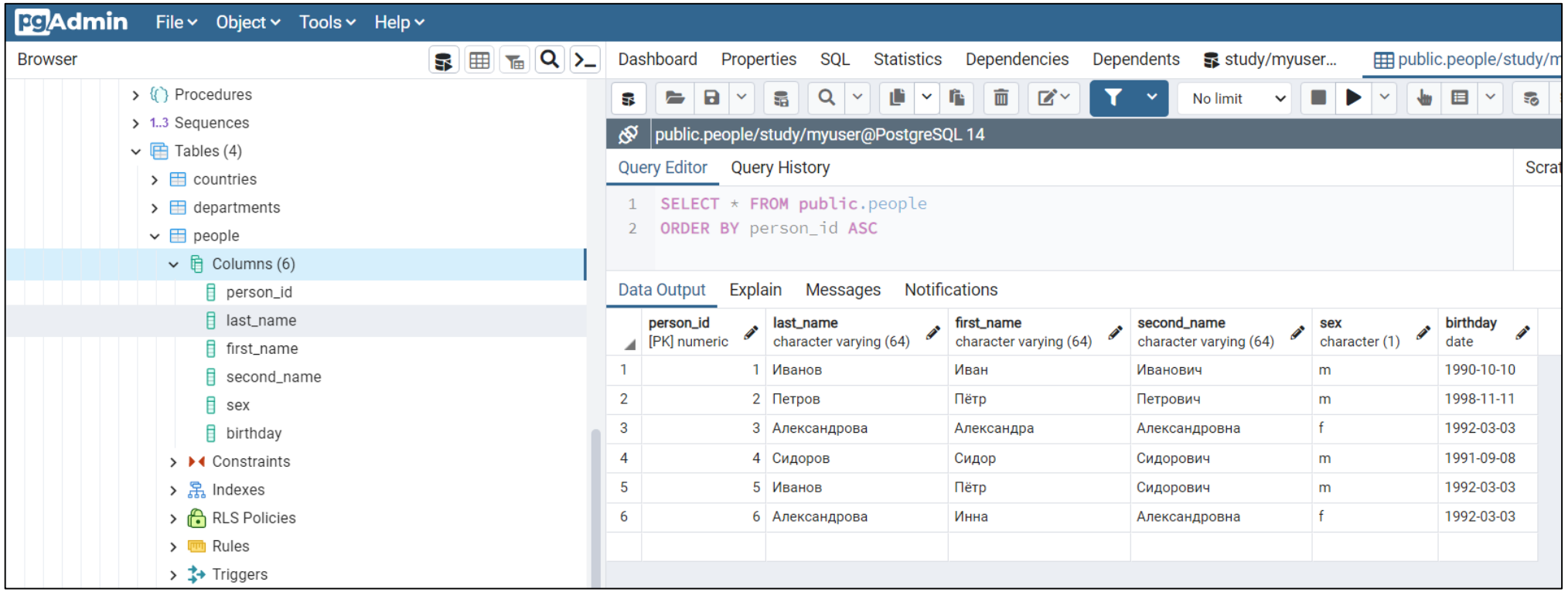
КУРС ЛЕКЦИЙ

В.Ю. РАДЫГИН., Д.Ю. КУПРИЯНОВ: ЛЕКЦИЯ 2

Таблица people

Примеры в данной лекции мы будем рассматривать на основе таблицы people из лекции 1:

PEOPLE



Часть 1. Простейшее наполнение таблиц

Для изучения принципов работы с таблицами средствами запросов языка SQL нам потребуется уметь их наполнять. Позднее на тему изменения данных в таблицах (добавление, модификация, удаление) будет отдельная лекция. Сейчас мы рассмотрим только самый простой пример добавления 1 строки.

Представим, что в рассмотренной ранее таблице не было заполнения. Тогда для добавления первой строки

1	Иванов	Иван	Иванович	m	1990-10-10

можно использовать команду INSERT в формате VALUES.

Добавление 1 строки

Query | Query History

```
1  INSERT INTO people
2      (id, last_name, first_name, second_name, sex, birthday)
3      VALUES
4      (1, 'Иванов', 'Иван', 'Иванович', 'm', '1990-10-10');
5
```

Data Output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 69 msec.
```

Часть 2. Построение простых запросов выборки

Всё, что мы изучали до этого относилось к DDL-компоненте языка SQL. Теперь рассмотрим компоненту DML.

Наиболее часто возникающая при работе с базой данных задача – это получение (или выборка) информации, хранящейся в таблицах. Так как мы изучаем реляционные базы данных, то результатом выборки информации из базы данных является множество строк.

В языке SQL для выборки информации из таблиц используется команда SELECT.

Команда SELECT

Команда SELECT в общем случае состоит из большого числа элементов. Поэтому мы будем с ней знакомиться поэтапно, наращивая сложность с каждым шагом.

Самый простой вариант выборки – это выборка всех строк со всеми столбцами из одной таблицы. В этом случае синтаксис команды SELECT следующий:

```
SELECT * FROM [схема.]имя_таблицы;
```

Пример 1: Выборка «всей» таблицы

```
SELECT * FROM people;
```

	person_id [PK] numeric	last_name character varying (64)	first_name character varying (64)	second_name character varying (64)	sex character (1)	birthday date
1	1	Иванов	Иван	Иванович	m	1990-10-10
2	2	Петров	Пётр	Петрович	m	1998-11-11
3	3	Александрова	Александра	Александровна	f	1992-03-03
4	4	Сидоров	Сидор	Сидорович	m	1991-09-08
5	5	Иванов	Пётр	Сидорович	m	1992-03-03
6	6	Александрова	Инна	Александровна	f	1992-03-03

Фильтр полей в выборке SELECT

Команда SELECT позволяет выбирать не только строки, состоящие из всех колонок таблицы, но и строки с произвольной комбинацией полей таблицы. При этом вместо символа «*» указываются названия нужных колонок, через запятую:

```
SELECT поле [, поле [, …]] FROM [схема.]имя_таблицы;
```

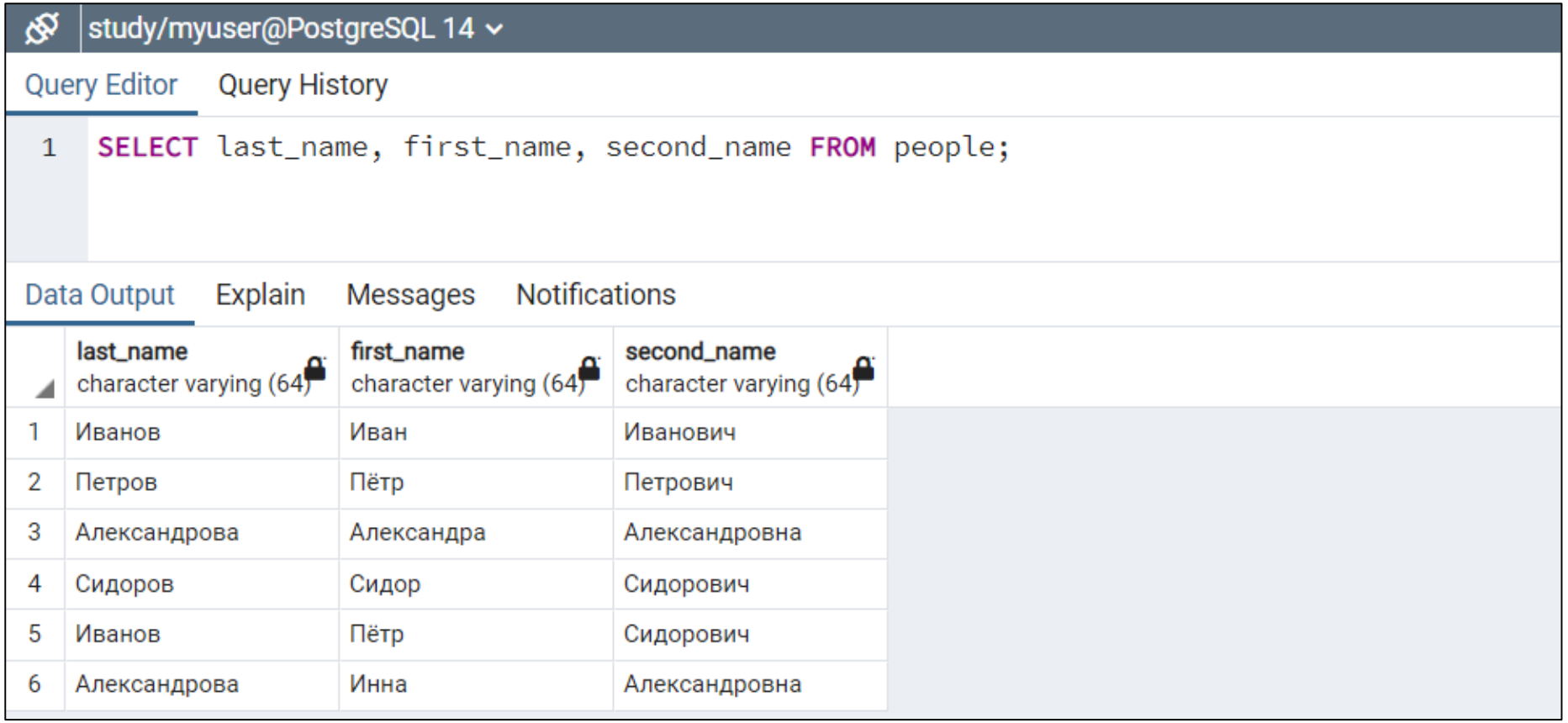
Причём одну и ту же колонку можно выбирать несколько раз.

Пример 2: Выборка ФИО из таблицы

Выберем из таблицы people только поля с фамилией, именем и отчеством:

```
SELECT last_name, first_name, second_name FROM people;
```

Пример 2: Выборка ФИО из таблицы

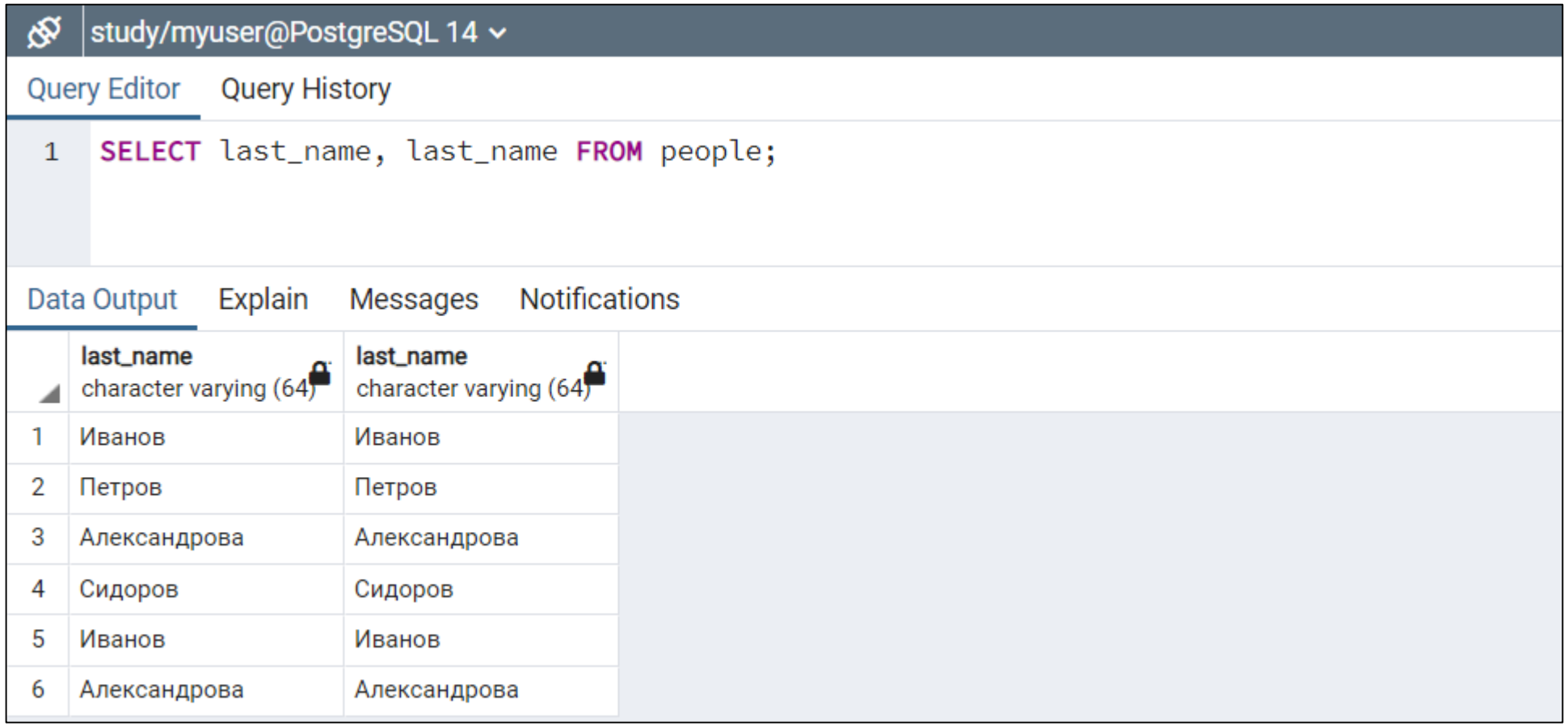


	last_name character varying (64)	first_name character varying (64)	second_name character varying (64)
1	Иванов	Иван	Иванович
2	Петров	Пётр	Петрович
3	Александрова	Александра	Александровна
4	Сидоров	Сидор	Сидорович
5	Иванов	Пётр	Сидорович
6	Александрова	Инна	Александровна

Пример 3: Выборка поля «фамилия» дважды

```
SELECT last_name, last_name FROM people;
```

Пример 3: Выборка поля «фамилия» дважды



Выбор с использованием выражений

Команда SELECT позволяет осуществлять обработку выбираемой информации путём замены колонок на выражения, состоящей из колонок, констант, операторов и функций. Причём выражение может даже не содержать колонок вовсе. Тем не менее, оно будет вычислено для каждой строки.

Подробно про функции и операторы будет рассказано дальше.

Пример 4: Выборка фамилии с инициалами

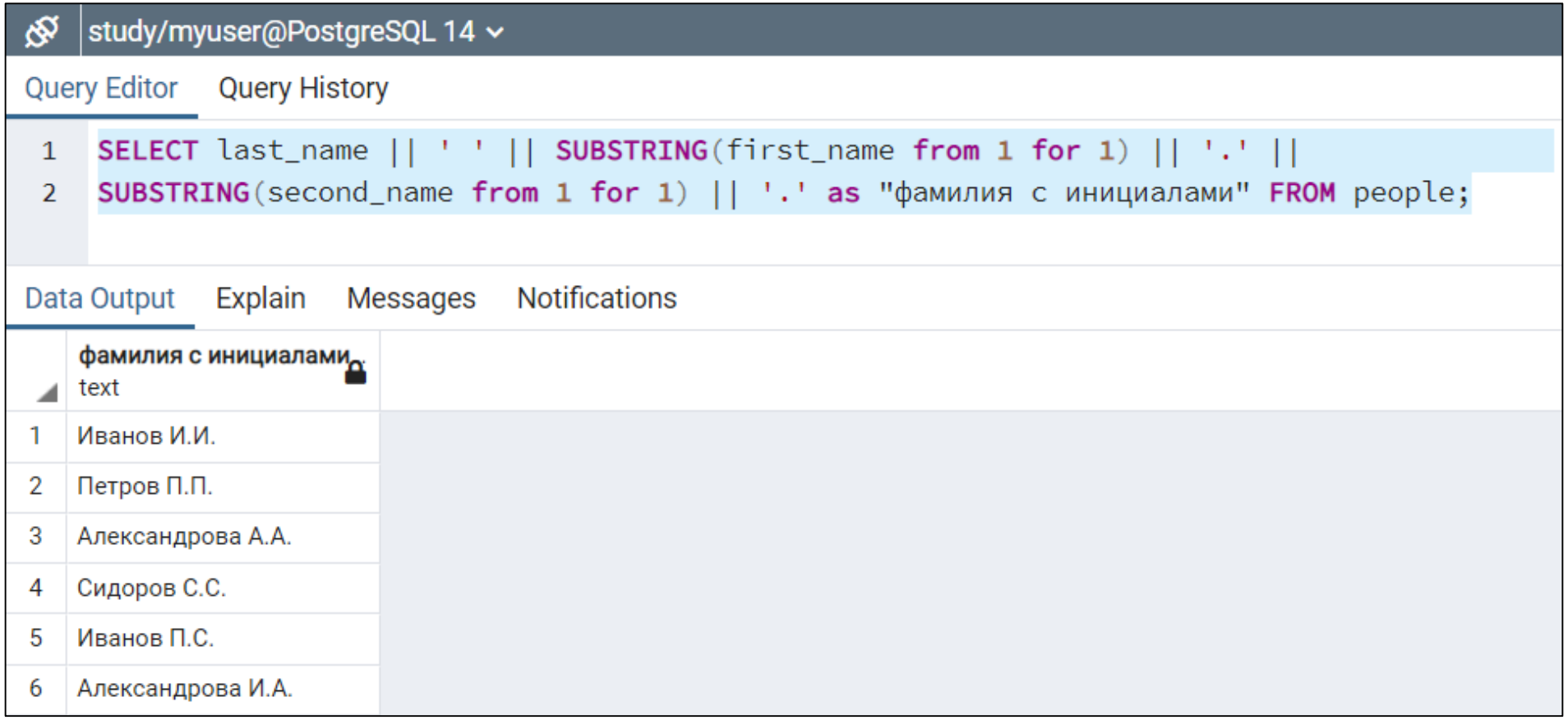
Выберем из таблицы people только фамилию с инициалами в виде одной колонки. Например, для Иванова Ивана Ивановича нужно будет выбрать «Иванов И.И.». При этом будем считать, что у нас всегда есть фамилия, имя и отчество.

Функция substring вырезает подстроку из строки.

Оператор «||» склеивает строки.

```
SELECT last_name || ' ' ||
       SUBSTRING(first_name from 1 for 1) || '.' ||
       SUBSTRING(second_name from 1 for 1) || '.'
         as "фамилия с инциалами"
FROM people;
```

Пример 4: Выборка фамилии с инициалами



Пример 4: Можно заменить from и for на запятые

study/myuser@PostgreSQL 14

Query Editor | Query History

```
1  SELECT last_name || ' ' || substring(first_name, 1, 1) || '.' ||
2  substring(second_name, 1, 1) || '.' as "фамилия с инициалами" FROM people;
```

Data Output | Explain | Messages | Notifications

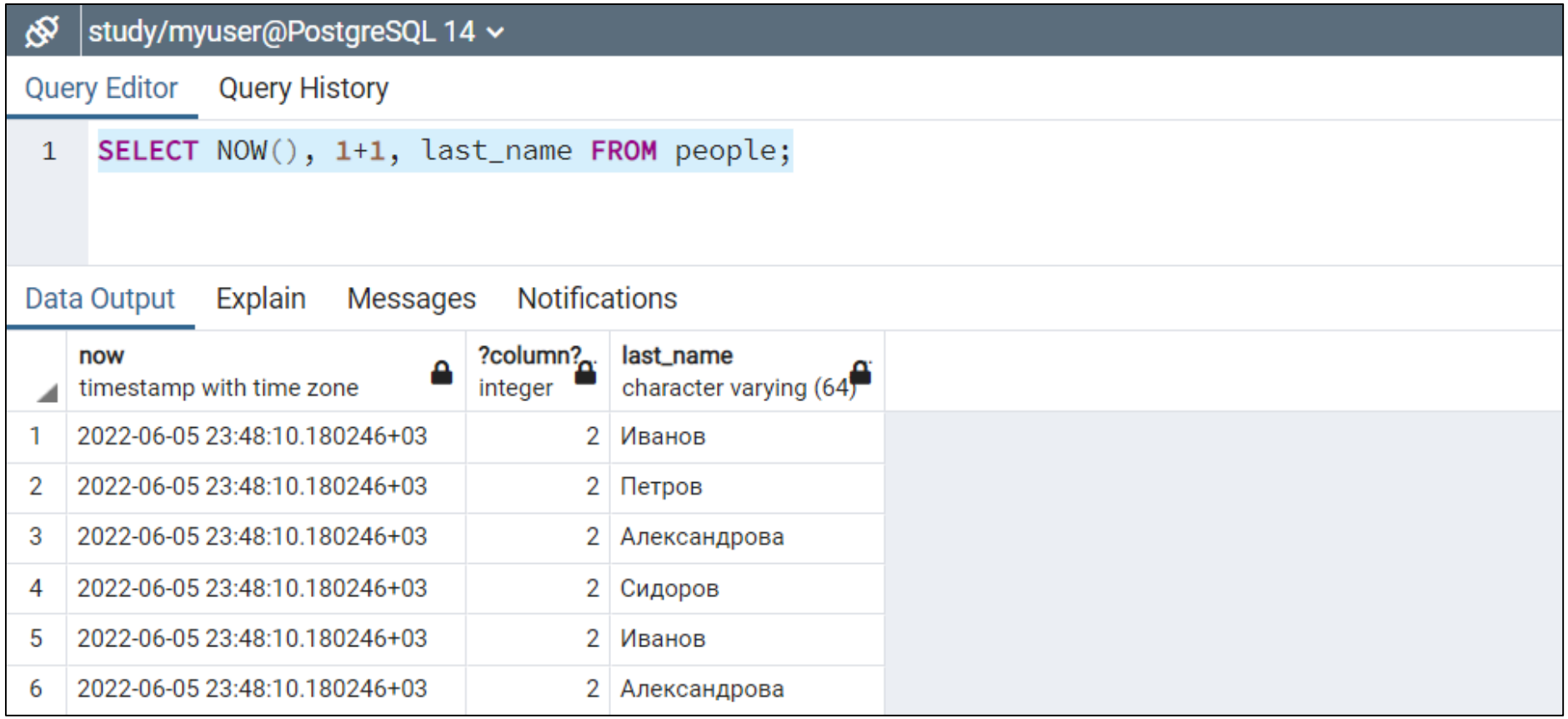
	фамилия с инициалами text
1	Иванов И.И.
2	Петров П.П.
3	Александрова А.А.
4	Сидоров С.С.
5	Иванов П.С.
6	Александрова И.А.

Пример 5: Использование выражений без значений полей таблицы

Выражения могут не содержать колонок таблиц:

```
SELECT NOW(), 1+1, last_name FROM people;
```

Пример 5: Использование выражений без значений полей таблицы



	now timestamp with time zone	?column? integer	last_name character varying (64)
1	2022-06-05 23:48:10.180246+03	2	Иванов
2	2022-06-05 23:48:10.180246+03	2	Петров
3	2022-06-05 23:48:10.180246+03	2	Александрова
4	2022-06-05 23:48:10.180246+03	2	Сидоров
5	2022-06-05 23:48:10.180246+03	2	Иванов
6	2022-06-05 23:48:10.180246+03	2	Александрова

Псевдонимы

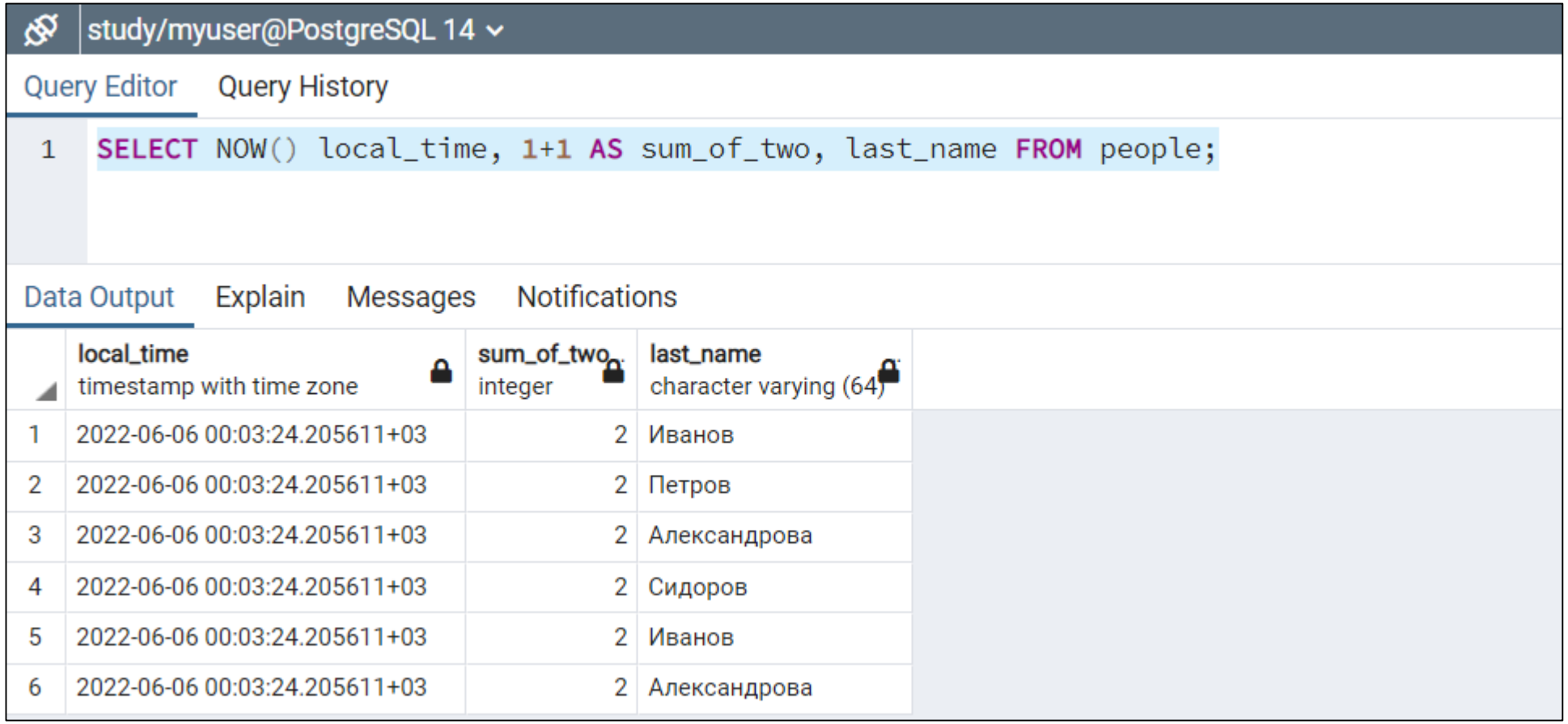
В примере 5 в одной из результирующих колонок, полученных в результате вычисления выражений, было неопределенное имя «?column?». Если имя колонки или вычисляемого выражения нужно заменить по каким-то причинам, то ему можно присвоить псевдоним (alias):

```
SELECT название_поля [AS] псевдоним FROM [схема.]имя_таблицы;
```

Пример 6: Использование псевдонима для вычисляемого атрибута (AS необязательно)

```
SELECT NOW() local_time, 1+1 as sum_of_two, last_name
FROM people;
```

Пример 6: Использование псевдонима для вычисляемого атрибута (AS необязательно)



Удаление повторений

Выбираемые строки могут повторяться. Иногда нужно учитывать повторения, а иногда хочется получить данные без дубликатов. Для решения задачи устранения повторяющихся строк применяется конструкция DISTINCT. DISTINCT может применяться как в режиме с символом «*», так и в режиме со списком атрибутов. В обоих случаях она размещается сразу после слова SELECT:

```
SELECT DISTINCT * FROM [схема.]имя_таблицы;
```

```
SELECT DISTINCT выражение[, выражение [, …]]
       FROM [схема.]имя_таблицы;
```

Пример 7: Удаление повторений среди всех строк таблицы

```
SELECT DISTINCT * FROM people;
```

Пример 7: Удаление повторений среди всех строк таблицы

study/myuser@PostgreSQL 14

Query Editor | Query History

```
1  SELECT DISTINCT * FROM people;
```

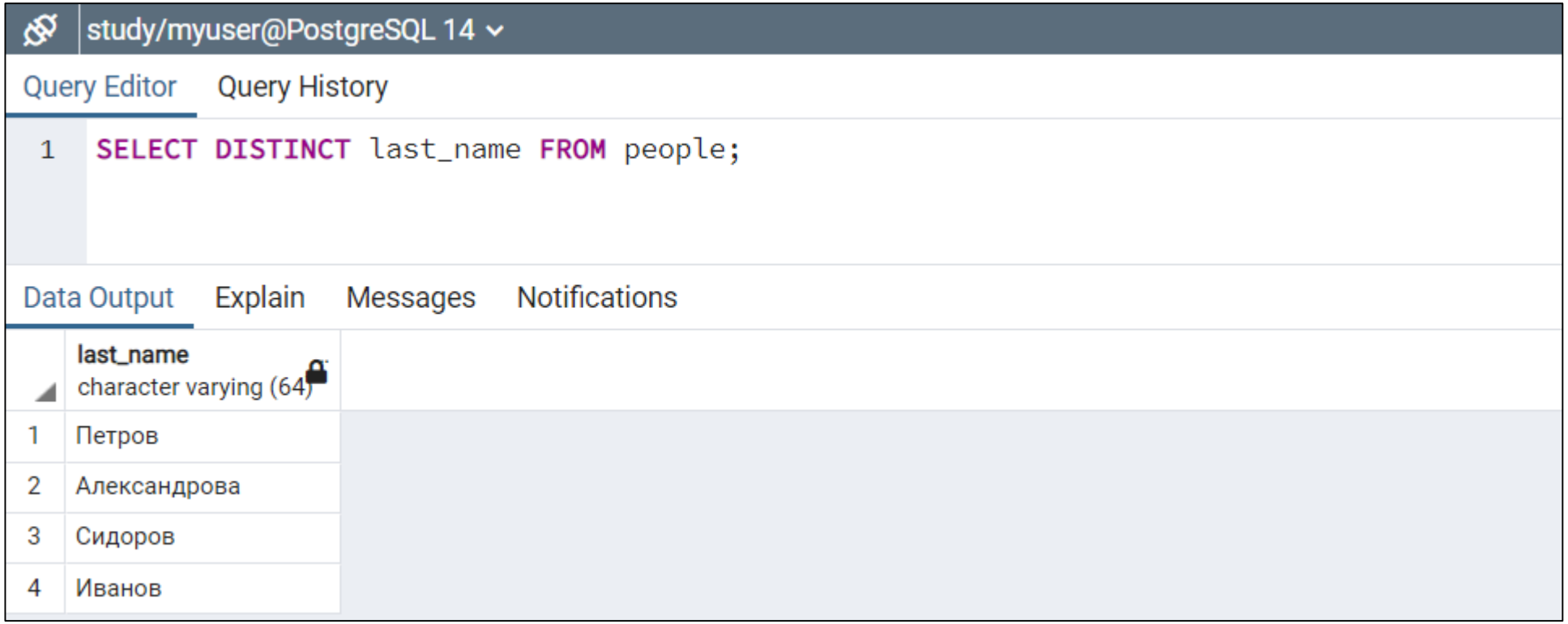
Data Output | Explain | Messages | Notifications

	person_id [PK] numeric	last_name character varying (64)	first_name character varying (64)	second_name character varying (64)	sex character (1)	birthday date
1	6	Александрова	Инна	Александровна	f	1992-03-03
2	2	Петров	Пётр	Петрович	m	1998-11-11
3	5	Иванов	Пётр	Сидорович	m	1992-03-03
4	4	Сидоров	Сидор	Сидорович	m	1991-09-08
5	3	Александрова	Александра	Александровна	f	1992-03-03
6	1	Иванов	Иван	Иванович	m	1990-10-10

Пример 8: Удаление повторений среди фамилий людей

```
SELECT DISTINCT last_name FROM people;
```

Пример 8: Удаление повторений среди фамилий людей



Формат вывода дат и времени

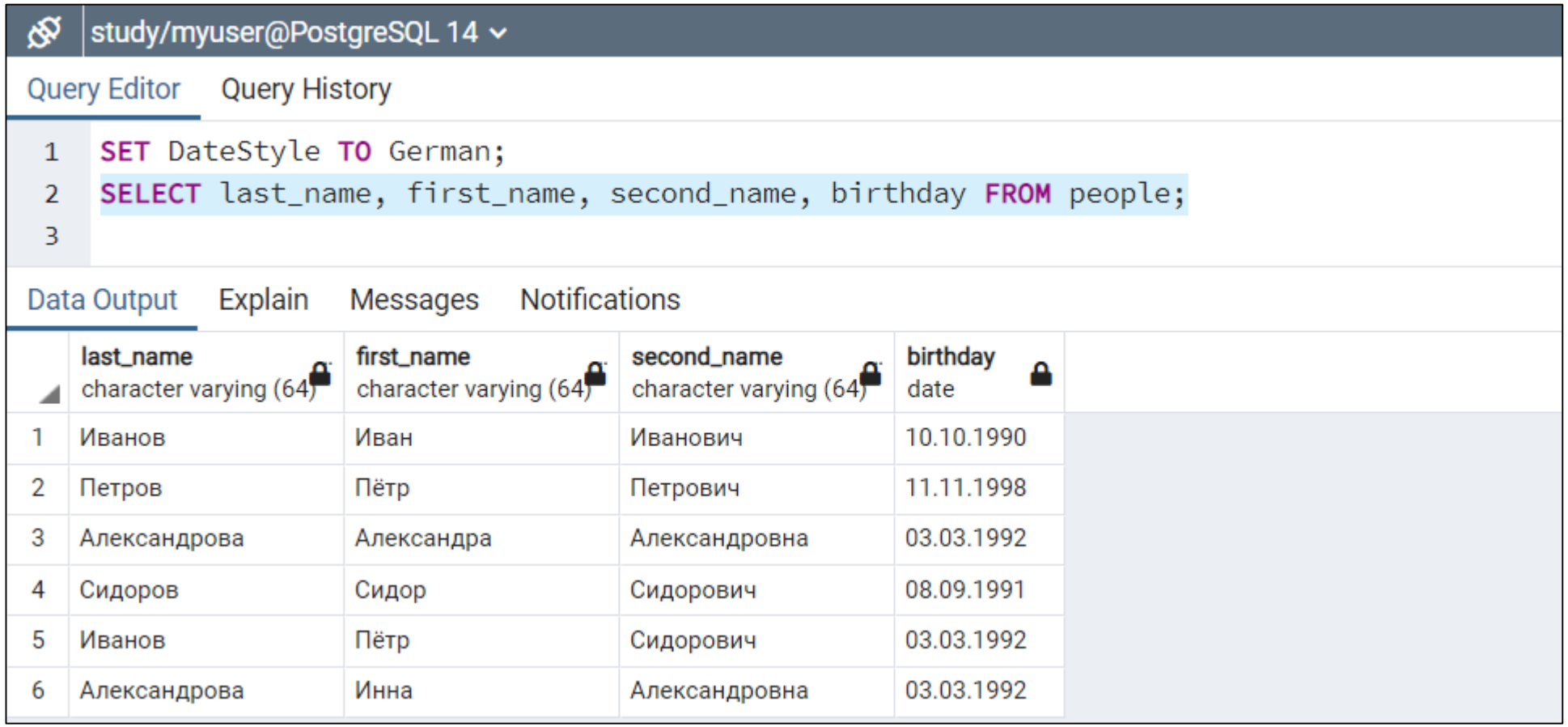
В примере 7 можно было заметить, что значения даты рождения выводятся в непривычном нам стиле (формат стандарта ISO). Исторически считается, что формат даты и времени в Postgres должен соответствовать стандарту ISO. Однако формат вывода можно изменить с помощью команды SET DateStyle и добиться привычного нам формата выбрав германский стиль.

```
SET DateStyle TO German;
```

Проверим теперь что произошло с датами с помощью следующей команды.

```
SELECT last_name, first_name, second_name, birthday FROM people;
```

Формат вывода дат и времени



	last_name character varying (64)	first_name character varying (64)	second_name character varying (64)	birthday date
1	Иванов	Иван	Иванович	10.10.1990
2	Петров	Пётр	Петрович	11.11.1998
3	Александрова	Александра	Александровна	03.03.1992
4	Сидоров	Сидор	Сидорович	08.09.1991
5	Иванов	Пётр	Сидорович	03.03.1992
6	Александрова	Инна	Александровна	03.03.1992

Усложним таблицу

Для следующих примеров нам понадобится более сложная таблица. Добавим в таблицу people новую колонку salary (заработная плата):

```
ALTER TABLE people ADD salary numeric(8, 2);
```

Затем заполним новую колонку.

Модифицированная таблица PEOPLE

PEOPLE

	person_id [PK] numeric	last_name character varying (64)	first_name character varying (64)	second_name character varying (64)	sex character (1)	birthday date	salary numeric (8,2)
1	1	Иванов	Иван	Иванович	m	10.10.1990	125000.00
2	2	Петров	Пётр	Петрович	m	11.11.1998	50000.00
3	3	Александрова	Александра	Александровна	f	03.03.1992	97500.25
4	4	Сидоров	Сидор	Сидорович	m	08.09.1991	180000.00
5	5	Иванов	Пётр	Сидорович	m	03.03.1992	50000.00
6	6	Александрова	Инна	Александровна	f	03.03.1992	77000.00

Ограничения множества строк

Обычно не каждая задача требует выборки всех строк из таблицы. Гораздо чаще необходимо выбрать лишь часть строк, в соответствии с определенным условием. Для ограничения множества выбираемых строк (фильтрации) предусмотрена конструкция WHERE:

```
SELECT [DISTINCT] * | выражение[, выражение [, …]]
        FROM [схема.]имя_таблицы WHERE предикат;
```

Здесь под конструкцией «предикат» подразумевается некоторое выражение, состоящее из полей, операторов и функций, результат которого для каждой строки может принимать только булевы значения: истина или ложь.

Операторы сравнения

Оператор	Описание	Примеры
<	Меньше	SELECT * FROM people WHERE salary < 125000;
>	Больше	SELECT * FROM people WHERE salary > 125000;
<=	Меньше или равно	SELECT * FROM people WHERE salary <= 125000;
>=	Больше или равно	SELECT * FROM people WHERE salary >= 125000;
=	Равно	SELECT * FROM people WHERE birthday = '10.10.1990';
!= <>	Не равно	SELECT * FROM people WHERE sex != 'f'; SELECT * FROM people WHERE first_name <> 'Иван';

Операторы сравнения

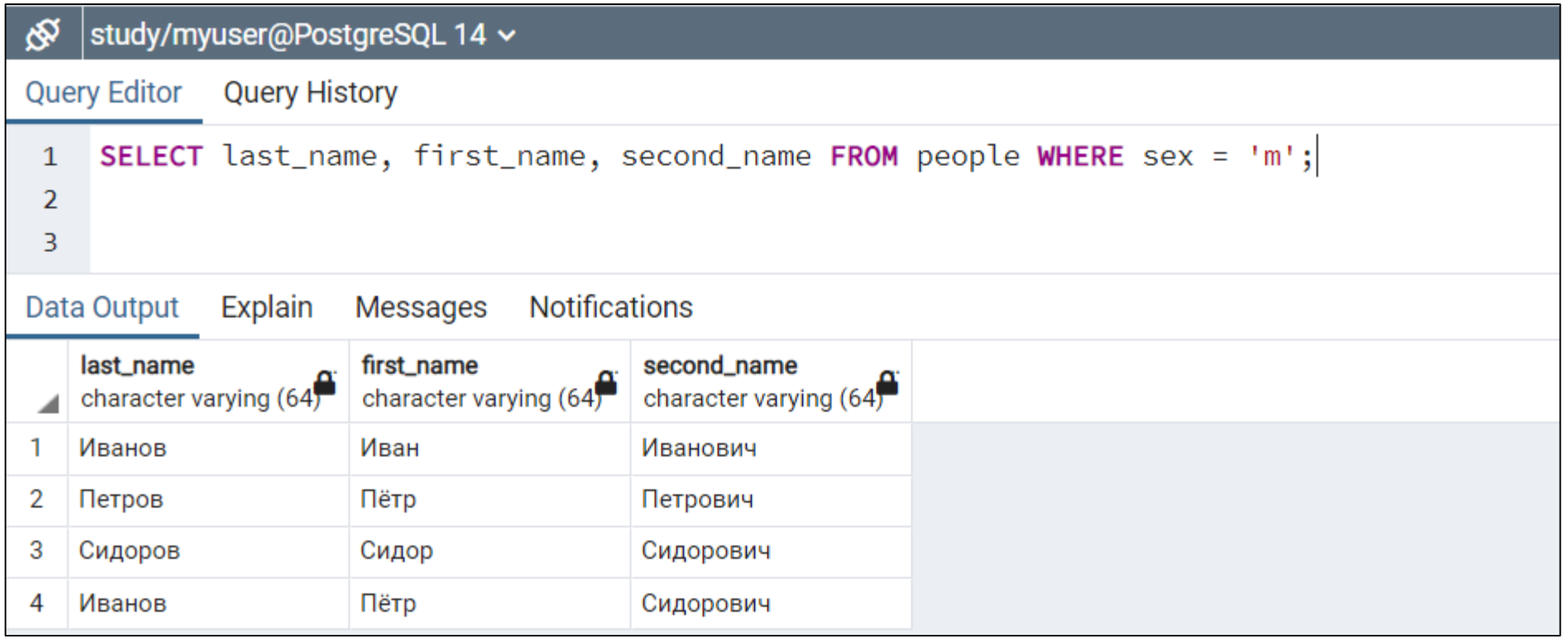
Оператор	Описание	Примеры
BETWEEN … AND	Попадает в диапазон	SELECT * FROM people WHERE salary BETWEEN 50000 AND 125000;
IN (список)	Попадает во множество	SELECT * FROM people WHERE salary IN (50000, 77000);
NOT IN (список)	Не попадает во множество	SELECT * FROM people WHERE salary NOT IN (50000, 77000);
IS NULL	Пусто	SELECT * FROM people WHERE second_name IS NULL;
IS NOT NULL	Не пусто	SELECT * FROM people WHERE second_name IS NOT NULL;

Пример 9: Только мужчины

Выберем фамилии, имена и отчества всех мужчин из таблицы people:

```
SELECT last_name, first_name, second_name FROM people WHERE sex = 'm';
```

Пример 9: Только мужчины



Логические операторы

Оператор	Описание	Примеры
AND	Логическое «И». Истинно, если истинны оба операнда.	SELECT * FROM people WHERE salary = 50000 AND birthday = '11.11.1998';
OR	Логическое «ИЛИ». Истинно, если хотя бы один из операндов истинен.	SELECT * FROM people WHERE salary = 50000 OR salary = 77000;
NOT	Логическое отрицание. Истинно, если операнд ложен.	SELECT * FROM people WHERE NOT salary = 50000;

Пример 10: Только мужчины с большим доходом

Выберем фамилии всех мужчин из таблицы people, у которых доход больше 100 000:

```
SELECT * FROM people WHERE sex = 'm' AND salary > 100000;
```

Пример 10: Только мужчины с большим доходом

study/myuser@PostgreSQL 14

Query Editor | Query History

```
1  SELECT * FROM people WHERE sex = 'm' AND salary > 100000;
2
3
```

Data Output | Explain | Messages | Notifications

	person_id [PK] numeric	last_name character varying (64)	first_name character varying (64)	second_name character varying (64)	sex character (1)	birthday date	salary numeric (8,2)
1	1	Иванов	Иван	Иванович	m	10.10.1990	125000.00
2	4	Сидоров	Сидор	Сидорович	m	08.09.1991	180000.00

Сравнение по шаблону

Язык SQL предусматривает возможность сравнения текстового значения с заданным шаблоном. Таким образом, можно описывать целое множество вариантов текстовых (символьных) строк при помощи единой конструкции – шаблона.

Сравнение по шаблону выполняется при помощи оператора LIKE. Шаблон задаётся при помощи обычных символов и двух спецсимволов: «%» и «_». Обычные символы при сравнении соответствуют самим себе.

Символ «%» трактуется как любое множество любых символов от 0 до бесконечности. Символ «_» трактуется как любой один символ.

Например, шаблону 'к_т' соответствуют строки «кот», «кит», но не соответствует строка «крот». Шаблону 'к%т' соответствуют строки «кот», «кит», «крот», «кт», но не соответствует строка «кета».

Пример 11: Все, у кого имя начинается на букву И

Выберем полную информацию о всех, у кого имя начинается на букву И:

```
SELECT * FROM people WHERE first_name LIKE 'И%';
```

Пример 11: Все, у кого имя начинается на букву И

study/myuser@PostgreSQL 14

Query Editor | Query History

```
1  SELECT * FROM people WHERE first_name LIKE 'И%';
2
3
```

Data Output | Explain | Messages | Notifications

	person_id [PK] numeric	last_name character varying (64)	first_name character varying (64)	second_name character varying (64)	sex character (1)	birthday date	salary numeric (8,2)
1	1	Иванов	Иван	Иванович	m	10.10.1990	125000.00
2	6	Александрова	Инна	Александровна	f	03.03.1992	77000.00

Пример 12: Все, кто родился в марте

Выберем полную информацию о всех, кто родился в марте:

```
SELECT * FROM people
WHERE to_char(birthday, 'DD.MM.YYYY') LIKE '%.03.%';
```

Пример 12: Все, кто родился в марте

study/myuser@PostgreSQL 14

Query Editor | Query History

```
1  SELECT * FROM people WHERE to_char(birthday, 'DD.MM.YYYY') LIKE '%.03.%';
2
3
```

Data Output | Explain | Messages | Notifications

	person_id [PK] numeric	last_name character varying (64)	first_name character varying (64)	second_name character varying (64)	sex character (1)	birthday date	salary numeric (8,2)
1	3	Александрова	Александра	Александровна	f	03.03.1992	97500.25
2	5	Иванов	Пётр	Сидорович	m	03.03.1992	50000.00
3	6	Александрова	Инна	Александровна	f	03.03.1992	77000.00

Особые символы

Что делать, если в шаблоне нужно непосредственно использовать символ «%» или символ «_»?

Вместо символа «%» можно использовать следующую конструкцию: «\%».

Вместо символа «_» можно использовать следующую конструкцию: «_».

Еще немного о шаблонах

К оператору LIKE можно применять отрицание NOT. Тогда получается оператор NOT LIKE, проверяющий, чтобы текстовая строка не удовлетворяла шаблону.

Вместо оператора LIKE можно использовать ключевое слово ILIKE, чтобы поиск был регистр-независимым с учётом текущей языковой среды. Этот оператор не описан в стандарте SQL (это расширение PostgreSQL).

Кроме того, в PostgreSQL есть оператор «~~», равнозначный LIKE, и «~~*», соответствующий ILIKE. Есть также два оператора «!~~» и «!~~*», представляющие NOT LIKE и NOT ILIKE, соответственно. Все эти операторы также относятся к особенностям PostgreSQL.

Приоритеты операторов (от большего к меньшему)

Порядок вычисления	Оператор / элемент	Очередность	Описание
1	.	слева-направо	разделитель имён таблицы и столбца
2	::	слева-направо	приведение типов в стиле PostgreSQL
3	[ ]	слева-направо	выбор элемента массива
4	+ -	справа-налево	унарный плюс, унарный минус
5	^	слева-направо	возведение в степень
6	* / %	слева-направо	умножение, деление, остаток от деления
7	+ -	слева-направо	сложение, вычитание

Приоритеты операторов (от большего к меньшему)

Порядок вычисления	Оператор / элемент	Очередность	Описание
8	(любой другой оператор)	слева-направо	все другие встроенные и пользовательские операторы
9	BETWEEN IN LIKE ILIKE SIMILAR		проверка на вхождение в диапазон, проверка на вхождение во множество, сравнение строк, сравнение строк без учета регистра, сравнение строк на основе SQL-регулярных выражений
10	< > = <= >= <>		операторы сравнения

Приоритеты операторов (от большего к меньшему)

Порядок вычисления	Оператор / элемент	Очередность	Описание
11	IS IS NULL IS NOT NULL		сравнение с TRUE или FALSE (IS FALSE), сравнение с NULL, проверка на не пустоту (не NULL)
12	NOT	справа-налево	логическое отрицание
13	AND	слева-направо	логическая конъюнкция
14	OR	слева-направо	логическая дизъюнкция

Сортировка

Записи в таблице хранятся, грубо говоря, в произвольном порядке. Выбирать же их часто нужно в строго определённом порядке. Для сортировки выбираемых строк используется конструкция ORDER BY:

```
SELECT [DISTINCT] * | выражение[, выражение [, …]]
       FROM [схема.]имя_таблицы WHERE предикат
       ORDER BY поле|выражение [тип сортировки]
         [, поле|выражение [тип сортировки] [, …]];
```

Тип сортировки может быть ASC – по возрастанию и DESC – по убыванию. Если тип сортировки не указан, то всегда сортируется по возрастанию.

Пример 13: Сортировка по убыванию заработной платы

Выберем все строки со всеми атрибутами, отсортировав их по убыванию заработной платы:

```
SELECT * FROM people ORDER BY salary DESC;
```

Пример 13: Сортировка по убыванию заработной платы

study/myuser@PostgreSQL 14

Query Editor | Query History

```
1  SELECT * FROM people ORDER BY salary DESC;
2
3
```

Data Output | Explain | Messages | Notifications

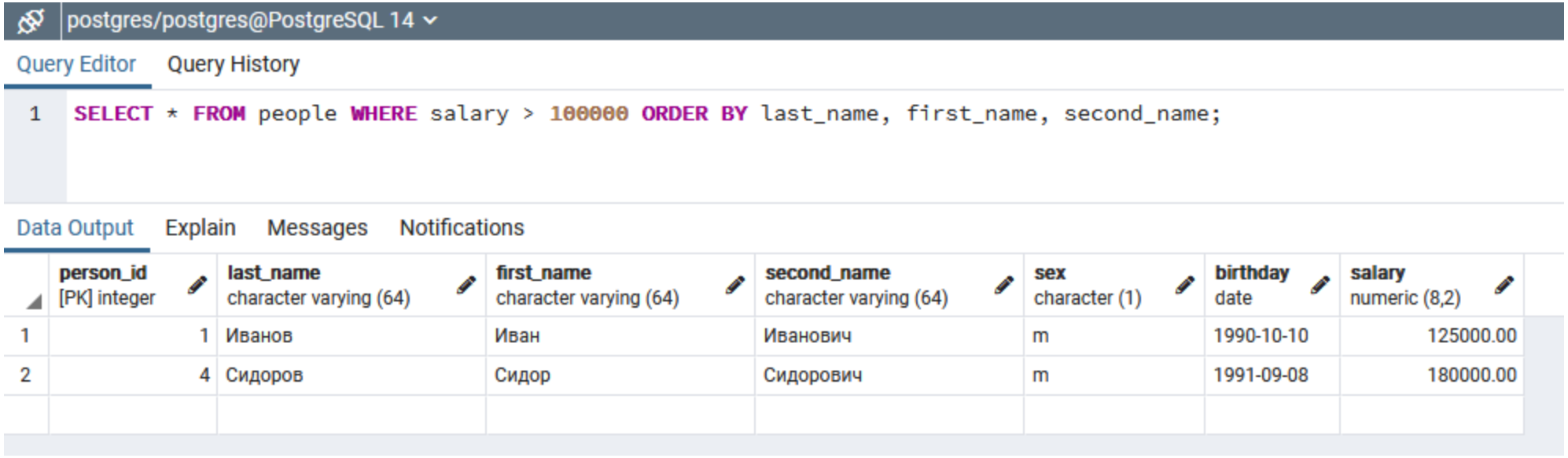
	person_id [PK] numeric	last_name character varying (64)	first_name character varying (64)	second_name character varying (64)	sex character (1)	birthday date	salary numeric (8,2)
1	4	Сидоров	Сидор	Сидорович	m	08.09.1991	180000.00
2	1	Иванов	Иван	Иванович	m	10.10.1990	125000.00
3	3	Александрова	Александра	Александровна	f	03.03.1992	97500.25
4	6	Александрова	Инна	Александровна	f	03.03.1992	77000.00
5	2	Петров	Пётр	Петрович	m	11.11.1998	50000.00
6	5	Иванов	Пётр	Сидорович	m	03.03.1992	50000.00

Пример 14: Сортировка по ФИО (комбинирование сортировки с условием)

Выберем все строки со всеми полями, сотрудников с заработной платой больше 100 тысяч, отсортировав их по возрастанию фамилии, имени и отчества:

```
SELECT * FROM people
WHERE salary > 100000
ORDER BY last_name, first_name, second_name;
```

Пример 14: Сортировка по ФИО (комбинирование сортировки с условием)



Функции

При построении выражений в SQL-запросах можно использовать функции. Различают два вида функций: однострочные функции (одного значения) и многострочные (агрегатные или групповые) функции.

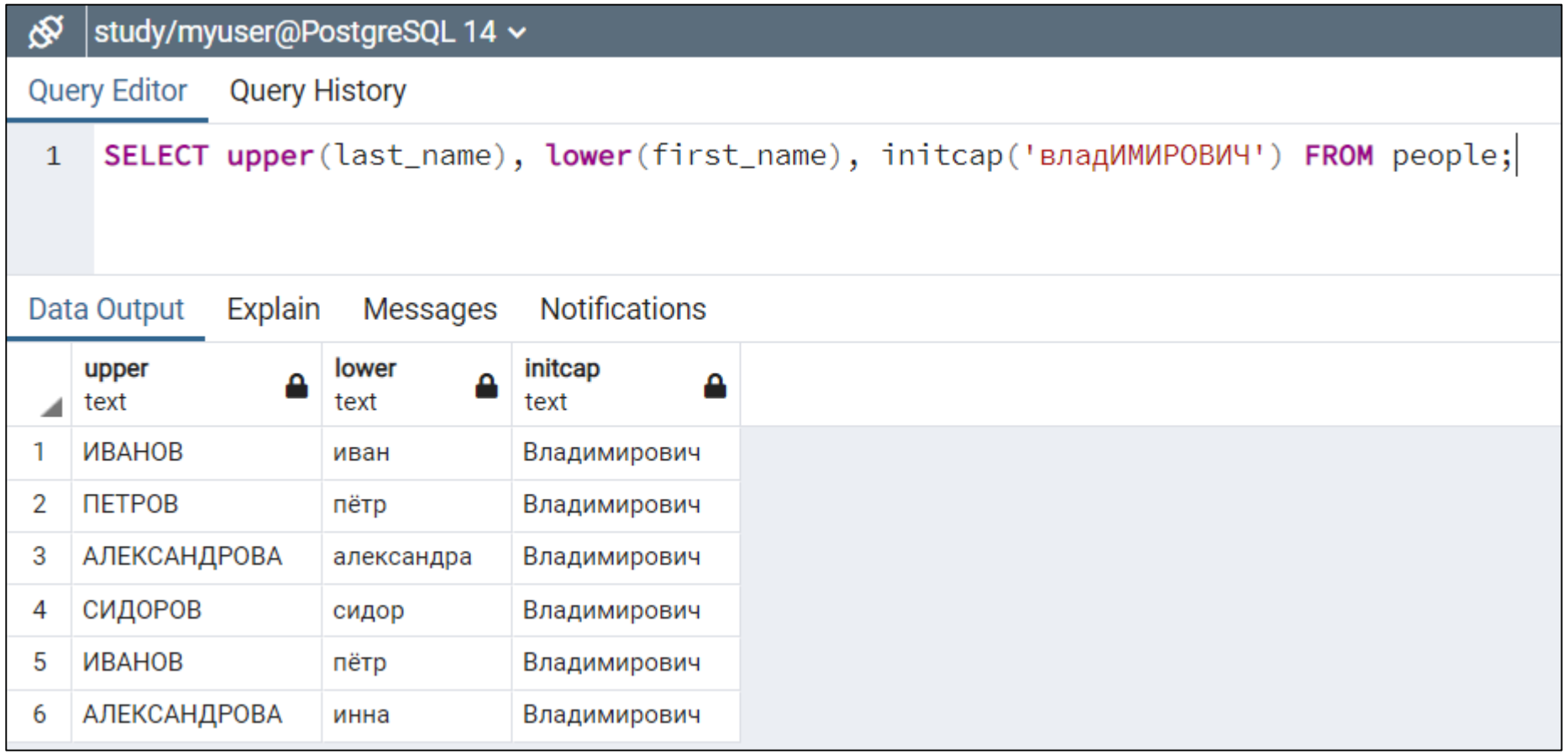
Однострочные функции на основе одной строки, через свои аргументы, получают одну новую строку – значение функции. Например, функция lower из одного текстового поля одной строки получает одно новое текстовое значение (все буквы исходного значения заменяются на эквивалентные в нижнем регистре).

Многострочные функции на основе нескольких строк, через свои аргументы, получают одну новую строку – значение функции. Например, функция avg берет по одному числовому полю из множества строк и находит среднее значение.

Пример 15: Однострочные функции

```
SELECT upper(last_name),
       lower(first_name),
       initcap('владИМИРОВИЧ')
FROM people;
```

Пример 15: Однострочные функции



Пример 16: Многострочные функции

```
SELECT avg(salary),
       sum(salary) / count(salary),
       string_agg(salary::varchar, ' + '),
       count(salary)
FROM people;
```

Пример 16: Многострочные функции

study/myuser@PostgreSQL 14

Query Editor | Query History

```
1  SELECT avg(salary), sum(salary) / count(salary),
2         string_agg(salary::varchar, ' + '), count(salary)
3  FROM people;
```

Data Output | Explain | Messages | Notifications

	avg numeric	?column? numeric	string_agg text	count bigint
1	96583.375000000000	96583.375000000000	125000.00 + 50000.00 + 97500.25 + 180000.00 + 50000.00 + 77000.00	6

Текстовые однострочные операторы и функции

Все однострочные операторы и функции можно поделить по типу обрабатываемых аргументов. Рассмотрим операторы и функции, обрабатывающие строки (текст). В дальнейшем будем называть их текстовыми.

Оператор	Описание	Примеры
||	text || text → text text || anynonarray → text anynonarray || text → text (соединяет две строки или строку и не строку)	SELECT last_name || ‘ ‘ || first_name FROM people; SELECT last_name || ' ' || birthday FROM people; SELECT person_id || ' ' || last_name FROM people;

Текстовые однострочные функции

Функция	Описание	Примеры
concat(t1, t2)	Соединяет текстовые представления всех аргументов, игнорируя NULL.	SELECT concat(person_id, last_name) FROM people;
length(t) или char_length(t)	Возвращает число символов в строке.	SELECT length(last_name), char_length(last_name) FROM people;
octet_length(t)	Возвращает число байт в строке.	SELECT octet_length(last_name) FROM people;
bit_length(t)	Возвращает число бит в строке.	SELECT bit_length(last_name) FROM people;

Текстовые однострочные функции

Функция	Описание	Примеры
lower(t)	Переводит символы строки в нижний регистр в соответствии с правилами локализации базы данных.	SELECT lower(last_name) FROM people;
upper(t)	Переводит символы строки в верхний регистр в соответствии с правилами локализации базы данных.	SELECT upper(last_name) FROM people;
initcap(t)	Переводит первый символ строки в верхний регистр, а остальные символы в нижний регистр в соответствии с правилами локализации базы данных.	SELECT initcap(last_name) FROM people;

Текстовые однострочные функции

Функция	Описание	Примеры
strpos(t, substring)	Возвращает расположение подстроки в строке.	SELECT strpos(last_name, 'a') FROM people;
substring(t [from start] [for count])	Извлекает из строки подстроку, начиная с позиции start (если она указана), длиной до count символов (если она указана). Параметры start и count могут опускаться, но не оба сразу.	SELECT substring(last_name from 1 for 1) FROM people; SELECT substring(last_name for 3) FROM people; SELECT substring(last_name from 3) FROM people;

Текстовые однострочные функции

Функция	Описание	Примеры
substring(t, start, [count])	Извлекает из строки подстроку, начиная с позиции start, длиной до count символов (если она указана).	SELECT substring(last_name, 1, 3) FROM people;
substr(t, start, [count])	Извлекает из строки подстроку, начиная с позиции start, длиной до count символов (если она указана).	SELECT substr(last_name, 1, 3) FROM people;
repeat(t, num)	Повторяет содержимое t указанное число (num) раз	SELECT repeat('abc', 4) FROM people;

Текстовые однострочные функции

Функция	Описание	Примеры
left(t, n)	Возвращает первые *n* символов в строке. Когда *n* меньше нуля, возвращаются все символы слева, кроме последних |*n*|.	SELECT left(last_name, 3) FROM people; SELECT left(last_name, -2) FROM people;
right(t, n)	Возвращает последние *n* символов в строке. Когда *n* меньше нуля, возвращаются все символы справа, кроме первых |*n*|.	SELECT right(last_name, 3) FROM people; SELECT right(last_name, -2) FROM people;
replace(t, from, to)	Заменяет все вхождения в st подстроки from подстрокой to.	SELECT replace(last_name, 'ов', 'офф') FROM people;

Текстовые однострочные функции

Функция	Описание	Примеры
reverse(t)	Переставляет символы в строке в обратном порядке.	SELECT reverse(last_name) FROM people;
lpad(t, len, fill)	Дополняет строку t слева до длины len символами fill (по умолчанию пробелами). Если длина строки уже больше заданной, она обрезается справа.	SELECT lpad(first_name, 20, '.') FROM people;
rpad(t, len, fill)	Дополняет строку t справа до длины len символами fill (по умолчанию пробелами). Если длина строки уже больше заданной, она обрезается справа.	SELECT rpad(first_name, 20, '.') FROM people;

Текстовые однострочные функции

Функция	Описание	Примеры
trim([leading | trailing | both] [*chrs*] FROM *t*)	Удаляет наибольшую подстроку, содержащую только символы chrs (по умолчанию пробелы), с начала, с конца или с обеих сторон (BOTH, по умолчанию) строки t.	SELECT trim(both from ' Сидорова ') FROM people; SELECT trim(trailing 'a' from last_name) FROM people;
ltrim(t [, chrs])	Удаляет наибольшую подстроку, содержащую только символы characters (по умолчанию пробелы), с начала строки.	SELECT ltrim(last_name, 'ИПА') FROM people;
rtrim(t [, chrs])	Удаляет наибольшую подстроку, содержащую только символы characters (по умолчанию пробелы), с конца строки.	SELECT rtrim(last_name, 'ва') from people;

Числовые функции

Функция	Описание	Примеры
round(num [, s])	Округляет число по правилам округления до n знаков после запятой. Если n не указано, то число округляется до целого.	SELECT round(42.4567, 2); SELECT round(42.4567);
trunc(num [, s])	Обрезает число до n знаков после запятой. Если n не указано, то число обрезается до целого.	SELECT trunc(42.4567, 2); SELECT trunc(42.4567);
mod(x, y)	Вычисляет остаток от деления числа x на число y.	SELECT mod(3,2);

Получение текущих даты и времени

Для получения текущих даты и времени предназначена функция now():

```
SELECT now();
```

Для получения текущий даты предназначена конструкция current_date:

```
SELECT current_date;
```

Полезные ссылки

1. https://www.postgresql.org/docs/current/index.html – документация PostgreSQL
2. https://www.postgresql.org/docs/current/sql-droptable.html – страница документации, посвященная удалению таблиц (DROP TABLE).
3. https://www.postgresql.org/docs/current/ddl-alter.html – страница документации, посвященная модификации таблиц (ALTER TABLE).
4. https://www.postgresql.org/docs/current/queries.html – страница документации, посвященная созданию запросов к таблицам (SELECT).
5. https://www.postgresql.org/docs/current/functions-string.html – страница документации, посвященная строковым (текстовым) операторам и функциям.
6. https://www.postgresql.org/docs/current/functions-math.html – страница документации, посвященная математическим (числовым) операторам и функциям.