МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра цифровой трансформации (ЦТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8

по дисциплине «Разработка баз данных»

Студент группы *ИНБО-12-23. Албахтин И.В.* _____
(подпись)

Ассистент *Брайловский А.В.* _____
(подпись)

Москва 2025 г.

ПРАКТИЧЕСКАЯ РАБОТА №7. СЛОЖНЫЕ ТИПЫ ДАННЫХ И МАСШТАБИРОВАНИЕ POSTGRESQL

Цель работы:

Целью данной практической работы является освоение методов работы со сложными и неструктурированными типами данных (JSONB, BYTEA) в PostgreSQL, а также получение практических навыков реализации горизонтального масштабирования базы данных с помощью декларативного секционирования.

По завершении работы студент должен уметь:

• Проектировать таблицы для хранения чувствительных данных (например, паролей) с использованием бинарного типа данных BYTEA и криптографических функций.

• Реализовывать хранение полуструктурированных данных (логов, событий) в формате JSONB.

• Выполнять выборку, фильтрацию и агрегацию данных из JSON-документов с использованием специализированных операторов (->>, @>, ?) и условий сравнения.

• Модифицировать структуру JSON-документов «на лету» без изменения схемы таблицы.

• Настраивать декларативное секционирование (Partitioning) таблиц по диапазонам для оптимизации производительности при работе с большими объемами данных.

Постановка задачи:

Для выполнения практической работы необходимо последовательно выполнить следующие шаги, адаптируя примеры из БД «Аптека» к вашей собственной базе данных.

Задание №1: хранение паролей (BYTEA)

1. Создать справочную таблицу ролей и таблицу системных пользователей (или модифицировать существующую).
2. В таблице пользователей предусмотреть поля для хранения «сырого» пароля (только для учебной демонстрации) и его хеша (тип BYTEA).
3. Сохранить хеши паролей пользователей, используя функцию хеширования digest.
4. Выполнить запрос, проверяющий соответствие введенного пароля сохраненному хешу. Одна проверка должна быть успешной, другая – нет (допустимо сделать это в одном запросе).

Задание №2: секционирование таблицы логов

1. Создать новую секционированную таблицу для хранения логов событий.
2. Использовать стратегию секционирования по диапазонам (RANGE) по полю даты.
3. Создать две секции для периодов:
• 2-е полугодие 2024 (с 2024-07-01 по 2025-01-01).
• 1-е полугодие 2025 (с 2025-01-01 по 2025-07-01).
4. Создать секцию по умолчанию, куда будут попадать все данные, выходящие за рамки указанных диапазонов (будущее время).

Задание №3: генерация данных

Написать скрипт на PL/pgSQL для генерации 20 000 записей в диапазоне дат, покрывающем созданные секции. Сгенерированные данные должны содержать JSON-структуру с различным набором полей для разных типов событий (например, чтобы логи «продажи» отличались по структуре от логов «входа»).

Вывести сгенерированные данные – показать запросом SELECT, что данные успешно созданы и физически распределились по разным таблицам- секциям.

Задание №4: анализ и поиск по JSONB

Написать три запроса:

1. Фильтрация по значению – найти записи, где числовое поле внутри JSON удовлетворяет математическому условию (например, > 800), используя операторы извлечения -> и ->>.
2. Поиск по вхождению – найти записи, содержащие точный фрагмент JSON-структуры (например, {"quantity": 5}), используя оператор @>.
3. Агрегация – посчитать сумму или среднее значение числового поля из JSON-документа.

Задание №5: модификация данных JSONB

Продемонстрировать изменение данных внутри JSON-поля:

1. Массовое добавление нового поля во все записи определенного типа.
2. Точечное изменение значения по ключу в конкретной записи.

ВЫПОЛНЕНИЕ ПРАКТИЧЕСКОЙ РАБОТЫ

Таблица 1. Данные таблицы system_roles

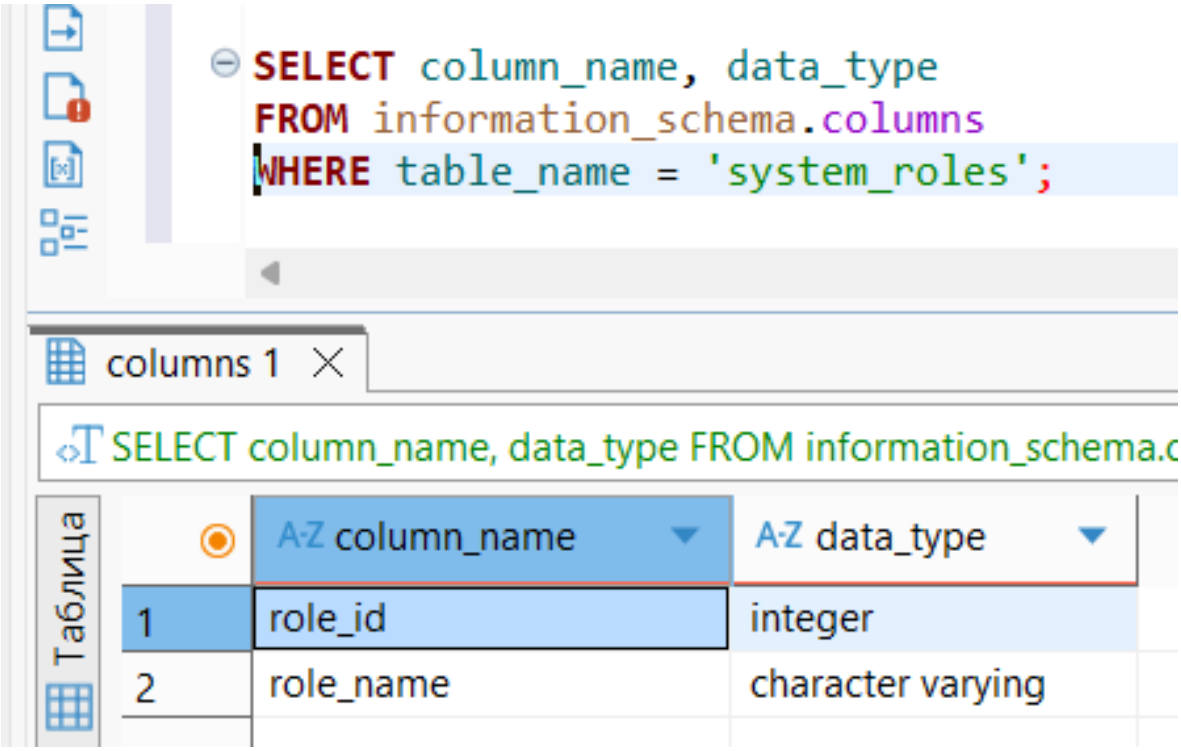


Таблица 2. Структура таблицы system_roles

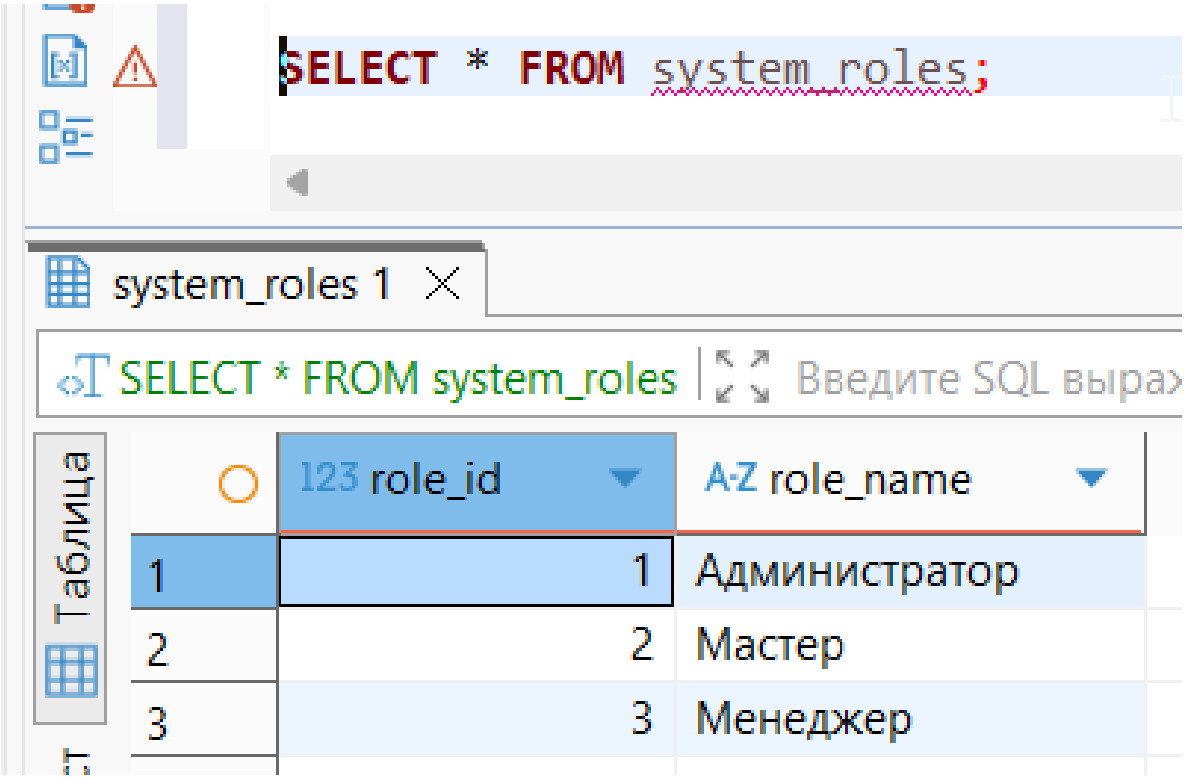
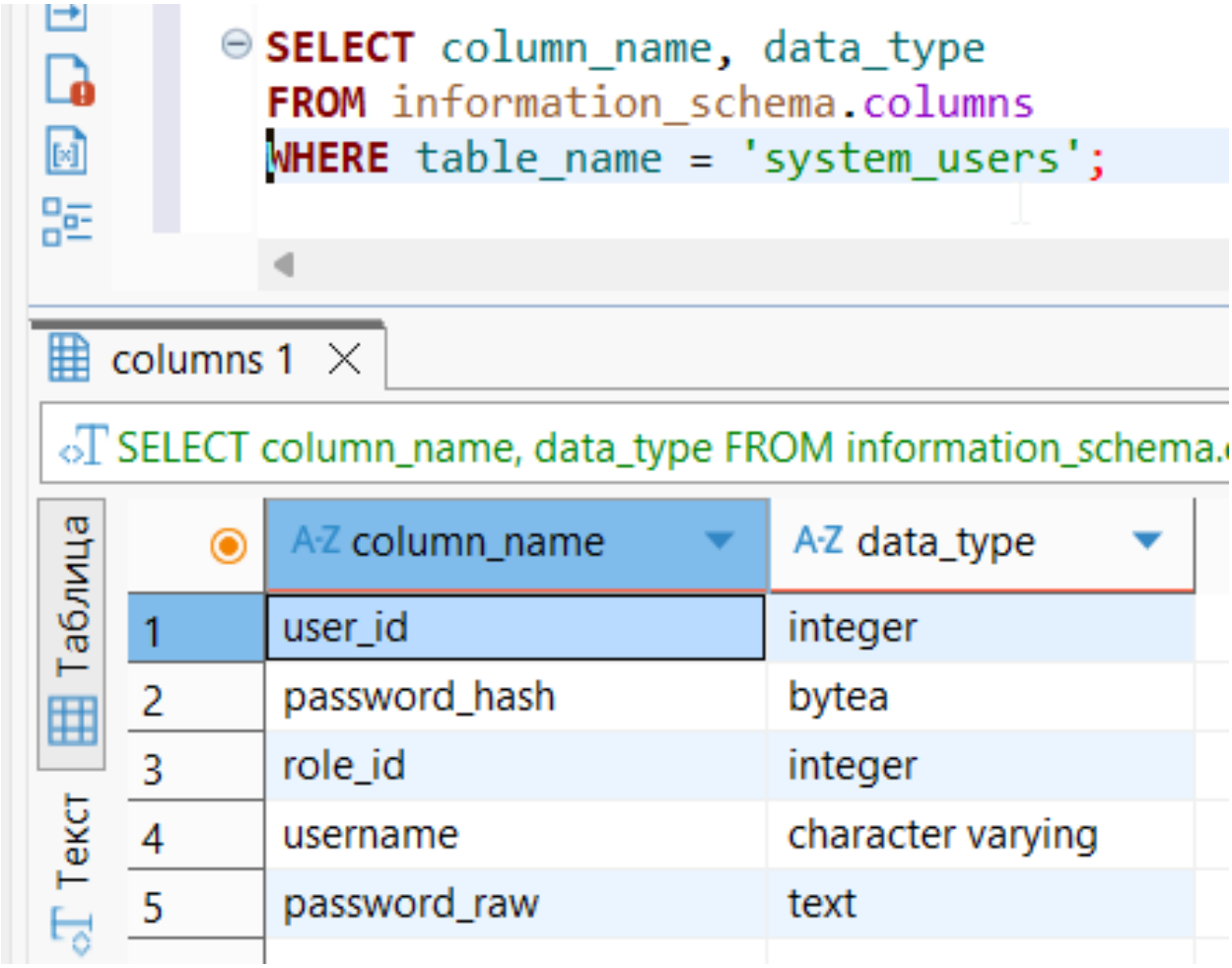
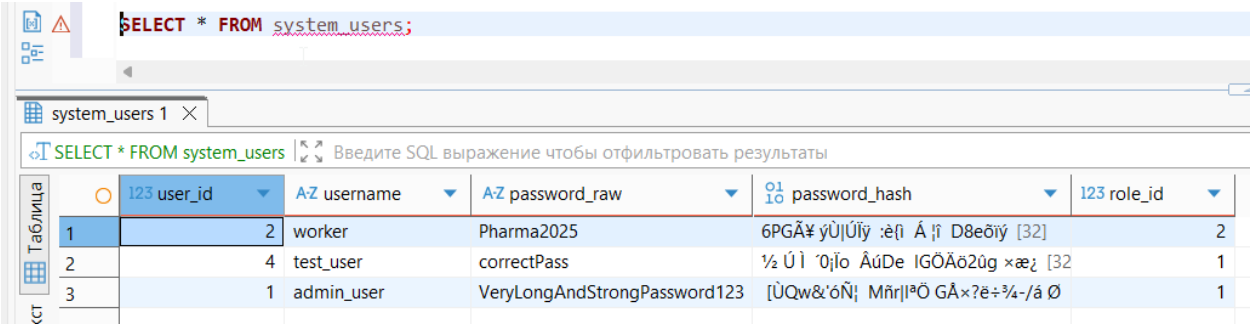


Таблица 3. Структура таблицы system_users



	column_name	data_type
1	user_id	integer
2	password_hash	bytea
3	role_id	integer
4	username	character varying
5	password_raw	text

Таблица 4. Данные таблицы system_users



	user_id	username	password_raw	password_hash	role_id
1	2	worker	Pharma2025	6PGÃ¥ ýÙ|ÚÏÿ :è{î Á ¦î D8eõïý [32]	2
2	4	test_user	correctPass	½ Ú Ì ´0¡Ïo ÂúDe IGÖÄö2úg ×æ¿ [32	1
3	1	admin_user	VeryLongAndStrongPassword123	[ÙQw&'óÑ¦ Mñr|ªÖ GÅ×?ë÷¾-/á Ø	1

Таблица 5. Структура таблицы change_log

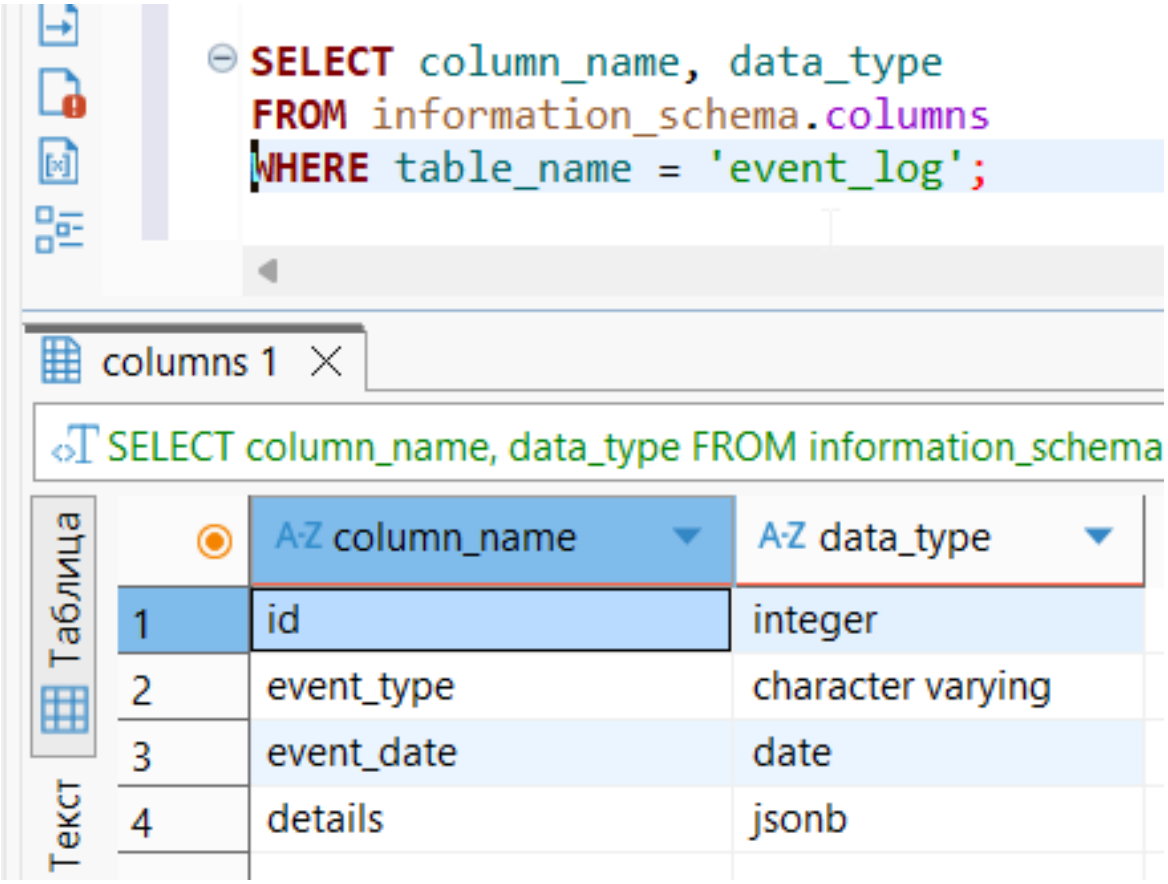


Таблица 6. Секции таблицы event_log

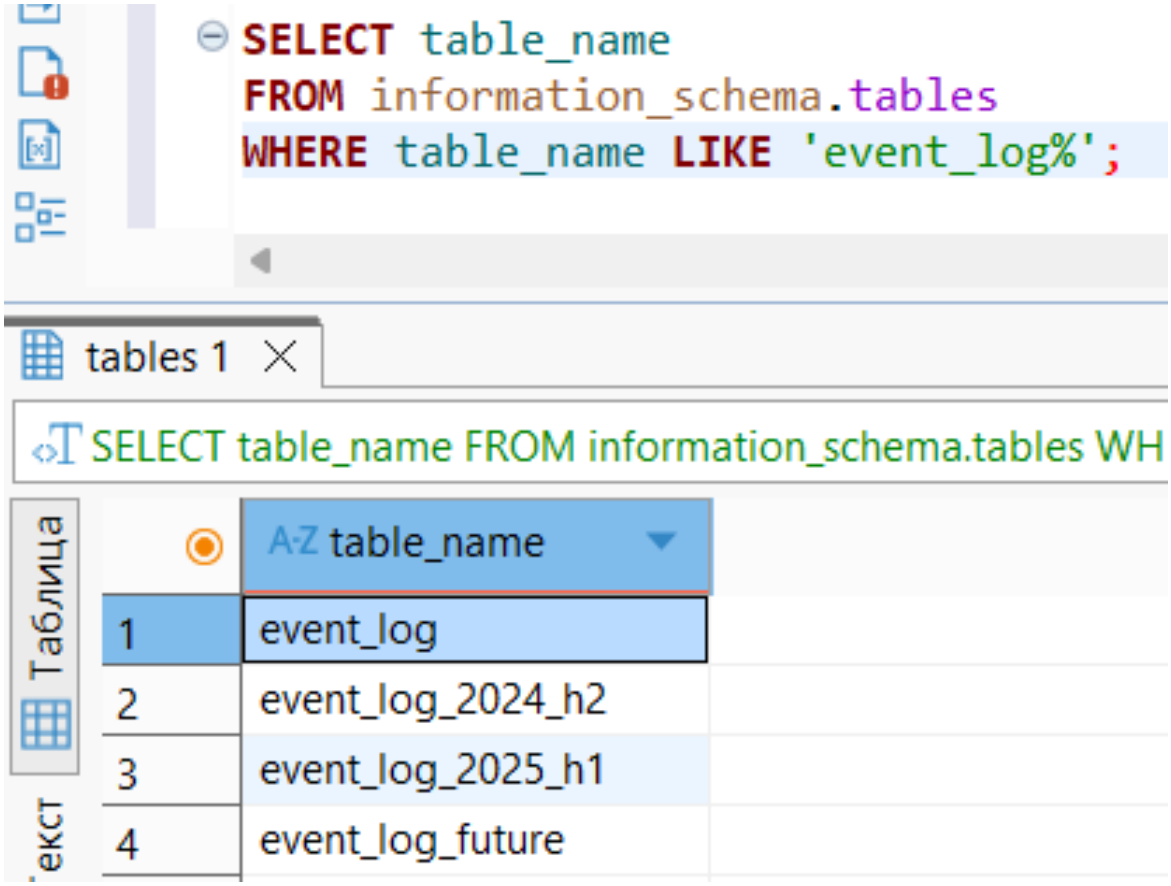
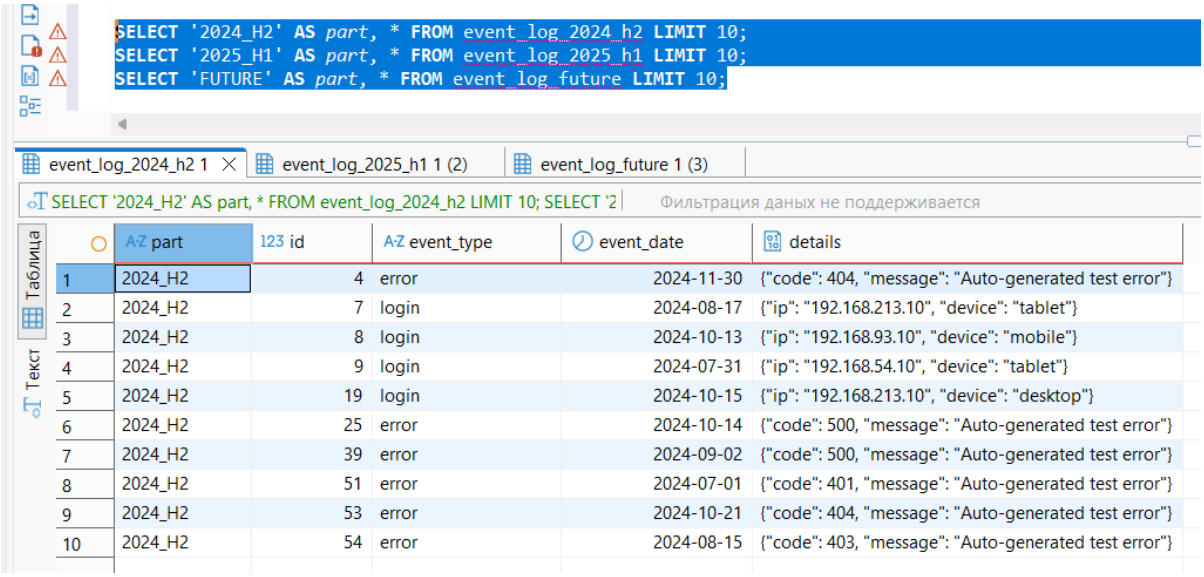


Таблица 7. Данные секции event_log_2024_h2



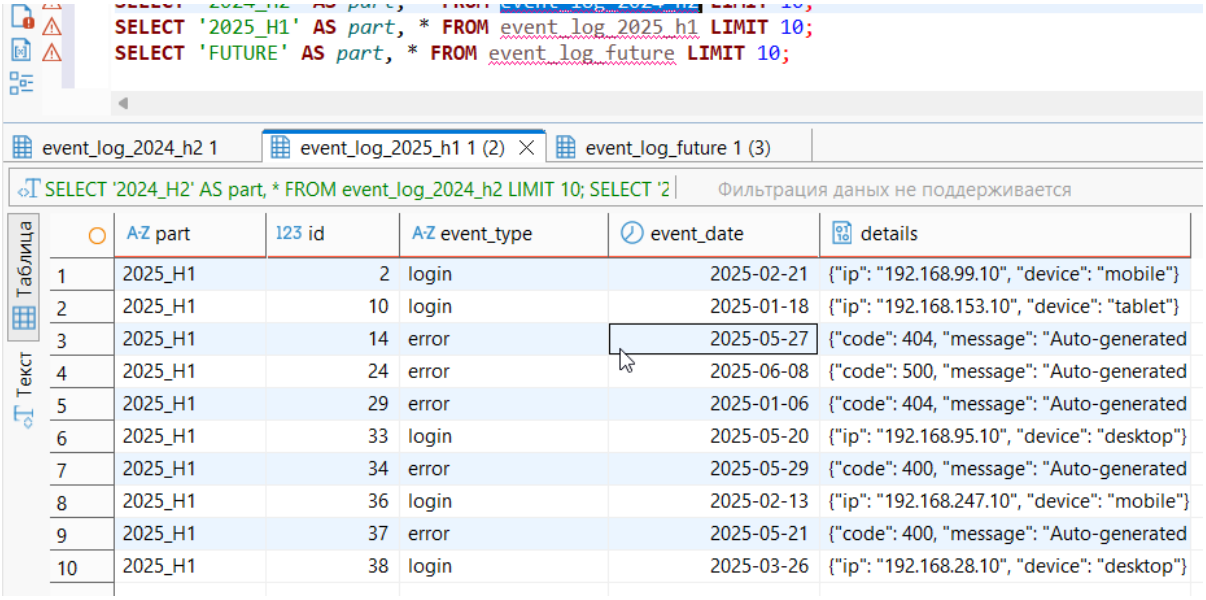
```
SELECT '2024_H2' AS part, * FROM event_log_2024_h2 LIMIT 10;
SELECT '2025_H1' AS part, * FROM event_log_2025_h1 LIMIT 10;
SELECT 'FUTURE' AS part, * FROM event_log_future LIMIT 10;
```

event_log_2024_h2 1 | event_log_2025_h1 1 (2) | event_log_future 1 (3)

SELECT '2024_H2' AS part, * FROM event_log_2024_h2 LIMIT 10; SELECT '2 — Фильтрация даных не поддерживается

	part	id	event_type	event_date	details
1	2024_H2	4	error	2024-11-30	{"code": 404, "message": "Auto-generated test error"}
2	2024_H2	7	login	2024-08-17	{"ip": "192.168.213.10", "device": "tablet"}
3	2024_H2	8	login	2024-10-13	{"ip": "192.168.93.10", "device": "mobile"}
4	2024_H2	9	login	2024-07-31	{"ip": "192.168.54.10", "device": "tablet"}
5	2024_H2	19	login	2024-10-15	{"ip": "192.168.213.10", "device": "desktop"}
6	2024_H2	25	error	2024-10-14	{"code": 500, "message": "Auto-generated test error"}
7	2024_H2	39	error	2024-09-02	{"code": 500, "message": "Auto-generated test error"}
8	2024_H2	51	error	2024-07-01	{"code": 401, "message": "Auto-generated test error"}
9	2024_H2	53	error	2024-10-21	{"code": 404, "message": "Auto-generated test error"}
10	2024_H2	54	error	2024-08-15	{"code": 403, "message": "Auto-generated test error"}

Таблица 8. Данные секции event_log_2025_h1



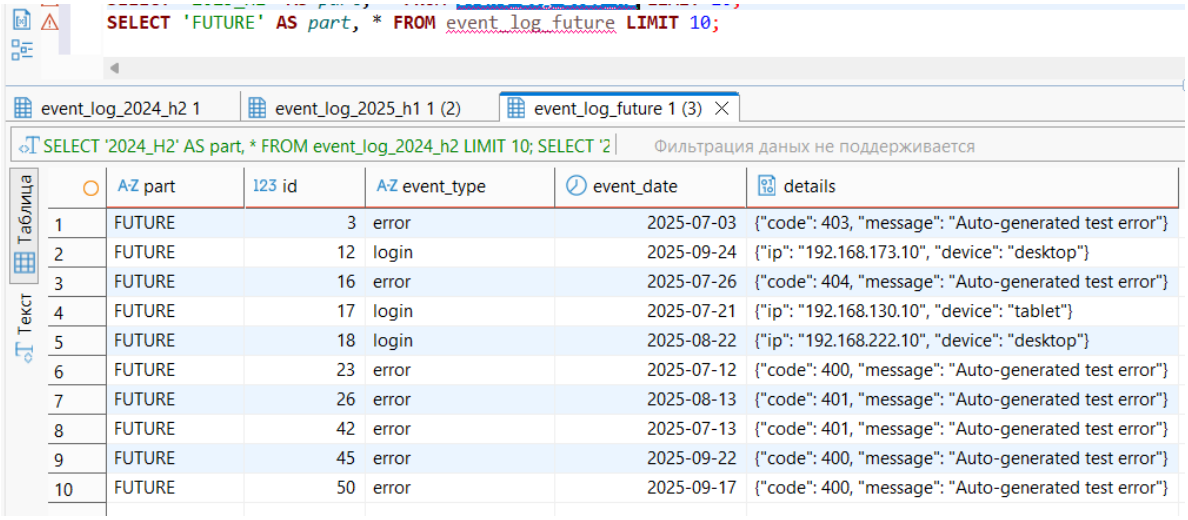
```
SELECT '2025_H1' AS part, * FROM event_log_2025_h1 LIMIT 10;
SELECT 'FUTURE' AS part, * FROM event_log_future LIMIT 10;
```

event_log_2024_h2 1 | event_log_2025_h1 1 (2) | event_log_future 1 (3)

SELECT '2024_H2' AS part, * FROM event_log_2024_h2 LIMIT 10; SELECT '2 — Фильтрация даных не поддерживается

	part	id	event_type	event_date	details
1	2025_H1	2	login	2025-02-21	{"ip": "192.168.99.10", "device": "mobile"}
2	2025_H1	10	login	2025-01-18	{"ip": "192.168.153.10", "device": "tablet"}
3	2025_H1	14	error	2025-05-27	{"code": 404, "message": "Auto-generated
4	2025_H1	24	error	2025-06-08	{"code": 500, "message": "Auto-generated
5	2025_H1	29	error	2025-01-06	{"code": 404, "message": "Auto-generated
6	2025_H1	33	login	2025-05-20	{"ip": "192.168.95.10", "device": "desktop"}
7	2025_H1	34	error	2025-05-29	{"code": 400, "message": "Auto-generated
8	2025_H1	36	login	2025-02-13	{"ip": "192.168.247.10", "device": "mobile"}
9	2025_H1	37	error	2025-05-21	{"code": 400, "message": "Auto-generated
10	2025_H1	38	login	2025-03-26	{"ip": "192.168.28.10", "device": "desktop"}

Таблица 9. Данные секции event_log_future



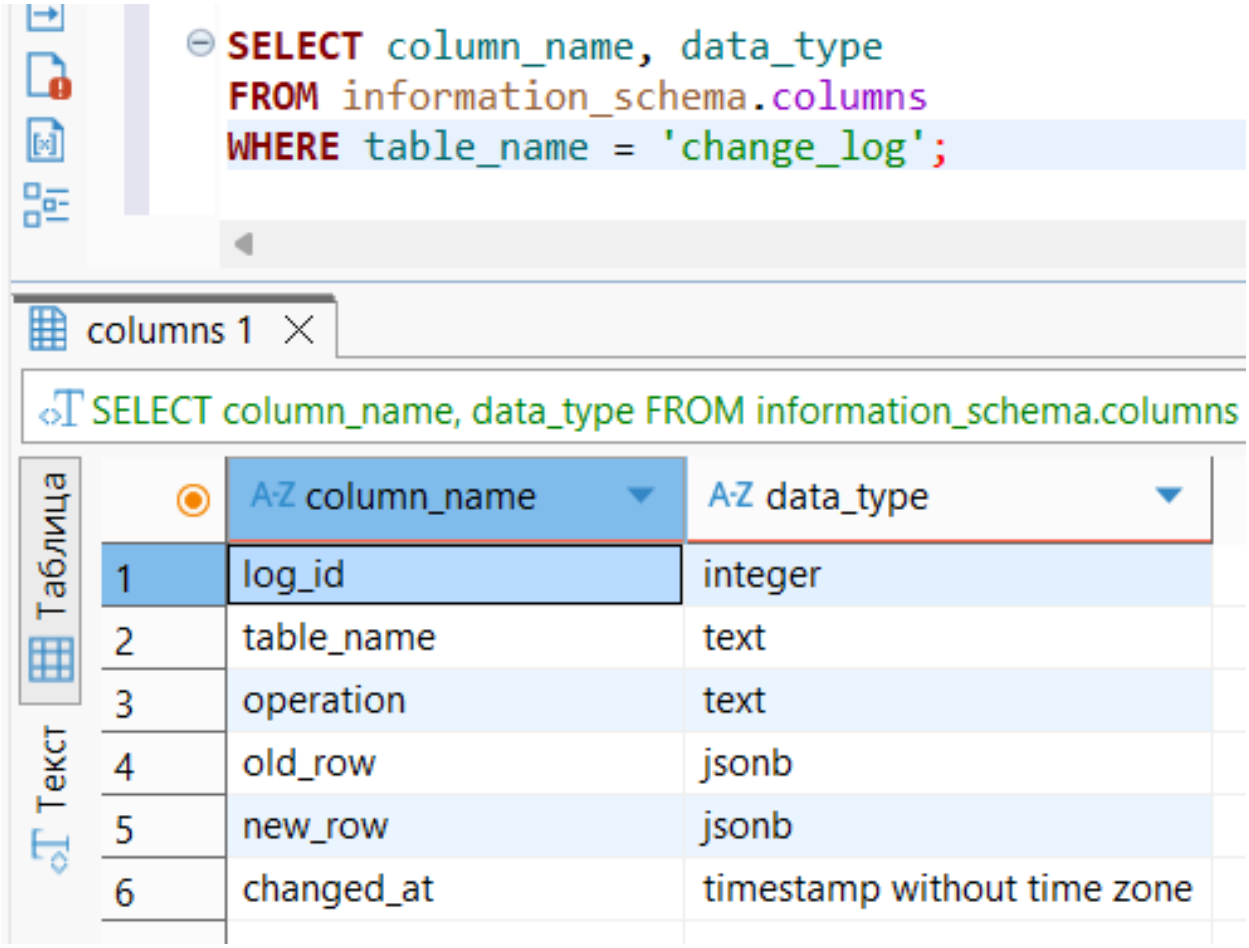
```
SELECT 'FUTURE' AS part, * FROM event_log_future LIMIT 10;
```

event_log_2024_h2 1 | event_log_2025_h1 1 (2) | event_log_future 1 (3)

SELECT '2024_H2' AS part, * FROM event_log_2024_h2 LIMIT 10; SELECT '2 — Фильтрация даных не поддерживается

	part	id	event_type	event_date	details
1	FUTURE	3	error	2025-07-03	{"code": 403, "message": "Auto-generated test error"}
2	FUTURE	12	login	2025-09-24	{"ip": "192.168.173.10", "device": "desktop"}
3	FUTURE	16	error	2025-07-26	{"code": 404, "message": "Auto-generated test error"}
4	FUTURE	17	login	2025-07-21	{"ip": "192.168.130.10", "device": "tablet"}
5	FUTURE	18	login	2025-08-22	{"ip": "192.168.222.10", "device": "desktop"}
6	FUTURE	23	error	2025-07-12	{"code": 400, "message": "Auto-generated test error"}
7	FUTURE	26	error	2025-08-13	{"code": 401, "message": "Auto-generated test error"}
8	FUTURE	42	error	2025-07-13	{"code": 401, "message": "Auto-generated test error"}
9	FUTURE	45	error	2025-09-22	{"code": 400, "message": "Auto-generated test error"}
10	FUTURE	50	error	2025-09-17	{"code": 400, "message": "Auto-generated test error"}

Таблица 10. Структура таблицы change_log



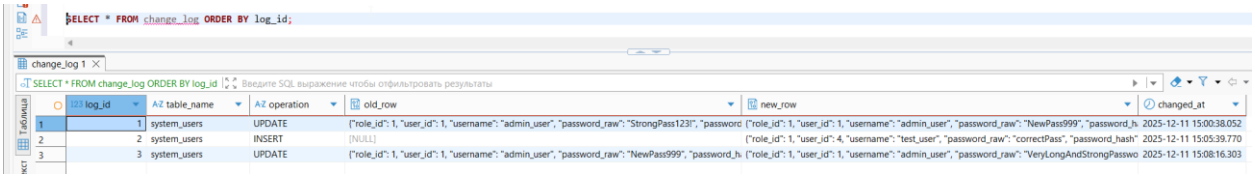
```
SELECT column_name, data_type
FROM information_schema.columns
WHERE table_name = 'change_log';
```

columns 1

SELECT column_name, data_type FROM information_schema.columns

	column_name	data_type
1	log_id	integer
2	table_name	text
3	operation	text
4	old_row	jsonb
5	new_row	jsonb
6	changed_at	timestamp without time zone

Таблица 11. Данные таблицы change_log



```
SELECT * FROM change_log ORDER BY log_id;
```

change_log 1

SELECT * FROM change_log ORDER BY log_id

	log_id	table_name	operation	old_row	new_row	changed_at
1	1	system_users	UPDATE	{"role_id": 1, "user_id": 1, "username": "admin_user", "password_raw": "StrongPass123!", "password	{"role_id": 1, "user_id": 1, "username": "admin_user", "password_raw": "NewPass999", "password_h	2025-12-11 15:00:38.052
2	2	system_users	INSERT	[NULL]	{"role_id": 1, "user_id": 4, "username": "test_user", "password_raw": "correctPass", "password_hash"	2025-12-11 15:05:39.770
3	3	system_users	UPDATE	{"role_id": 1, "user_id": 1, "username": "admin_user", "password_raw": "NewPass999", "password_h	{"role_id": 1, "user_id": 1, "username": "admin_user", "password_raw": "VeryLongAndStrongPasswo	2025-12-11 15:08:16.303

Задание №1: хранение паролей (BYTEA)

```
CREATE TABLE system_roles (
    role_id SERIAL PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL UNIQUE
);
```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0,019s
Start time	Thu Dec 11 14:48:17 MSK 2025
Finish time	Thu Dec 11 14:48:17 MSK 2025
Query	CREATE TABLE system_roles (
	role_id SERIAL PRIMARY KEY,
	role_name VARCHAR(50) NOT NULL UNIQUE
	)

Рисунок 1 – Создание таблицы ролей

```
CREATE TABLE system_users (
    user_id SERIAL PRIMARY KEY,
    username VARCHAR(100) NOT NULL UNIQUE,
    password_raw TEXT,
    password_hash BYTEA NOT NULL,
    role_id INT REFERENCES system_roles(role_id)
);
```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0,016s
Start time	Thu Dec 11 14:49:14 MSK 2025
Finish time	Thu Dec 11 14:49:14 MSK 2025
Query	CREATE TABLE system_users (
	user_id SERIAL PRIMARY KEY,
	username VARCHAR(100) NOT NULL UNIQUE,

Рисунок 2 – Создание таблицы пользователей

```
INSERT INTO system_roles (role_name)
VALUES ('Администратор'),
       ('Мастер'),
       ('Менеджер');
```

Статистика 1

Name	Value
Updated Rows	3
Execute time	0,012s
Start time	Thu Dec 11 14:50:37 MSK 2025
Finish time	Thu Dec 11 14:50:37 MSK 2025
Query	INSERT INTO system_roles (role_name)
	VALUES ('Администратор'),
	('Мастер'),
	('Менеджер')

Рисунок 3 – Заполнение ролей

```
INSERT INTO system_users (username, password_raw, password_hash, role_id)
VALUES
('admin_user', 'StrongPass123!', digest('StrongPass123!', 'sha256'), 1),
('worker',     'Pharma2025',     digest('Pharma2025_BAD', 'sha256'), 2);
```

Статистика 1

Name	Value
Updated Rows	2
Execute time	0,013s
Start time	Thu Dec 11 14:51:11 MSK 2025
Finish time	Thu Dec 11 14:51:11 MSK 2025
Query	INSERT INTO system_users (username, password_raw, password_hash, role_id)
	VALUES
	('admin_user', 'StrongPass123!', digest('StrongPass123!', 'sha256'), 1),
	('worker', 'Pharma2025', digest('Pharma2025_BAD', 'sha256'), 2)

Рисунок 4 – Вставка пользователей с SHA-256

```
SELECT
    username,
    password_raw,
    encode(password_hash, 'hex') AS password_hash_hex,
    digest(password_raw, 'sha256') = password_hash AS is_valid
FROM system_users;
```

system_users 1

SELECT username, password_raw, encode(password_hash, 'hex') AS passw | Введите SQL выражение чтобы отфильтровать результаты

Таблица

	username	password_raw	password_hash_hex	is_valid
1	admin_user	StrongPass123!	805bd951772627f3d1a607084df1727c6caad60447c5d73febf7be2d2fe17fd8	[v]
2	worker	Pharma2025	365047c3a515fdd97cdacfff821e3ae87bec1816c11ea6ee8b0d443865f5effd	[ ]

Рисунок 5 – Проверка корректности хешей

Задание №2. секционирование таблицы логов

```
CREATE TABLE event_log (
    id SERIAL,
    event_type VARCHAR(50) NOT NULL,
    event_date DATE NOT NULL,
    details JSONB NOT NULL
) PARTITION BY RANGE (event_date);
```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0,073s
Start time	Thu Dec 11 15:30:08 MSK 2025
Finish time	Thu Dec 11 15:30:08 MSK 2025
Query	CREATE TABLE event_log (id SERIAL, event_type VARCHAR(50) NOT NULL, event_date DATE NOT NULL, details JSONB NOT NULL) PARTITION BY RANGE (event_date)

Рисунок 6 – Создание родительской таблицы (без PK)

```
CREATE TABLE event_log_2024_h2
    PARTITION OF event_log
    FOR VALUES FROM ('2024-07-01') TO ('2025-01-01');
```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0,022s
Start time	Thu Dec 11 15:30:41 MSK 2025
Finish time	Thu Dec 11 15:30:41 MSK 2025
Query	CREATE TABLE event_log_2024_h2 PARTITION OF event_log FOR VALUES FROM ('2024-07-01') TO ('2025-01-01')

Рисунок 7 - Создание первой секции

```
CREATE TABLE event_log_2025_h1
    PARTITION OF event_log
    FOR VALUES FROM ('2025-01-01') TO ('2025-07-01');
```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0,019s
Start time	Thu Dec 11 15:31:03 MSK 2025
Finish time	Thu Dec 11 15:31:03 MSK 2025
Query	CREATE TABLE event_log_2025_h1 PARTITION OF event_log FOR VALUES FROM ('2025-01-01') TO ('2025-07-01')

Рисунок 8 – Создание второй секции

```
CREATE TABLE event_log_future
    PARTITION OF event_log
    FOR VALUES FROM ('2025-07-01') TO (MAXVALUE);
```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0,017s
Start time	Thu Dec 11 15:31:36 MSK 2025
Finish time	Thu Dec 11 15:31:36 MSK 2025
Query	CREATE TABLE event_log_future PARTITION OF event_log FOR VALUES FROM ('2025-07-01') TO (MAXVALUE)

Рисунок 9 – Создание секции по умолчанию

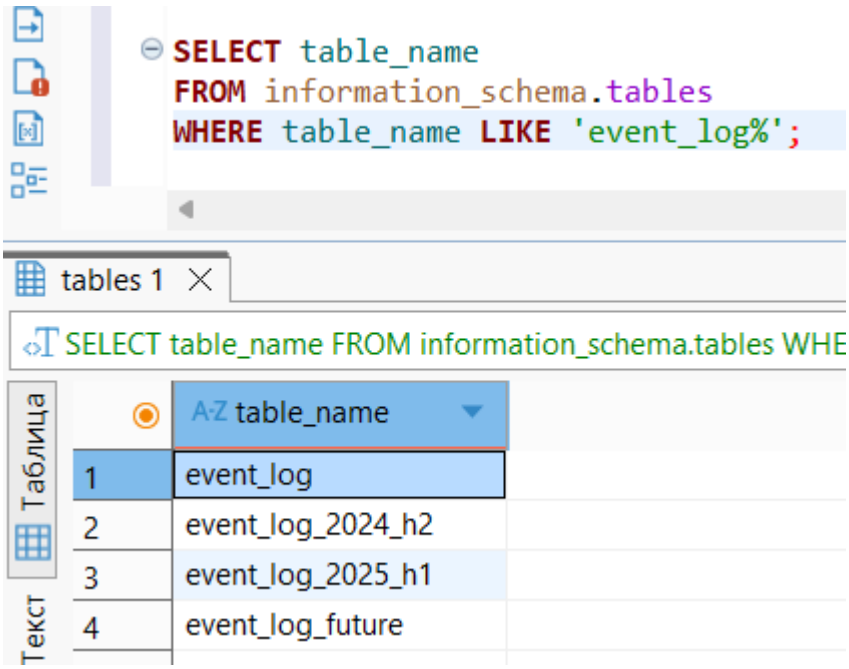
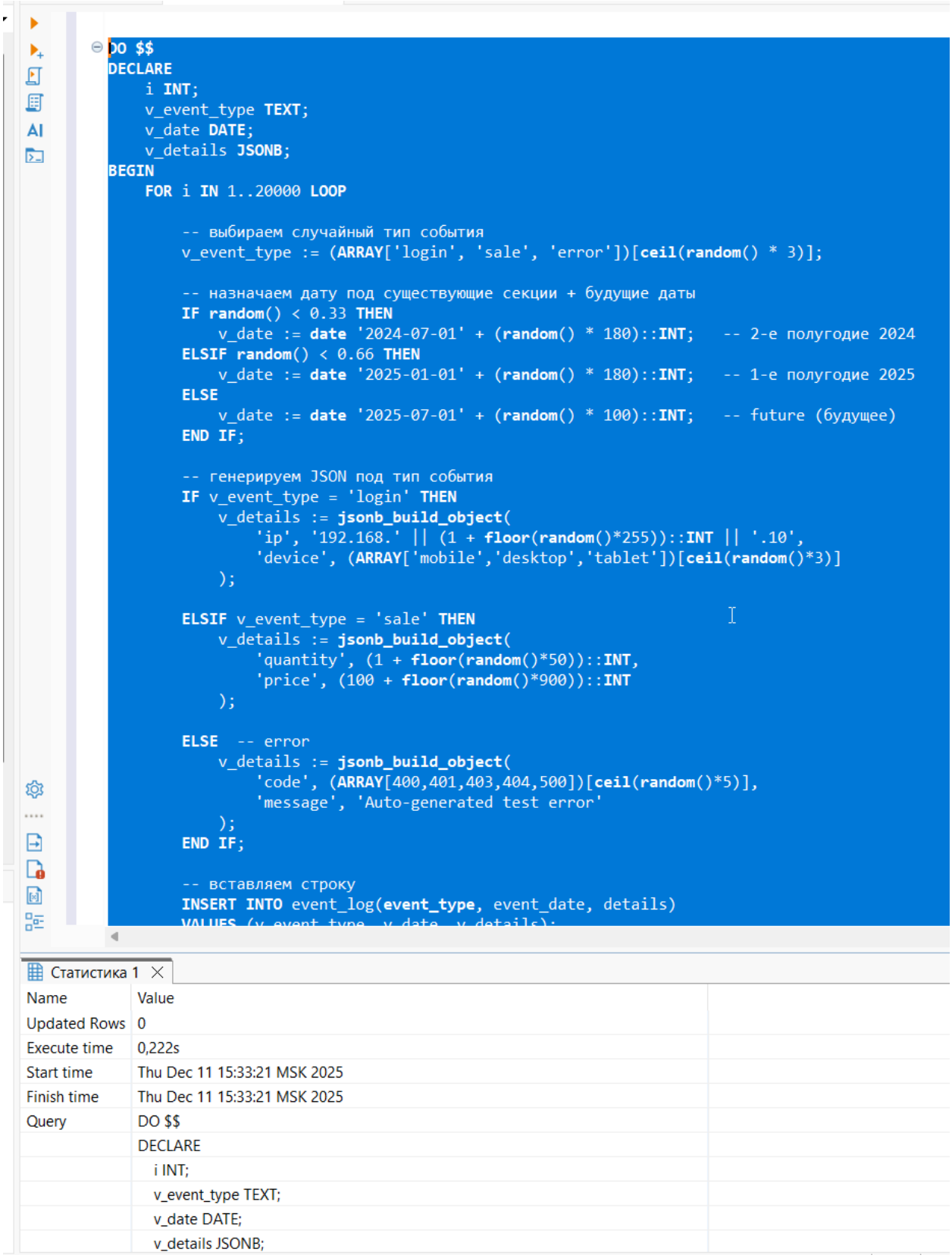


Рисунок 10 – Проверка

Задание №3: генерация данных



```
DO $$
DECLARE
    i INT;
    v_event_type TEXT;
    v_date DATE;
    v_details JSONB;
BEGIN
    FOR i IN 1..20000 LOOP

        -- выбираем случайный тип события
        v_event_type := (ARRAY['login', 'sale', 'error'])[ceil(random() * 3)];

        -- назначаем дату под существующие секции + будущие даты
        IF random() < 0.33 THEN
            v_date := date '2024-07-01' + (random() * 180)::INT;   -- 2-е полугодие 2024
        ELSIF random() < 0.66 THEN
            v_date := date '2025-01-01' + (random() * 180)::INT;   -- 1-е полугодие 2025
        ELSE
            v_date := date '2025-07-01' + (random() * 100)::INT;   -- future (будущее)
        END IF;

        -- генерируем JSON под тип события
        IF v_event_type = 'login' THEN
            v_details := jsonb_build_object(
                'ip', '192.168.' || (1 + floor(random()*255))::INT || '.10',
                'device', (ARRAY['mobile','desktop','tablet'])[ceil(random()*3)]
            );

        ELSIF v_event_type = 'sale' THEN
            v_details := jsonb_build_object(
                'quantity', (1 + floor(random()*50))::INT,
                'price', (100 + floor(random()*900))::INT
            );

        ELSE  -- error
            v_details := jsonb_build_object(
                'code', (ARRAY[400,401,403,404,500])[ceil(random()*5)],
                'message', 'Auto-generated test error'
            );
        END IF;

        -- вставляем строку
        INSERT INTO event_log(event_type, event_date, details)
        VALUES (v_event_type, v_date, v_details);
```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0,222s
Start time	Thu Dec 11 15:33:21 MSK 2025
Finish time	Thu Dec 11 15:33:21 MSK 2025
Query	DO $$
	DECLARE
	i INT;
	v_event_type TEXT;
	v_date DATE;
	v_details JSONB;

Рисунок 11 – Создадим скрипт генерации 20 000 записей

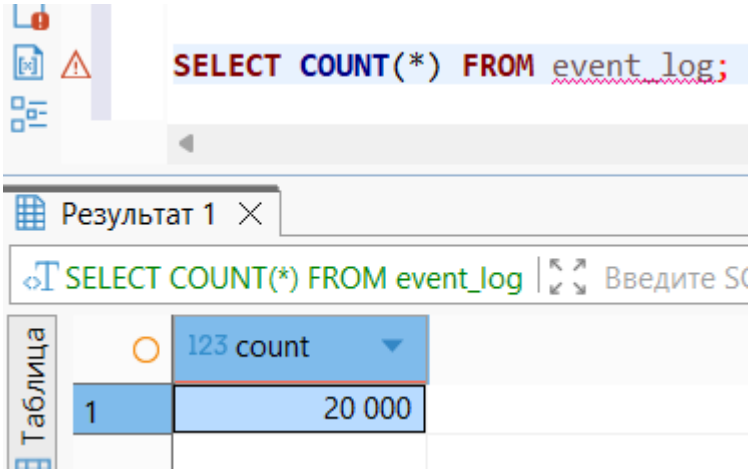


Рисунок 12 – Проверка отработки скрипта

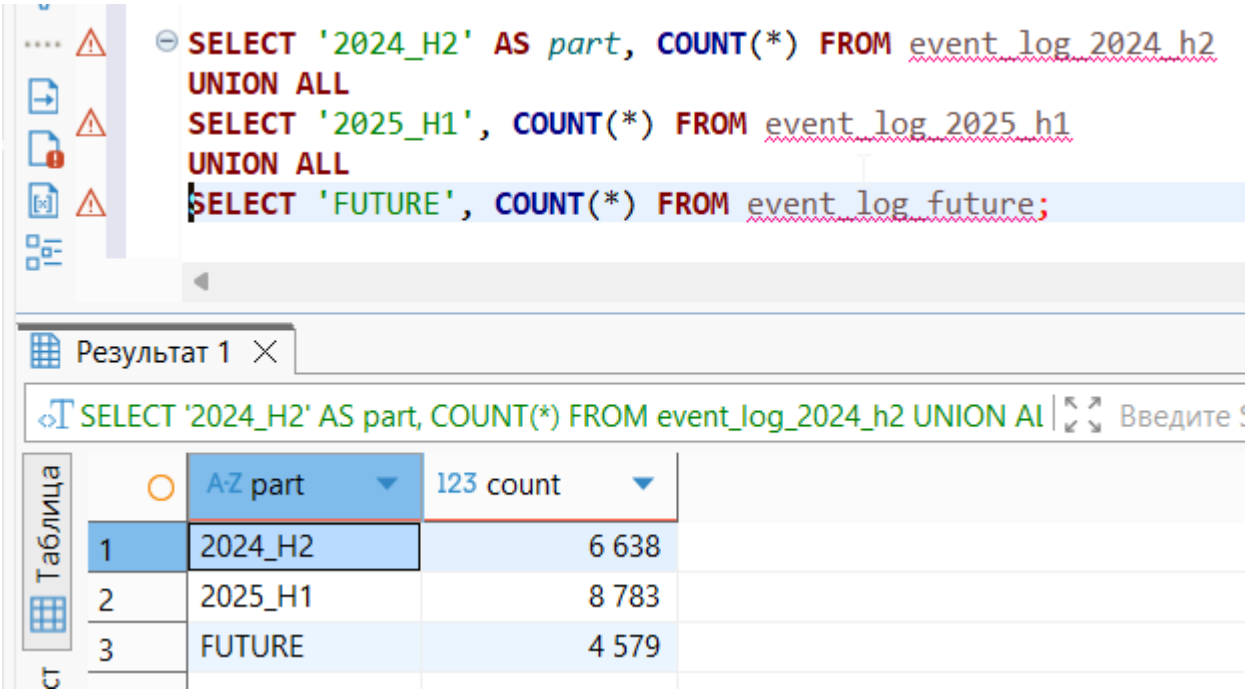


Рисунок 13 - Проверка распределения по секциям

Задание №4: анализ и поиск по JSONB

```sql
SELECT *
FROM event_log
WHERE event_type = 'sale'
  AND (details->>'price')::INT > 800;
```

event_log 1

SELECT * FROM event_log WHERE event_type = 'sale' AND (details->>'pr

Введите SQL выражение чт

Таблица | Текст | Запись

	id	event_type	event_date	details
1	48	sale	2024-10-29	{"price": 927, "quantity": 27}
2	247	sale	2024-12-26	{"price": 961, "quantity": 1}
3	325	sale	2024-09-28	{"price": 890, "quantity": 36}
4	364	sale	2024-08-31	{"price": 839, "quantity": 4}
5	370	sale	2024-09-19	{"price": 867, "quantity": 23}
6	380	sale	2024-12-16	{"price": 926, "quantity": 30}
7	568	sale	2024-08-05	{"price": 822, "quantity": 50}
8	621	sale	2024-09-12	{"price": 973, "quantity": 25}
9	654	sale	2024-10-02	{"price": 883, "quantity": 4}
10	673	sale	2024-12-22	{"price": 842, "quantity": 2}
11	809	sale	2024-08-22	{"price": 996, "quantity": 40}
12	827	sale	2024-09-23	{"price": 996, "quantity": 36}
13	859	sale	2024-12-26	{"price": 824, "quantity": 11}
14	953	sale	2024-12-15	{"price": 938, "quantity": 14}
15	975	sale	2024-12-20	{"price": 945, "quantity": 47}
16	1 052	sale	2024-12-24	{"price": 929, "quantity": 34}
17	1 108	sale	2024-08-20	{"price": 991, "quantity": 29}
18	1 145	sale	2024-12-06	{"price": 895, "quantity": 33}
19	1 152	sale	2024-10-14	{"price": 846, "quantity": 40}
20	1 199	sale	2024-08-03	{"price": 901, "quantity": 18}
21	1 221	sale	2024-12-08	{"price": 822, "quantity": 3}
22	1 246	sale	2024-10-25	{"price": 827, "quantity": 25}
23	1 297	sale	2024-08-24	{"price": 868, "quantity": 23}
24	1 410	sale	2024-11-12	{"price": 823, "quantity": 22}
25	1 430	sale	2024-08-24	{"price": 903, "quantity": 21}
26	1 436	sale	2024-07-25	{"price": 808, "quantity": 32}

Рисунок 14 – Найти все продажи, где цена > 800

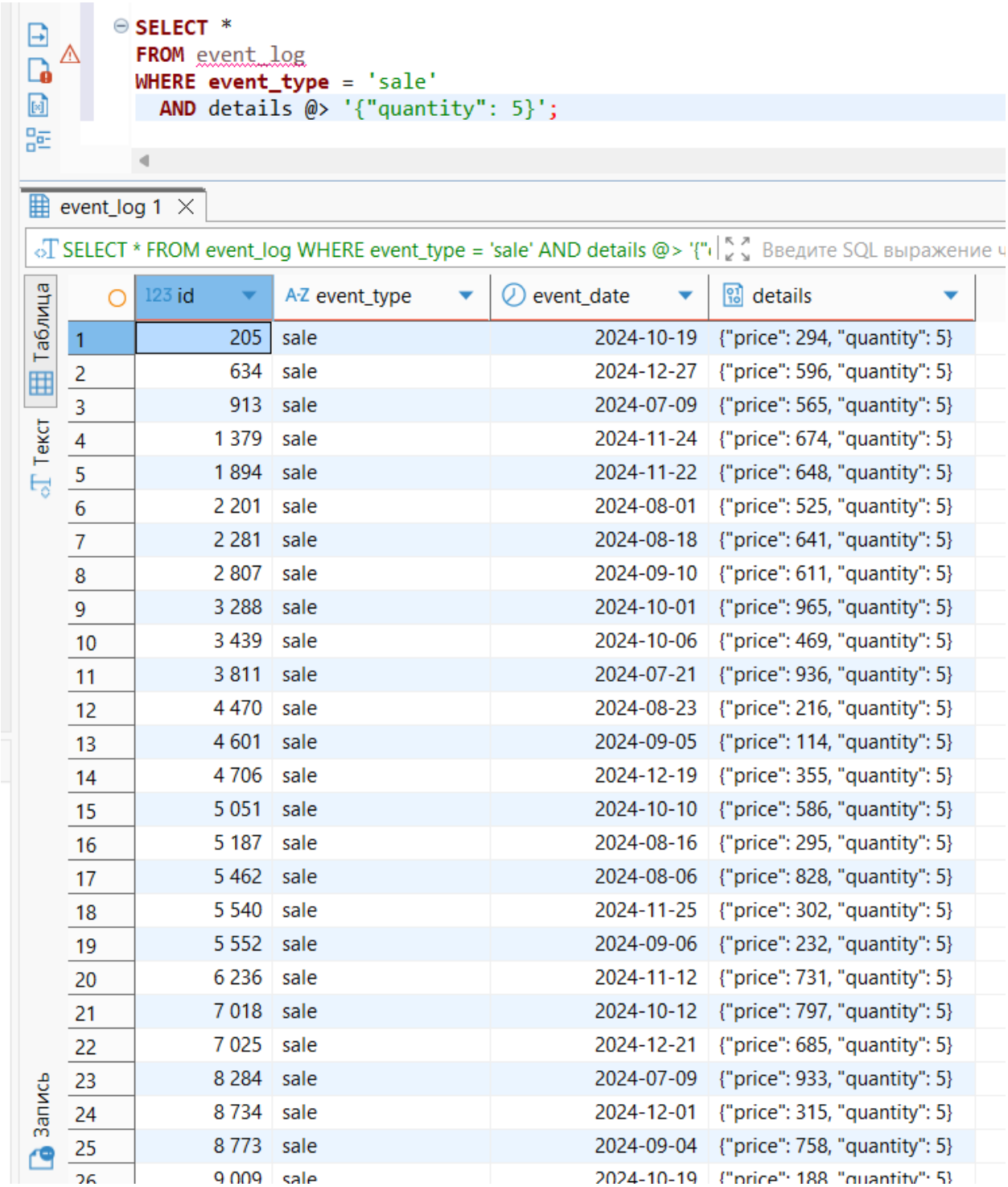


Рисунок 15 – Найти продажи, где количество = 5

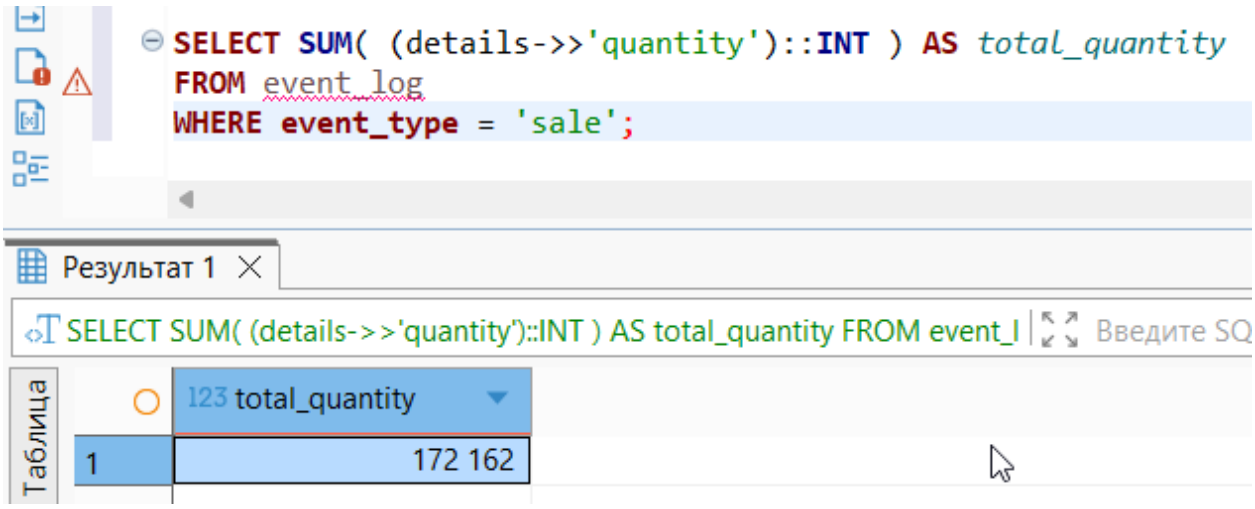


Рисунок 16 – Общая сумма quantity по всем ТО

Задание №5: модификация данных JSONB

```
UPDATE event_log
SET details = jsonb_set(details, '{version}', '"1.0"', true)
WHERE event_type = 'sale';
```

Статистика 1

Name	Value
Updated Rows	6746
Execute time	0,058s
Start time	Thu Dec 11 15:38:00 MSK 2025
Finish time	Thu Dec 11 15:38:00 MSK 2025
Query	UPDATE event_log
	SET details = jsonb_set(details, '{version}', '"1.0"', true)
	WHERE event_type = 'sale'

Рисунок 17 – Массовое добавление ключа

```
SELECT *
FROM event_log
WHERE event_type = 'sale'
LIMIT 10;
```

event_log 1

SELECT * FROM event_log WHERE event_type = 'sale' LIMIT 10 | Введите SQL выражение чтобы отфильтровать р

Таблица | Текст

	123 id	A-Z event_type	event_date	detail
1	1	sale	2024-07-17	{"price": 637, "version": "1.0", "quantity": 26}
2	22	sale	2024-10-24	{"price": 615, "version": "1.0", "quantity": 47}
3	48	sale	2024-10-29	{"price": 927, "version": "1.0", "quantity": 27}
4	56	sale	2024-07-03	{"price": 201, "version": "1.0", "quantity": 29}
5	58	sale	2024-10-09	{"price": 332, "version": "1.0", "quantity": 26}
6	59	sale	2024-10-30	{"price": 231, "version": "1.0", "quantity": 50}
7	70	sale	2024-08-09	{"price": 120, "version": "1.0", "quantity": 39}
8	73	sale	2024-08-11	{"price": 411, "version": "1.0", "quantity": 8}
9	74	sale	2024-10-12	{"price": 746, "version": "1.0", "quantity": 28}
10	87	sale	2024-12-09	{"price": 389, "version": "1.0", "quantity": 9}

Рисунок 18 – Проверка результата

```
UPDATE event_log
SET details = jsonb_set(details, '{quantity}', '999', false)
WHERE id = 87;
```

Статистика 1

Name	Value
Updated Rows	1
Execute time	0,013s
Start time	Thu Dec 11 15:39:23 MSK 2025
Finish time	Thu Dec 11 15:39:23 MSK 2025
Query	UPDATE event_log
	SET details = jsonb_set(details, '{quantity}', '999', false)
	WHERE id = 87

Рисунок 19 - Изменим quantity у записи с id 87

```
SELECT id, event_type, details
FROM event_log
WHERE id = 87;
```

event_log 1

SELECT id, event_type, details FROM event_log WHERE id = 87

Введите SQL выражение ч

Таблица

	123 id	A-Z event_type	details
1	87	sale	{"price": 389, "version": "1.0", "quantity": 999}

Рисунок 20 - Проверка

ВЫВОД

В ходе работы были освоены методы хранения чувствительных данных с использованием BYTEA и хеширования паролей через digest. Создана секционированная таблица логов, позволяющая эффективно распределять большие объёмы данных по диапазонам дат.

Сгенерированные JSONB-события продемонстрировали возможности хранения и обработки полуструктурированных данных: фильтрацию, поиск по фрагменту и агрегацию.

Также выполнена модификация JSONB — массовое добавление поля и точечное изменение значения. Работа позволила закрепить навыки использования JSONB и секционирования для повышения гибкости и производительности PostgreSQL.