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

по дисциплине «Разработка баз данных»

Практическое занятие № 5

Студенты группы *ИКБО-66-23 Смирнов А.Ю.* _____
(подпись)

Ассистент *Копылова Я.А.*

_____
(подпись)

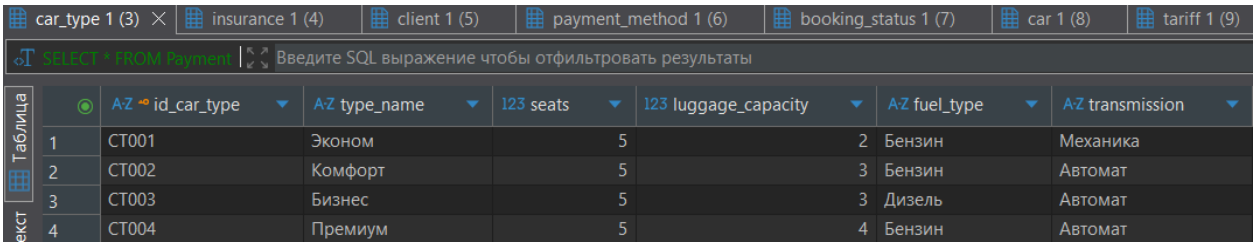
Отчет представлен «___»_____2025 г.

Москва 2025 г.

ПРАКТИЧЕСКАЯ РАБОТА №5. ОБЪЕКТЫ БАЗЫ ДАННЫХ: ПРЕДСТАВЛЕНИЯ И ХРАНИМЫЕ ПРОЦЕДУРЫ

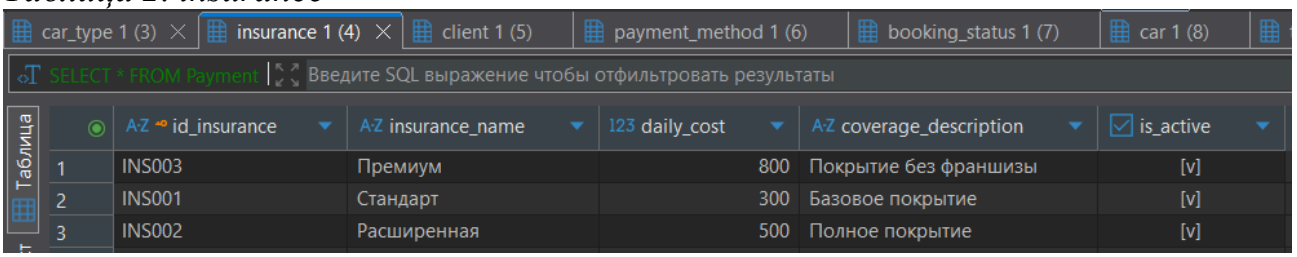
Цель работы: Работа направлена на формирование у студентов углубленных навыков работы с объектами баз данных в СУБД PostgreSQL, смещая акцент от прямого манипулирования данными к созданию переиспользуемых логических конструкций.

Таблица 1. Car_type



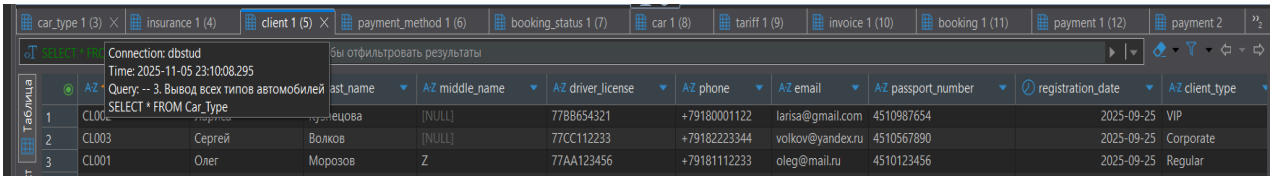
	id_car_type	type_name	seats	luggage_capacity	fuel_type	transmission
1	CT001	Эконом	5	2	Бензин	Механика
2	CT002	Комфорт	5	3	Бензин	Автомат
3	CT003	Бизнес	5	3	Дизель	Автомат
4	CT004	Премиум	5	4	Бензин	Автомат

Таблица 2. insurance



	id_insurance	insurance_name	daily_cost	coverage_description	is_active
1	INS003	Премиум	800	Покрытие без франшизы	[v]
2	INS001	Стандарт	300	Базовое покрытие	[v]
3	INS002	Расширенная	500	Полное покрытие	[v]

Таблица 3. client



1	CL00		[NULL]	77BB654321	+79180001122	larisa@gmail.com	4510987654	2025-09-25	VIP
2	CL003	Сергей	Волков	[NULL]	77CC112233	+79182223344	volkov@yandex.ru	4510567890	2025-09-25	Corporate
3	CL001	Олег	Морозов	Z	77AA123456	+79181112233	oleg@mail.ru	4510123456	2025-09-25	Regular

Таблица 4. Payment_method

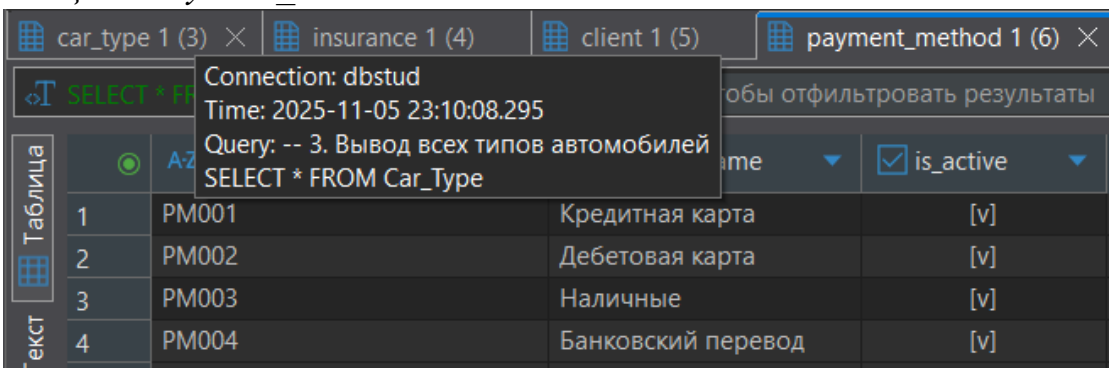


Таблица 5. Booking_status

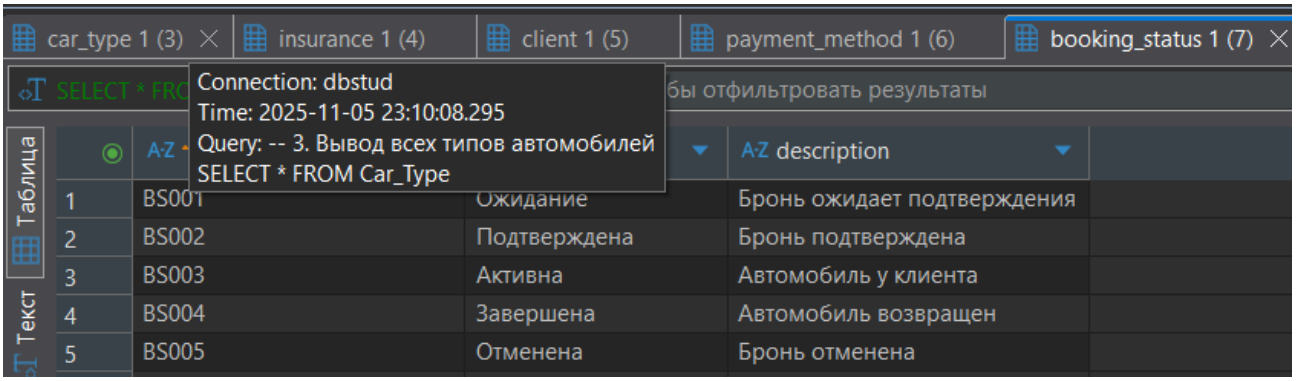


Таблица 6. Car

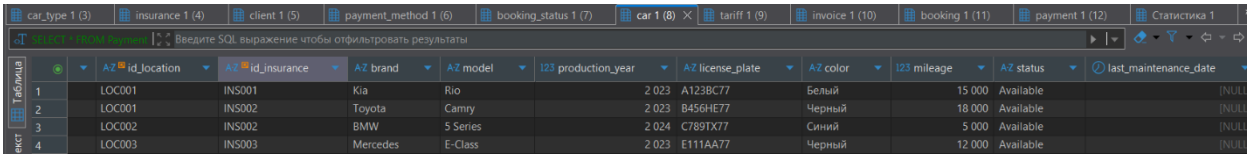


Таблица 7. tariff

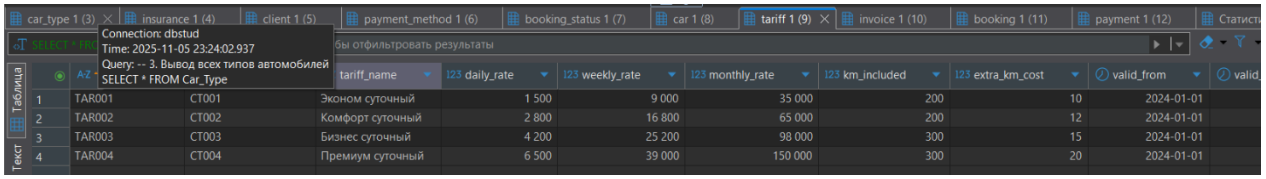


Таблица 8. invoice

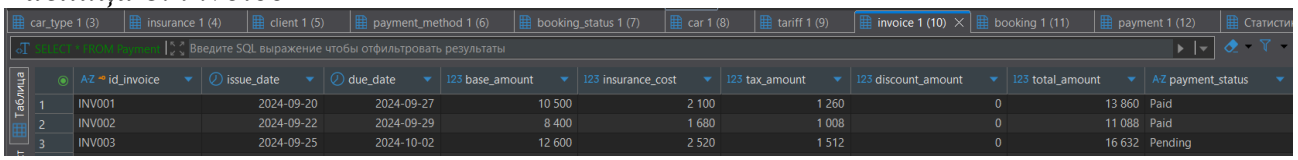


Таблица 9. booking

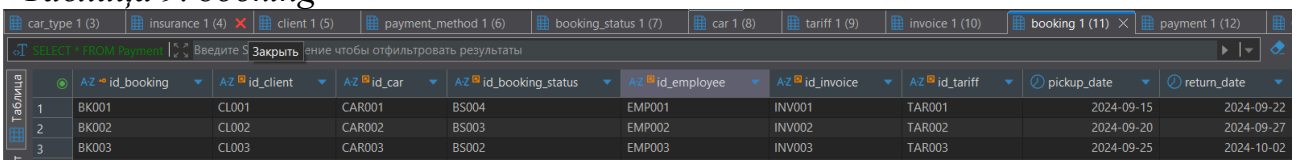
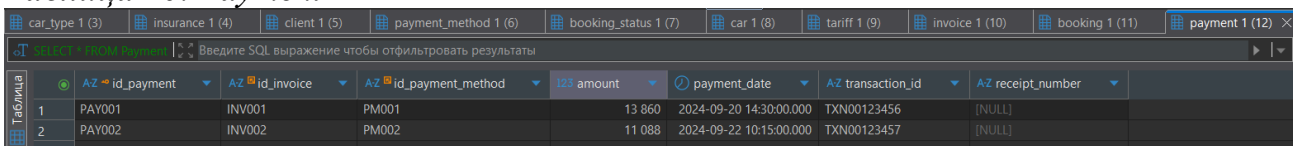
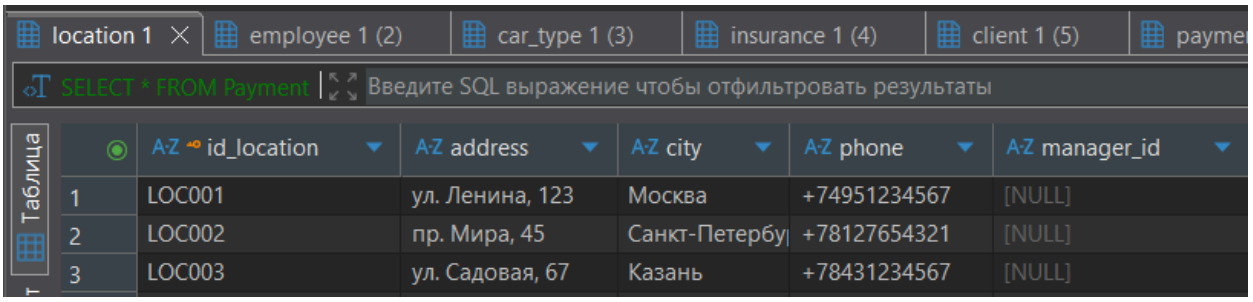


Таблица 10. Payment



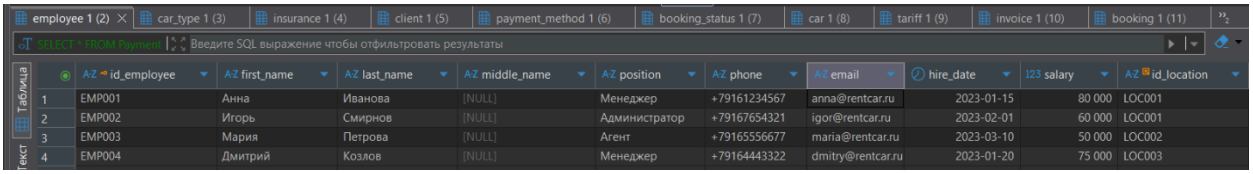
	id_payment	id_invoice	id_payment_method	amount	payment_date	transaction_id	receipt_number
1	PAY001	INV001	PM001	13 860	2024-09-20 14:30:00.000	TXN00123456	[NULL]
2	PAY002	INV002	PM002	11 088	2024-09-22 10:15:00.000	TXN00123457	[NULL]

Таблица 11. Location



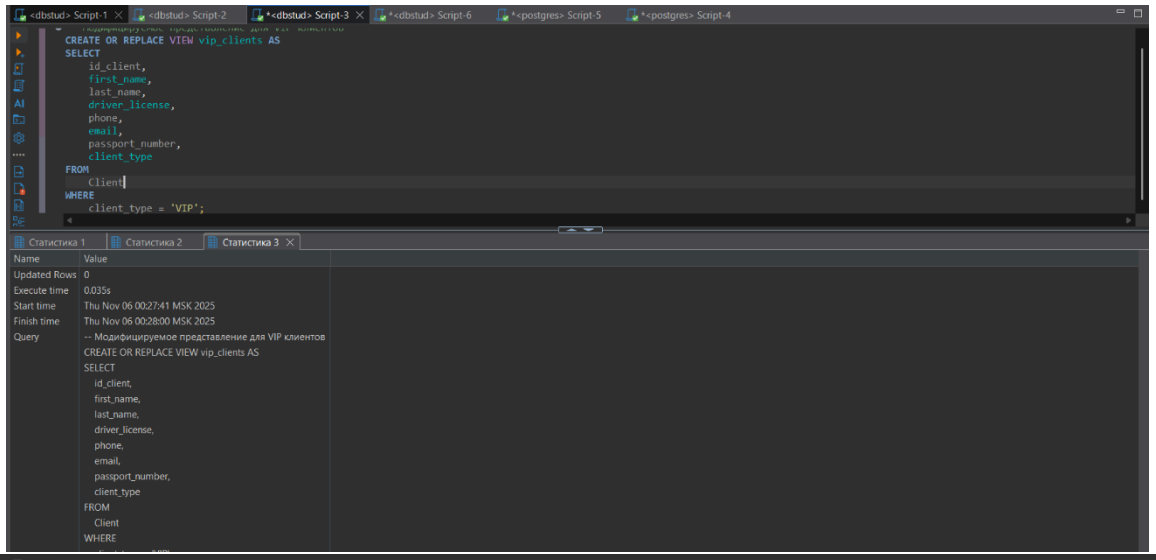
	id_location	address	city	phone	manager_id
1	LOC001	ул. Ленина, 123	Москва	+74951234567	[NULL]
2	LOC002	пр. Мира, 45	Санкт-Петербу	+78127654321	[NULL]
3	LOC003	ул. Садовая, 67	Казань	+78431234567	[NULL]

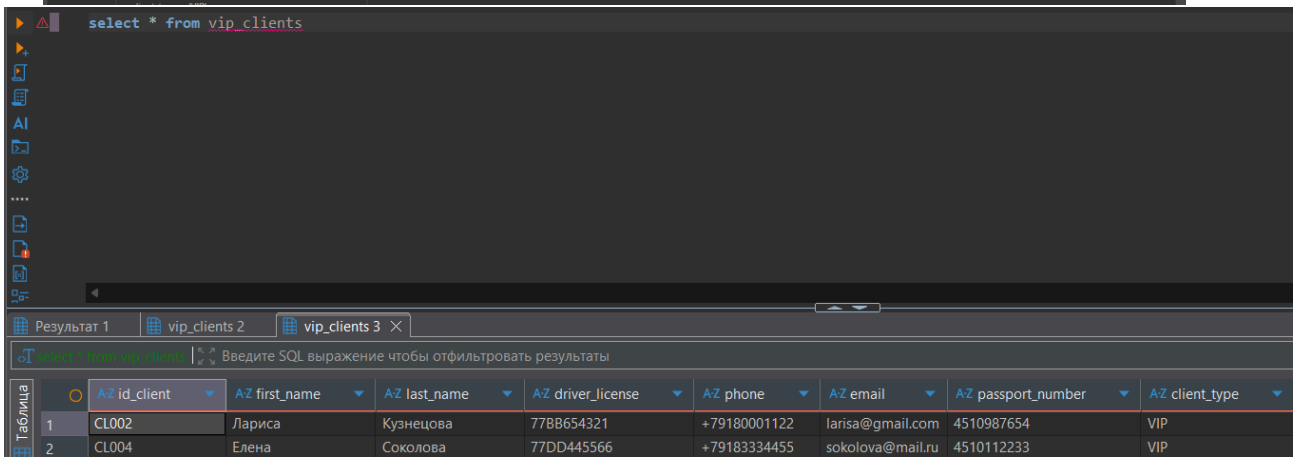
Таблица 11. Employee



	id_employee	first_name	last_name	middle_name	position	phone	email	hire_date	salary	id_location
1	EMP001	Анна	Иванова	[NULL]	Менеджер	+79161234567	anna@rentcar.ru	2023-01-15	80 000	LOC001
2	EMP002	Игорь	Смирнов	[NULL]	Администратор	+79167654321	igor@rentcar.ru	2023-02-01	60 000	LOC001
3	EMP003	Мария	Петрова	[NULL]	Агент	+79165556677	maria@rentcar.ru	2023-03-10	50 000	LOC002
4	EMP004	Дмитрий	Козлов	[NULL]	Менеджер	+79164443322	dmitry@rentcar.ru	2023-01-20	75 000	LOC003

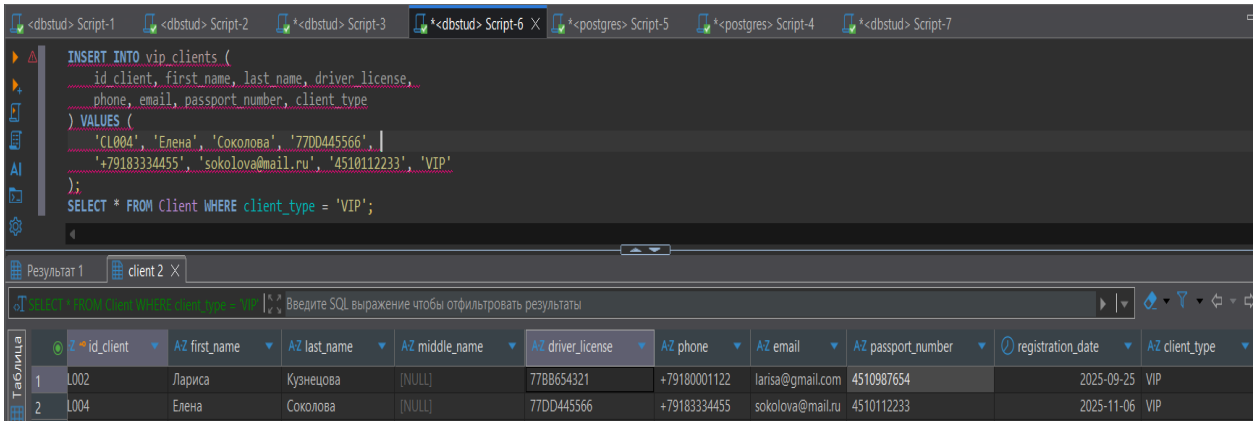
Задание №1: Создание модифицируемого представления

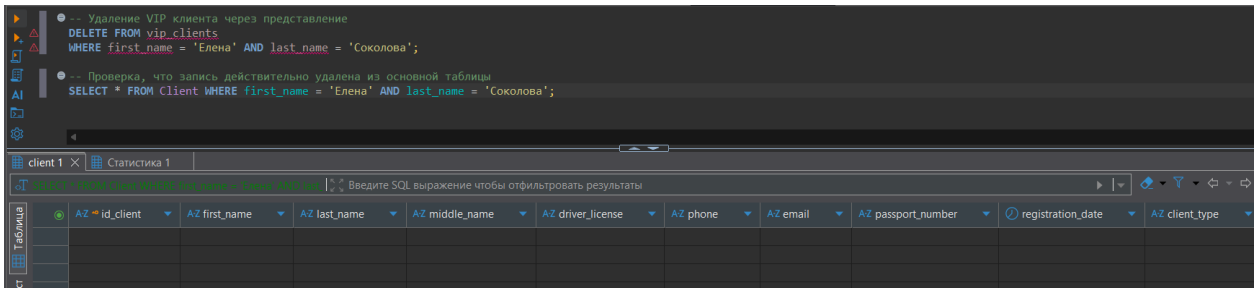




Представление выбирает клиентов с типом "VIP" с полной информацией об их данных. Оно является модифицируемым, так как удовлетворяет всем условиям PostgreSQL: одна базовая таблица (Client), нет агрегатных функций, GROUP BY, DISTINCT, оконных функций и других ограничивающих конструкций.

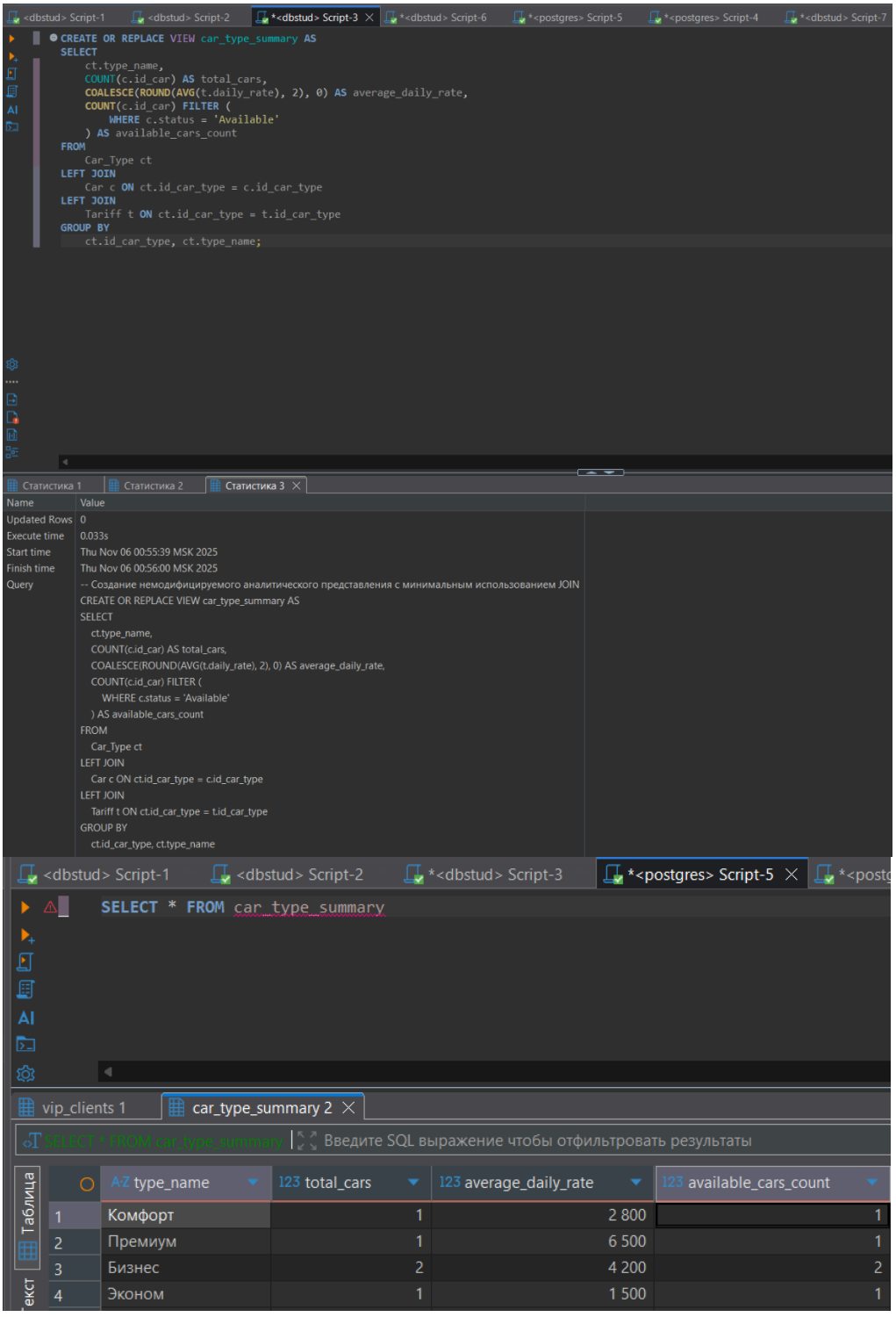
Задание №2: Модификация данных через представление





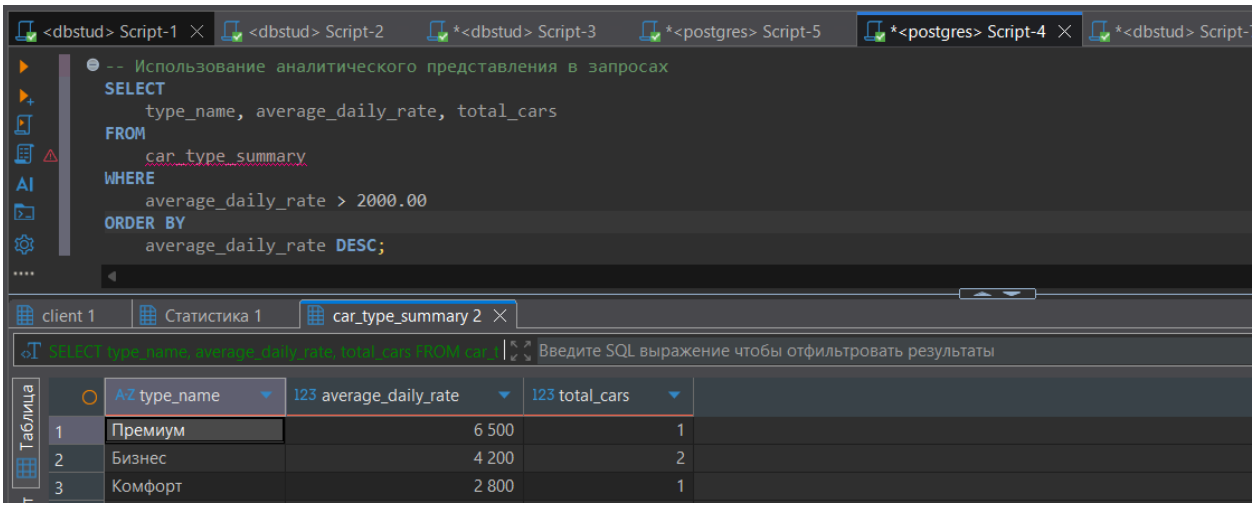
Демонстрируется возможность выполнения DML-операций через представление. DELETE удаляет VIP клиента по имени и фамилии, а SELECT проверяет отсутствие записи в базовой таблице Client. Изменения автоматически применяются к базовой таблице Client через модифицируемое представление vip_clients.

Задание №3: Создание немодифицируемого аналитического представления



Представление объединяет данные из трех таблиц (Car_Type, Car, Tariff) и использует агрегатные функции для анализа статистики по типам автомобилей. Оно является немодифицируемым из-за использования JOIN, GROUP BY и агрегатных функций.

Задание №4: Использование аналитического представления в запросах



Запрос использует созданное представление для отбора типов автомобилей с высокой средней суточной стоимостью аренды (больше 2000 рублей) и наличием автомобилей в парке. Это демонстрирует удобство использования представлений для сложной аналитики в сфере проката автомобилей.

Задание №5: Создание и обновление материализованного представления

Статистика 1 | Статистика 2 | Статистика 3 | Статистика 4 | totalrentalsbyclient 5

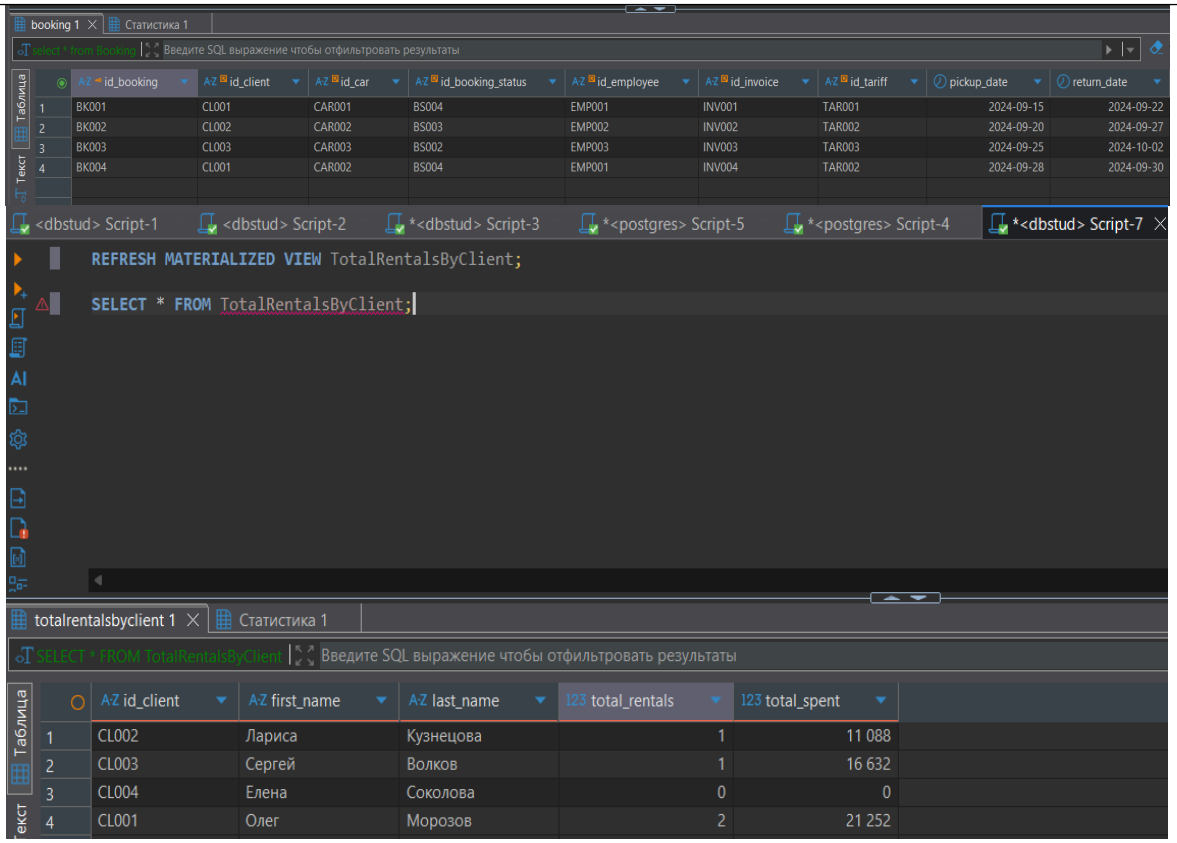
SELECT * FROM TotalRentalsByClient — Введите SQL выражение чтобы отфильтровать результаты

Таблица | Текст

	id_client	first_name	last_name	total_rentals	total_spent
1	CL002	Лариса	Кузнецова	1	11 088
2	CL003	Сергей	Волков	1	16 632
3	CL004	Елена	Соколова	0	0
4	CL001	Олег	Морозов	1	13 860

```sql
-- Создание материализованного представления для отчета по аренде автомобиле
CREATE MATERIALIZED VIEW TotalRentalsByClient AS
SELECT
    c.id_client,
    c.first_name,
    c.last_name,
    COUNT(b.id_booking) AS total_rentals,
    COALESCE(SUM(i.total_amount), 0) AS total_spent
FROM
    Client c
LEFT JOIN
    Booking b ON c.id_client = b.id_client
LEFT JOIN
    Invoice i ON b.id_invoice = i.id_invoice
GROUP BY
    c.id_client, c.first_name, c.last_name;

-- Просмотр данных материализованного представления
SELECT * FROM TotalRentalsByClient;
```

```
-- 2. Демонстрация обновления данных
-- Добавим новую сессию для проверки
INSERT INTO Booking (id_booking, id_client, id_car, id_booking_status, id_employee,
id_invoice, id_tariff, pickup_date, return_date, total_days, pickup_location,
return_location)
VALUES ('BK004', 'CL001', 'CAR002', 'BS004', 'EMP001', 'INV004', 'TAR002', '2024-09-
28', '2024-09-30', 2, 'LOC001', 'LOC001');
```



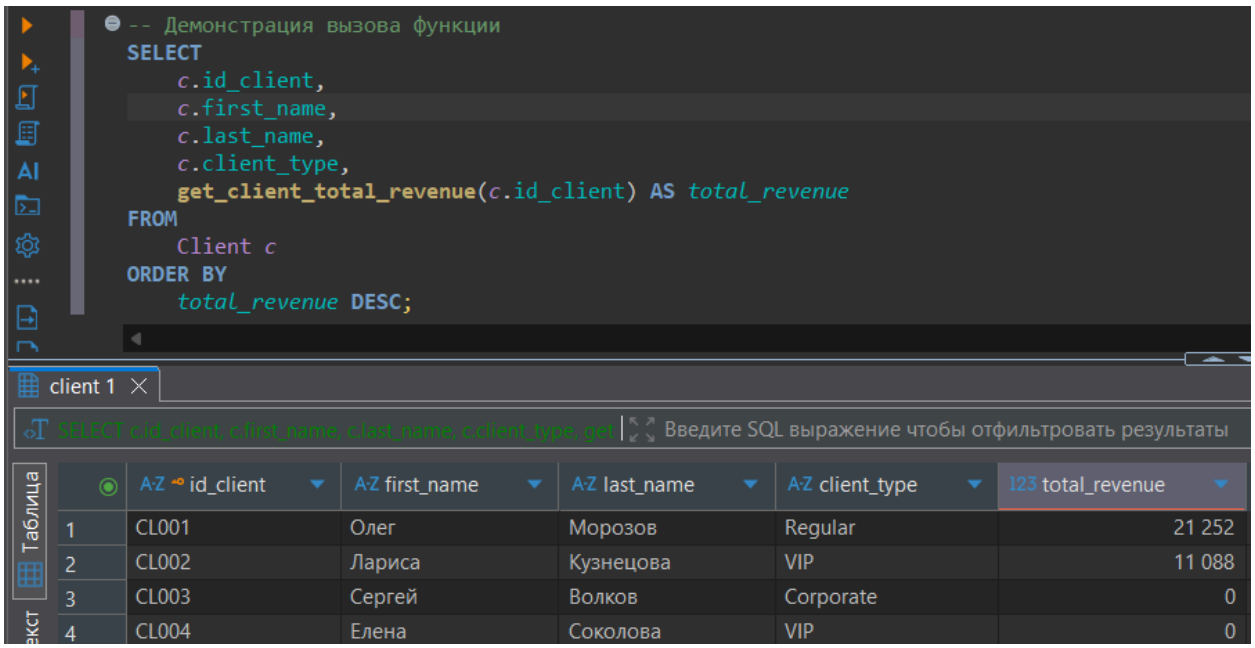
Материализованное представление TotalRentalsByClient хранит физическую копию данных для быстрого доступа к аналитике по арендной активности клиентов. Команда REFRESH MATERIALIZED VIEW обновляет данные при изменениях в базовых таблицах Booking.

Задан пользовательской функции

```
-- создает механизм для расчета общей суммы, потраченной конкретным клиентом на аренду автомобилей
CREATE OR REPLACE FUNCTION get_client_total_revenue(p_client_id VARCHAR(10))
RETURNS DECIMAL(12, 2) AS $$
DECLARE
    total_revenue DECIMAL(12, 2);
    client_exists BOOLEAN;
BEGIN
    -- Проверяем существование клиента
    SELECT EXISTS(SELECT 1 FROM Client WHERE id_client = p_client_id) INTO client_exists;

    IF NOT client_exists THEN
        RAISE EXCEPTION 'Client with ID % does not exist.', p_client_id;
    END IF;

    -- Вычисляем общую сумму потраченную клиентом
    SELECT
        COALESCE(SUM(i.total_amount), 0.00)
    INTO total_revenue
    FROM Booking AS b
    JOIN Invoice AS i ON b.id_invoice = i.id_invoice
    WHERE b.id_client = p_client_id AND i.payment_status = 'Paid';

    RETURN total_revenue;
END;
$$ LANGUAGE plpgsql;
```

Name	Value
Execute time	0.034s
Start time	Thu Nov 06 01:37:05 MSK 2025
Finish time	Thu Nov 06 01:37:30 MSK 2025

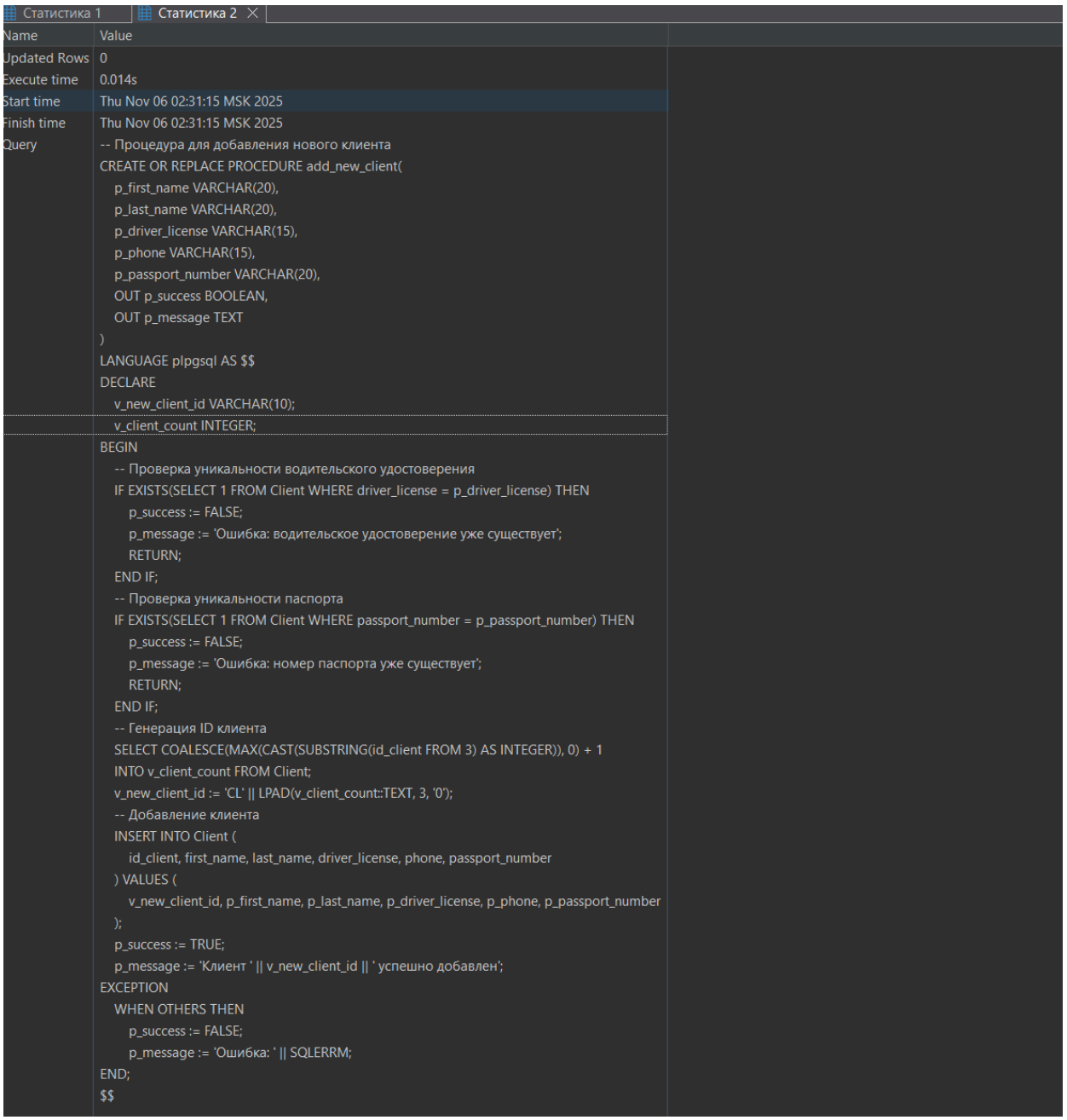
Query	CREATE OR REPLACE FUNCTION get_client_total_revenue(p_client_id VARCHAR(10))
	RETURNS DECIMAL(12, 2) AS $$
	DECLARE
	  total_revenue DECIMAL(12, 2);
	  client_exists BOOLEAN;
	BEGIN
	  -- Проверяем существование клиента
	  SELECT EXISTS(SELECT 1 FROM Client WHERE id_client = p_client_id) INTO client_exists;
	  IF NOT client_exists THEN
	  RAISE EXCEPTION 'Client with ID % does not exist.', p_client_id;
	  END IF;
	  -- Вычисляем общую сумму потраченную клиентом
	  SELECT
	  COALESCE(SUM(i.total_amount), 0.00)
	  INTO total_revenue
	  FROM Booking AS b
	  JOIN Invoice AS i ON b.id_invoice = i.id_invoice
	  WHERE b.id_client = p_client_id AND i.payment_status = 'Paid';
	  RETURN total_revenue;
	END;
	$$ LANGUAGE plpgsql



Функция get_client_total_revenue **инкапсулирует сложный запрос для расчета общей суммы, потраченной конкретным клиентом на аренду автомобилей. Она выполняет проверку существования клиента в базе данных и возвращает скалярное значение - общую выручку по указанному клиенту**.

Задание №7: Разработка хранимой процедуры

```
 -- Процедура для добавления нового клиента
-- Процедура для добавления нового клиента
CREATE OR REPLACE PROCEDURE add_new_client(
    p_first_name VARCHAR(20),
    p_last_name VARCHAR(20),
    p_driver_license VARCHAR(15),
    p_phone VARCHAR(15),
    p_passport_number VARCHAR(20),
    OUT p_success BOOLEAN,
    OUT p_message TEXT
)
LANGUAGE plpgsql AS $$
DECLARE
    v_new_client_id VARCHAR(10);
    v_client_count INTEGER;
BEGIN
    -- Проверка уникальности водительского удостоверения
    IF EXISTS(SELECT 1 FROM Client WHERE driver_license = p_driver_license) THEN
        p_success := FALSE;
        p_message := 'Ошибка: водительское удостоверение уже существует';
        RETURN;
    END IF;

    -- Проверка уникальности паспорта
    IF EXISTS(SELECT 1 FROM Client WHERE passport_number = p_passport_number) THEN
        p_success := FALSE;
        p_message := 'Ошибка: номер паспорта уже существует';
        RETURN;
    END IF;

    -- Генерация ID клиента
    SELECT COALESCE(MAX(CAST(SUBSTRING(id_client FROM 3) AS INTEGER)), 0) + 1
    INTO v_client_count FROM Client;
    v_new_client_id := 'CL' || LPAD(v_client_count::TEXT, 3, '0');

    -- Добавление клиента
    INSERT INTO Client (
        id_client, first_name, last_name, driver_license, phone, passport_number
    ) VALUES (
        v_new_client_id, p_first_name, p_last_name, p_driver_license, p_phone,
p_passport_number
    );

    p_success := TRUE;
    p_message := 'Клиент ' || v_new_client_id || ' успешно добавлен';

EXCEPTION
    WHEN OTHERS THEN
        p_success := FALSE;
        p_message := 'Ошибка: ' || SQLERRM;
END;
$$;
```



Name	Value
Updated Rows	0
Execute time	0.014s
Start time	Thu Nov 06 02:31:15 MSK 2025
Finish time	Thu Nov 06 02:31:15 MSK 2025

Query	-- Процедура для добавления нового клиента
	CREATE OR REPLACE PROCEDURE add_new_client(
	p_first_name VARCHAR(20),
	p_last_name VARCHAR(20),
	p_driver_license VARCHAR(15),
	p_phone VARCHAR(15),
	p_passport_number VARCHAR(20),
	OUT p_success BOOLEAN,
	OUT p_message TEXT
	)
	LANGUAGE plpgsql AS $$
	DECLARE
	v_new_client_id VARCHAR(10);
	v_client_count INTEGER;
	BEGIN
	-- Проверка уникальности водительского удостоверения
	IF EXISTS(SELECT 1 FROM Client WHERE driver_license = p_driver_license) THEN
	p_success := FALSE;
	p_message := 'Ошибка: водительское удостоверение уже существует';
	RETURN;
	END IF;
	-- Проверка уникальности паспорта
	IF EXISTS(SELECT 1 FROM Client WHERE passport_number = p_passport_number) THEN
	p_success := FALSE;
	p_message := 'Ошибка: номер паспорта уже существует';
	RETURN;
	END IF;
	-- Генерация ID клиента
	SELECT COALESCE(MAX(CAST(SUBSTRING(id_client FROM 3) AS INTEGER)), 0) + 1
	INTO v_client_count FROM Client;
	v_new_client_id := 'CL' \|\| LPAD(v_client_count::TEXT, 3, '0');
	-- Добавление клиента
	INSERT INTO Client (
	id_client, first_name, last_name, driver_license, phone, passport_number
	) VALUES (
	v_new_client_id, p_first_name, p_last_name, p_driver_license, p_phone, p_passport_number
	);
	p_success := TRUE;
	p_message := 'Клиент ' \|\| v_new_client_id \|\| ' успешно добавлен';
	EXCEPTION
	WHEN OTHERS THEN
	p_success := FALSE;
	p_message := 'Ошибка: ' \|\| SQLERRM;
	END;
	$$

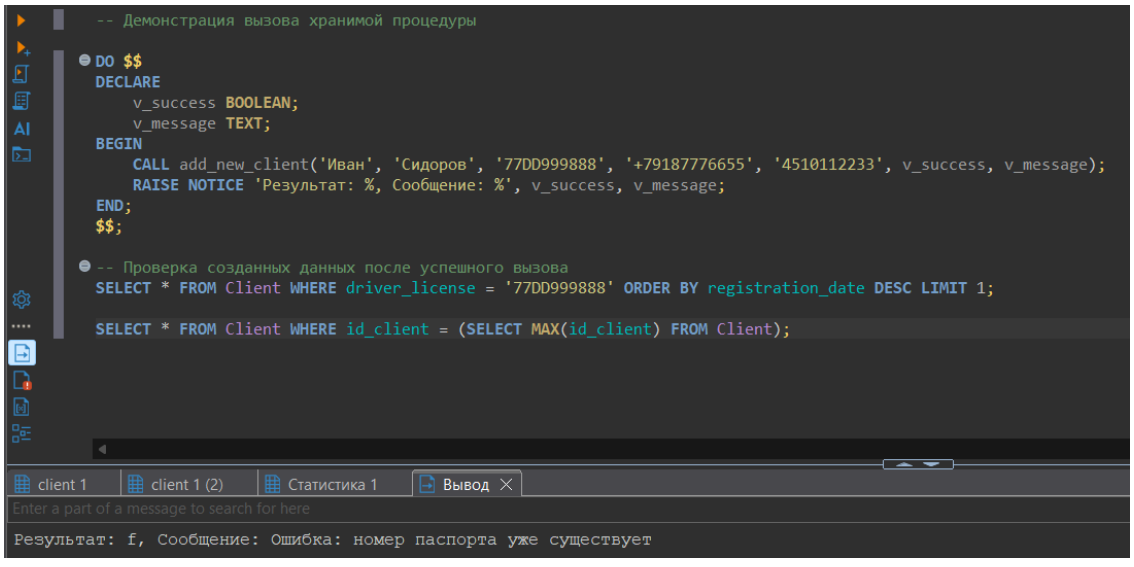
Процедура process_new_client инкапсулирует бизнес-логику добавления нового клиента

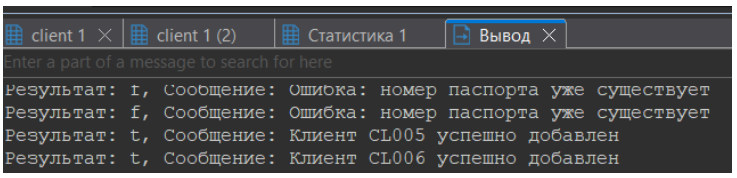
с проверками уникальности водительского удостоверения и паспортных данных.

Использует выходные параметры для возврата статуса операции.

Задание №8: Демонстрация вызова хранимой процедуры

```
-- 1. Успешный вызов процедуры
DO $$

DECLARE

    v_success BOOLEAN;

    v_message TEXT;


BEGIN

    CALL add_new_client('Иван', 'Сидоров', '7700999288', '+79187776455',
'4510212299', v_success, v_message);

    RAISE NOTICE 'Результат: %, Сообщение: %', v_success, v_message;


END;

$$;
```





Первый вызов демонстрирует успешное добавление клиента, второй - срабатывание проверки на уникальность данных.

Процедура возвращает детальные сообщения о результате операции через выходные параметры.

Вывод: в ходе работы освоены представления и хранимые процедуры в PostgreSQL. Научены создавать модифицируемые, немодифицируемые и материализованные представления, различать функции и процедуры, использовать PL/pgSQL с условной логикой и обработкой ошибок. Освоен вызов процедур через CALL и работа с их результатами.