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 № 4

по дисциплине «Разработка баз данных»

Студент группы *ИНБО-12-23. Албахтин И.В.*

(подпись)

Ассистент *Брайловский А.В.*

(подпись)

Москва 2025 г.

ПРАКТИЧЕСКАЯ РАБОТА №4. АНАЛИТИЧЕСКИЕ ЗАПРОСЫ: ОКОННЫЕ ФУНКЦИИ И ПОСТРОЕНИЕ СВОДНЫХ ТАБЛИЦ

Цель:

Целью данной практической работы является формирование у студентов углубленных навыков работы со сложными аналитическими запросами в СУБД PostgreSQL.

Постановка задачи:

Для выполнения практической работы необходимо последовательно выполнить четыре задачи, используя собственную базу данных. Все примеры в данном документе основаны на демонстрационной базе данных «Аптека», содержащей таблицы manufacturers (производители), medicines (лекарства) и sales (продажи).

Ваша задача — адаптировать каждую из поставленных задач к логической структуре и предметной области вашей базы данных. Приведенные ниже формулировки и последующие примеры кода служат шаблоном для понимания, какой тип аналитического запроса требуется составить.

Задание №1: использование ранжирующих функций

Для каждой основной «родительской» сущности в вашей БД *(например, производитель, категория товара, автор)* определить **три** наиболее значимых по некоторому **числовому признаку** дочерних сущности *(например, три самых дорогих товара, три самые популярные книги по количеству продаж)*.

В результирующей таблице должны быть указаны идентификатор группы, идентификатор дочерней сущности, её числовой признак и ранг. Для расчёта ранга использовать функцию **RANK**() или **DENSE_RANK**().

Задание №2: использование агрегатных оконных функций

Для ключевой сущности, имеющей **транзакции по времени** *(например, товар, услуга)*, рассчитать **нарастающий итог** *(кумулятивную сумму)* по

некоторому показателю *(например, объем продаж, количество заказов)* с разбивкой по временным периодам *(месяцам или годам)*.

Отчёт должен содержать идентификатор сущности (id/название/…), временной период, сумму за период и кумулятивную сумму.

Задание №3: использование функции смещения

Провести сравнительный анализ общих показателей **по периодам**.

Для **каждого периода** *(например, месяца)*, начиная со второго, необходимо вывести **общий показатель** за **текущий** период и аналогичный показатель за **предыдущий** период в одной строке. Это позволит наглядно оценить динамику.

Необходимо использовать функцию **LAG**().

Задание №4: построение сводной таблицы

Создать сводный отчет, который агрегирует некоторый числовой показатель для основной сущности по категориям, представленным в виде столбцов.

ВЫПОЛНЕНИЕ ПРАКТИЧЕСКОЙ РАБОТЫ

Таблица 1. Таблица worker (Сотрудник)

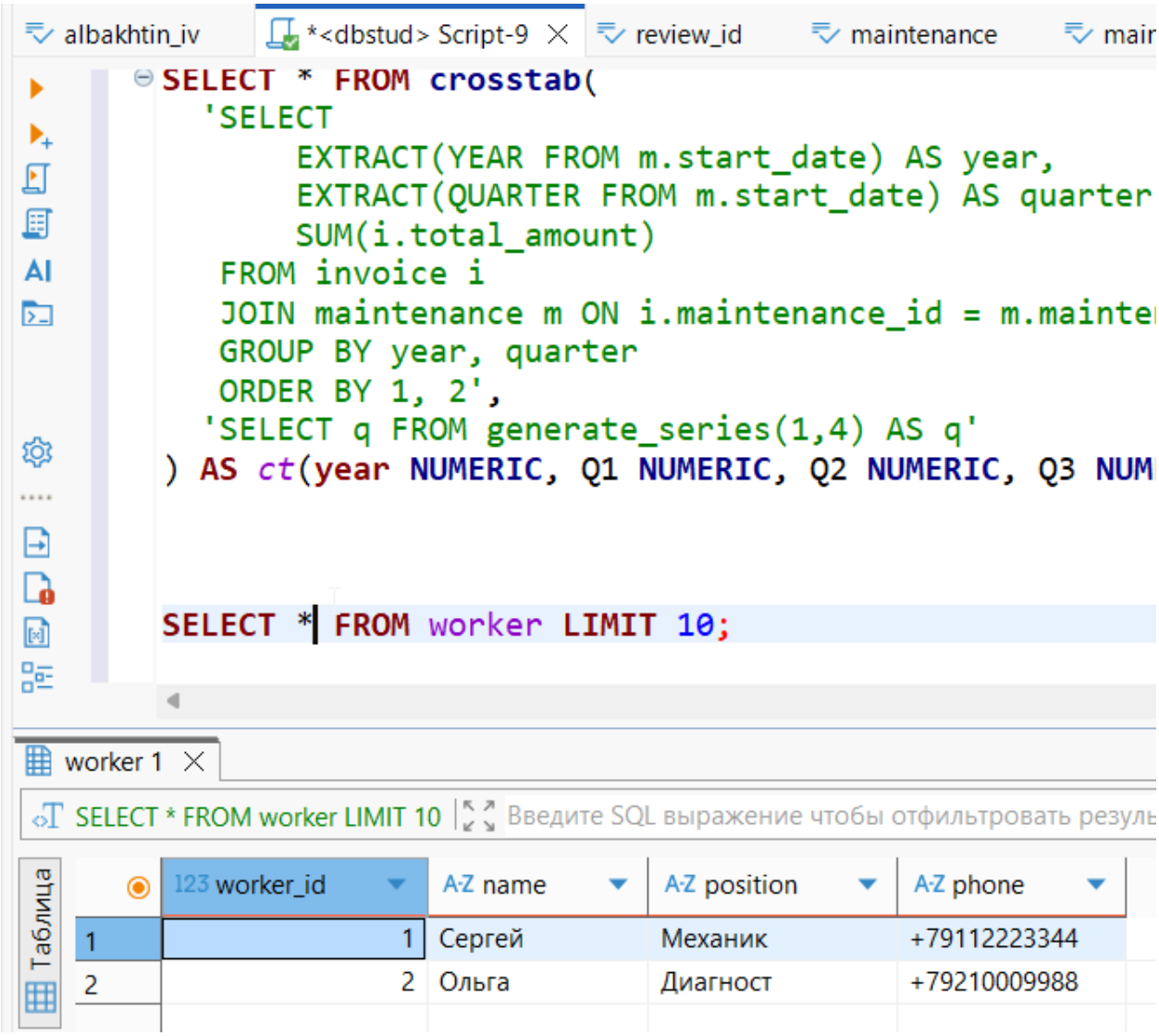
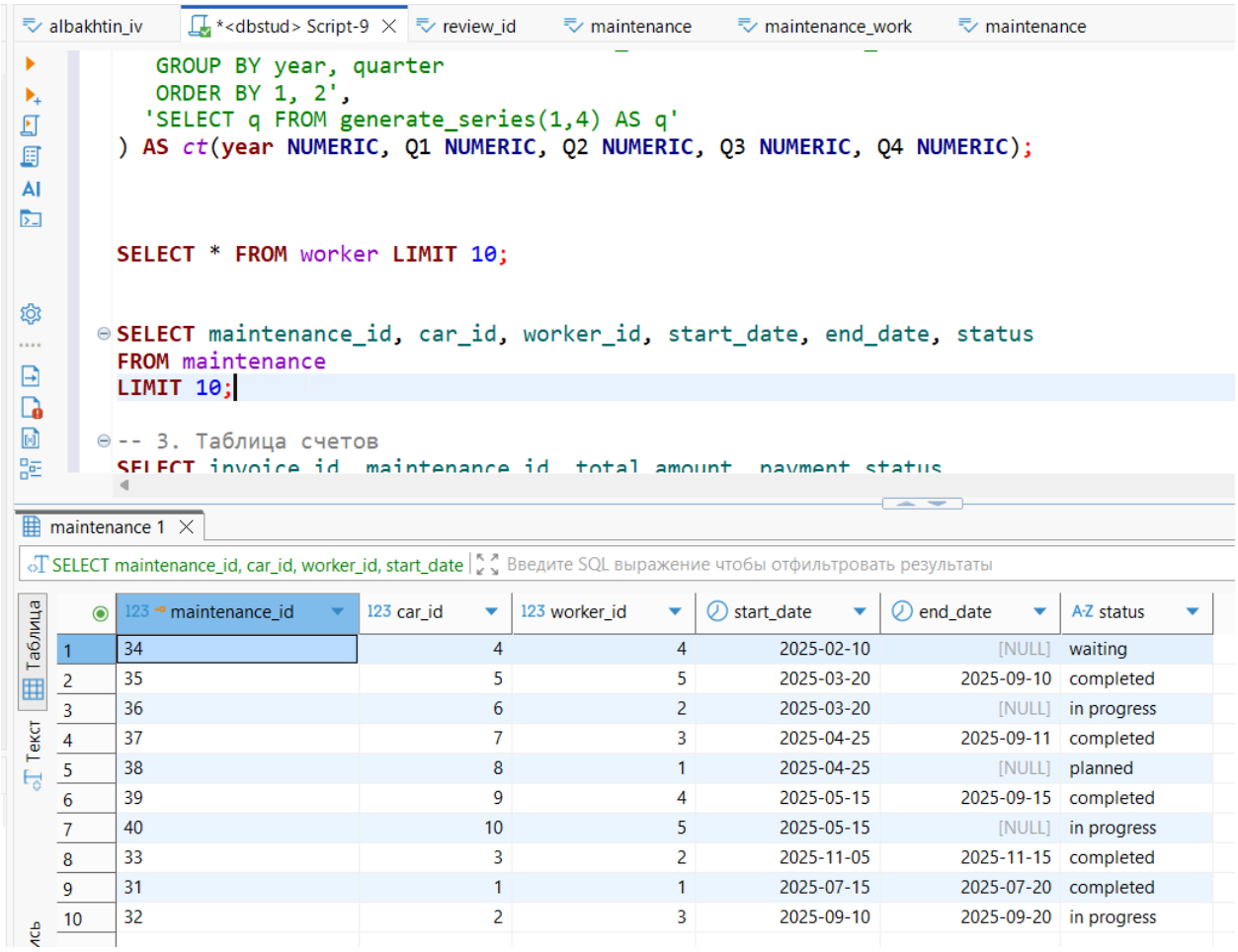


Таблица 2. Таблица maintenance (ТО)



albakhtin_iv | *<dbstud> Script-9 | review_id | maintenance | maintenance_work | maintenance

```
    GROUP BY year, quarter
    ORDER BY 1, 2',
   'SELECT q FROM generate_series(1,4) AS q'
) AS ct(year NUMERIC, Q1 NUMERIC, Q2 NUMERIC, Q3 NUMERIC, Q4 NUMERIC);



SELECT * FROM worker LIMIT 10;


SELECT maintenance_id, car_id, worker_id, start_date, end_date, status
FROM maintenance
LIMIT 10;

-- 3. Таблица счетов
SELECT invoice_id, maintenance_id, total_amount, payment_status
```

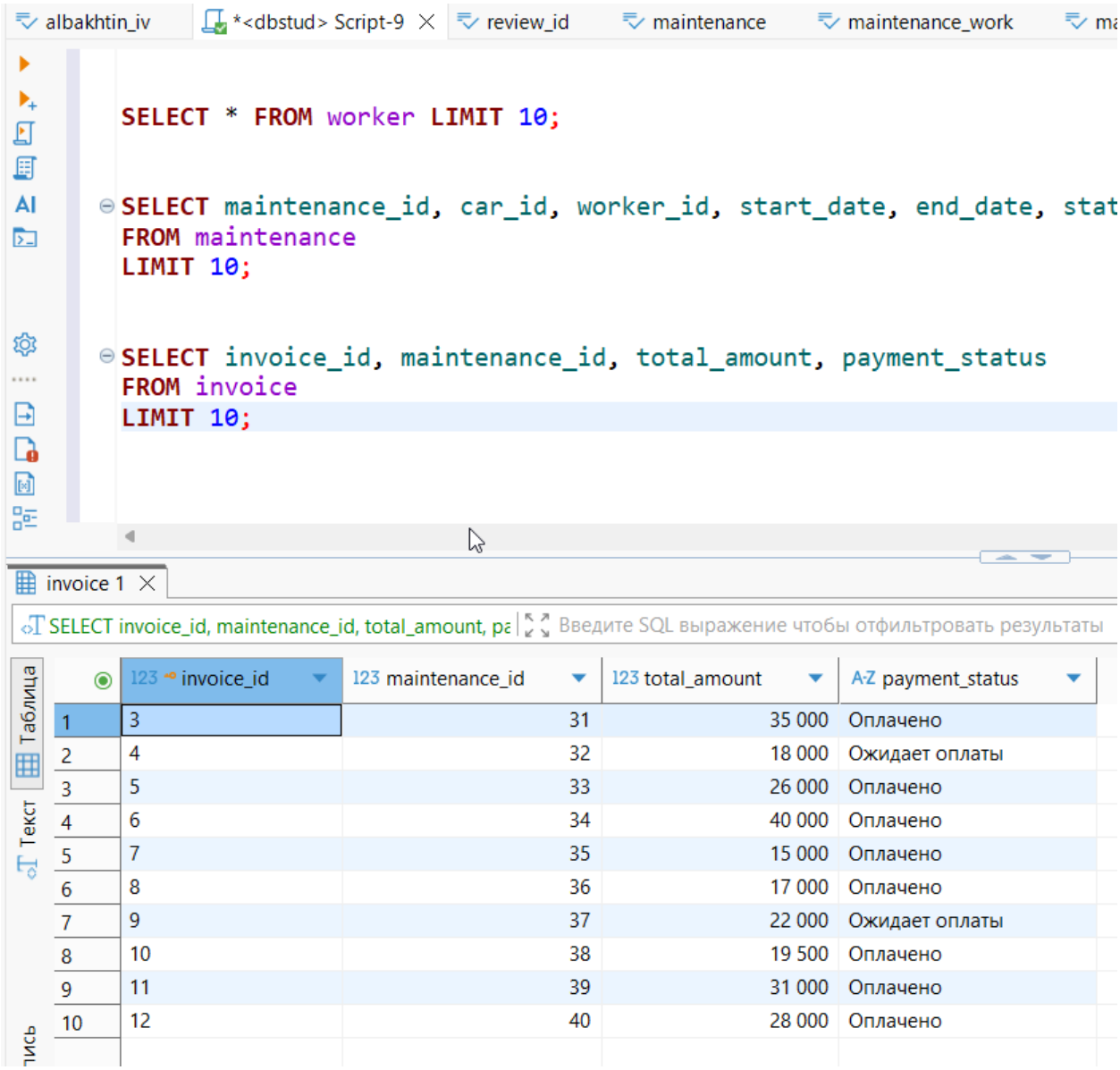
maintenance 1

SELECT maintenance_id, car_id, worker_id, start_date | Введите SQL выражение чтобы отфильтровать результаты

Таблица | Текст

	maintenance_id	car_id	worker_id	start_date	end_date	status
1	34	4	4	2025-02-10	[NULL]	waiting
2	35	5	5	2025-03-20	2025-09-10	completed
3	36	6	2	2025-03-20	[NULL]	in progress
4	37	7	3	2025-04-25	2025-09-11	completed
5	38	8	1	2025-04-25	[NULL]	planned
6	39	9	4	2025-05-15	2025-09-15	completed
7	40	10	5	2025-05-15	[NULL]	in progress
8	33	3	2	2025-11-05	2025-11-15	completed
9	31	1	1	2025-07-20	2025-07-20	completed
10	32	2	3	2025-09-10	2025-09-20	in progress

Таблица 3. Таблица invoice (счета)



Задание 1. Создание модифицируемого представления

albakhtin_iv | *<postgres> Script | invoice

```
    JOIN cat_tree ct ON c.parent_id = ct.category_id
)
SELECT
    LPAD('', level * 4, ' ') || name AS hierarchy,
    level
FROM cat_tree;


CREATE OR REPLACE VIEW paid_invoices AS
SELECT
    invoice_id,
    maintenance_id,
    total_amount,
    payment_status
FROM invoice
WHERE payment_status = 'Оплачено';
```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0.014s
Start time	Sun Nov 02 14:13:52 MSK 2025
Finish time	Sun Nov 02 14:13:52 MSK 2025
Query	CREATE OR REPLACE VIEW paid_invoices AS
	SELECT
	invoice_id,
	maintenance_id,
	total_amount,
	payment_status
	FROM invoice
	WHERE payment_status = 'Оплачено'

Рисунок 1 – Все оплаченные счета с фильтром payment_status = 'Оплачено' часть 1

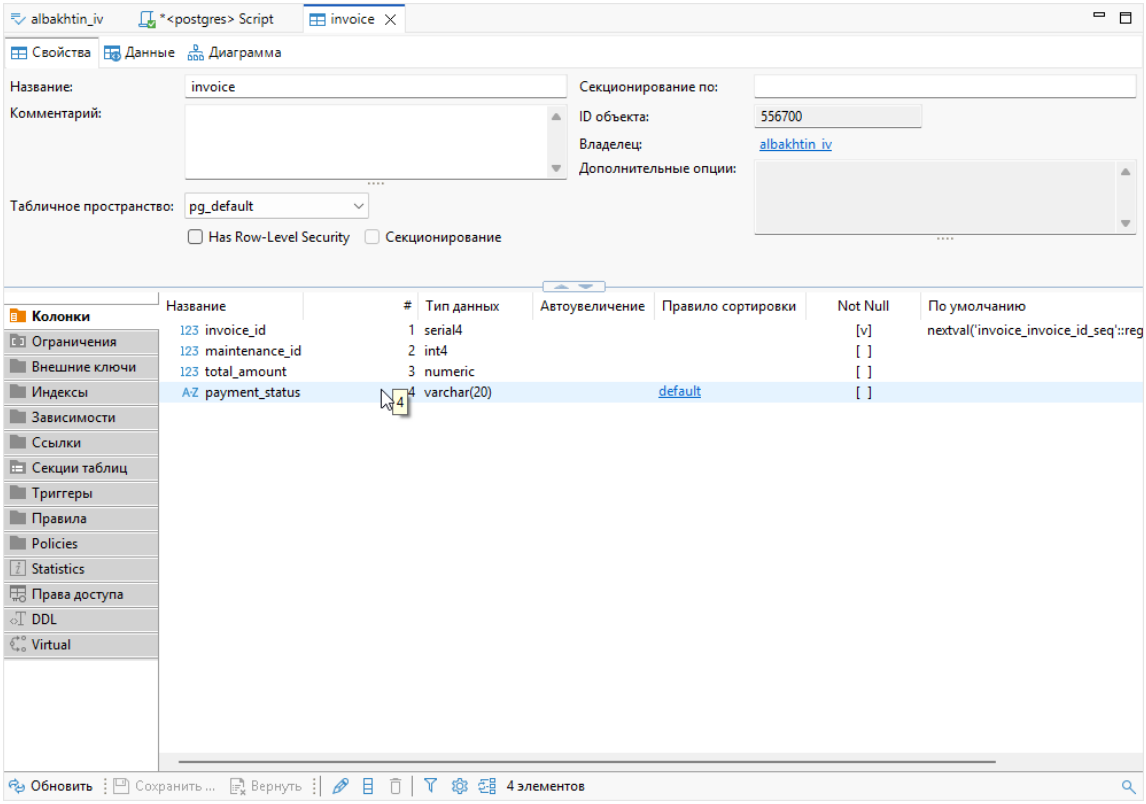


Рисунок 2 - Все оплаченные счета с фильтром payment_status = 'Оплачено' часть 2

Задание 2. Модификация данных через представление

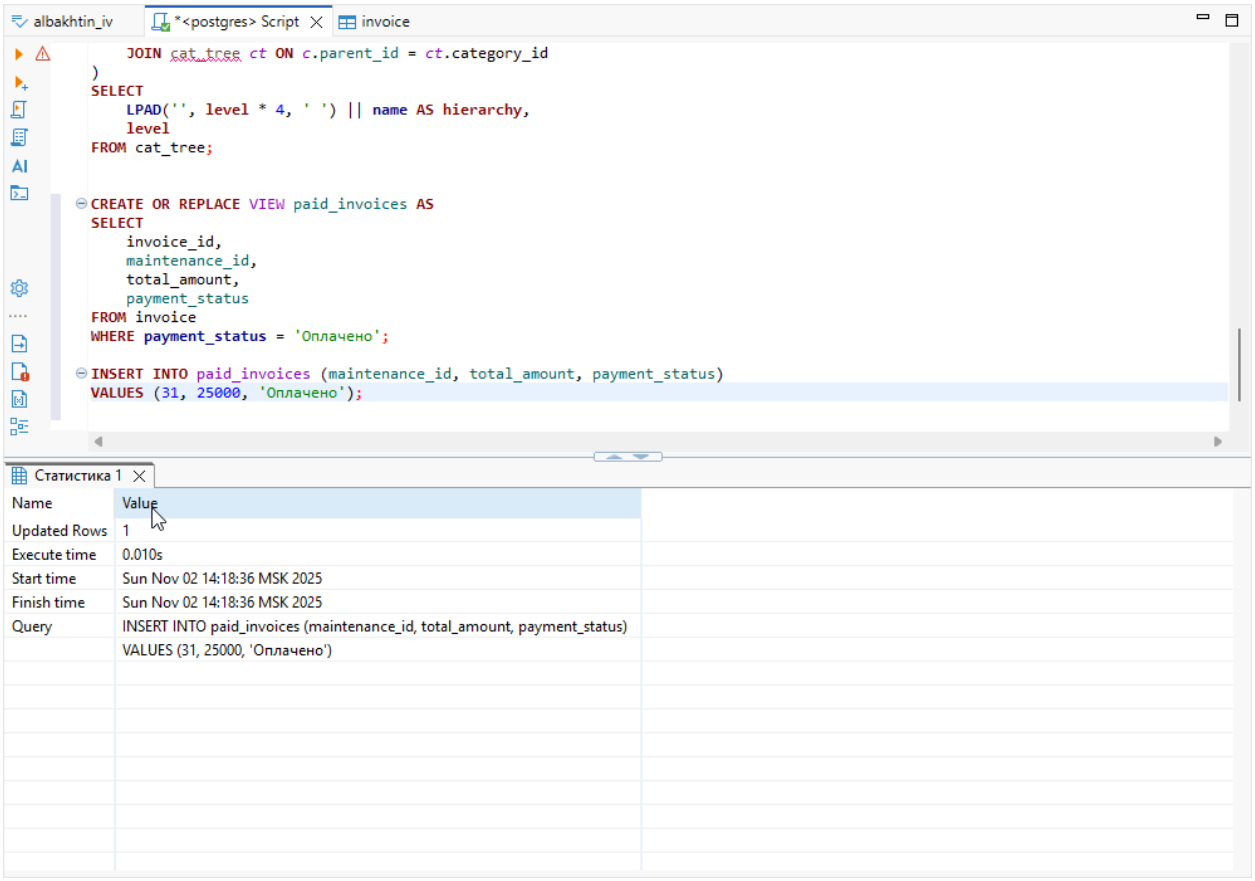


Рисунок 3 – Добавление новой записи через представление

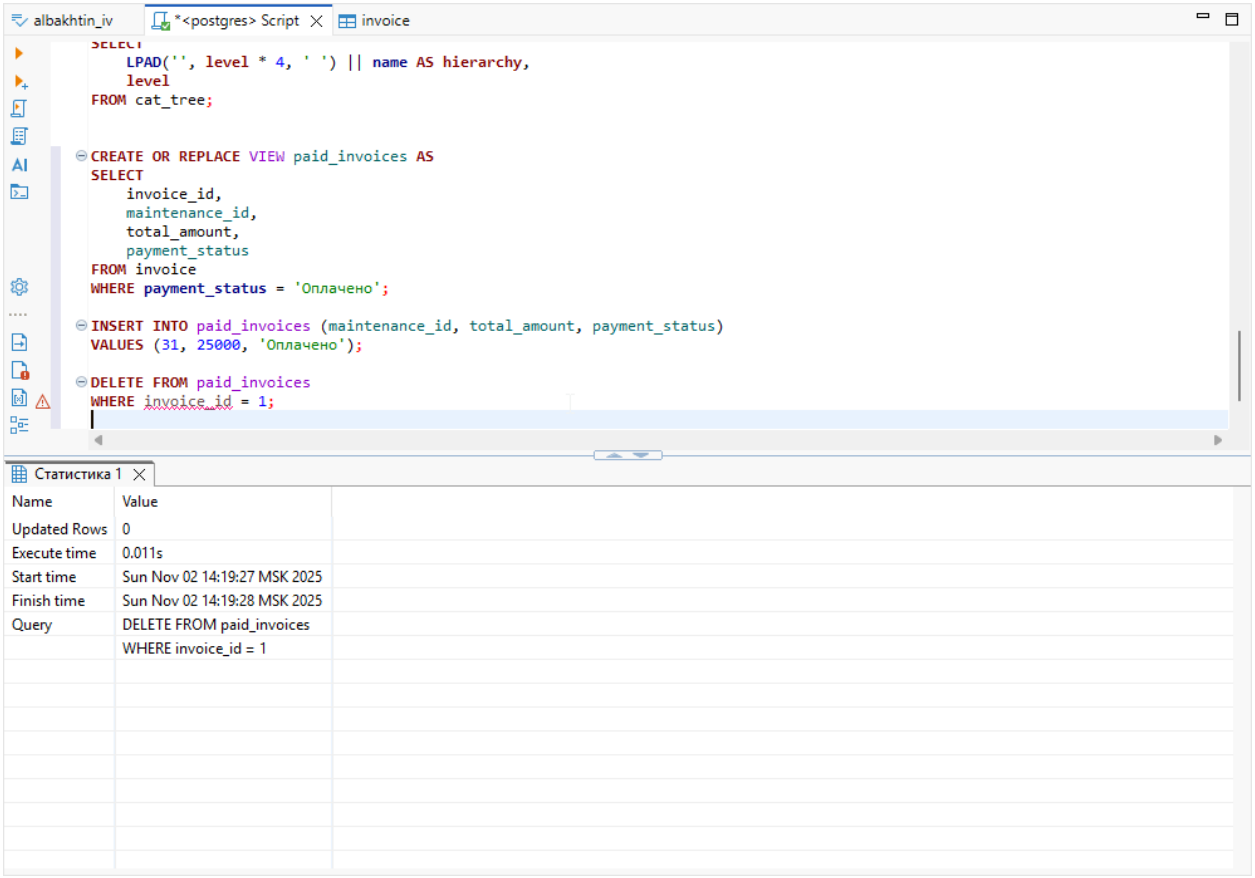


Рисунок 4 - Удаление записи через представление

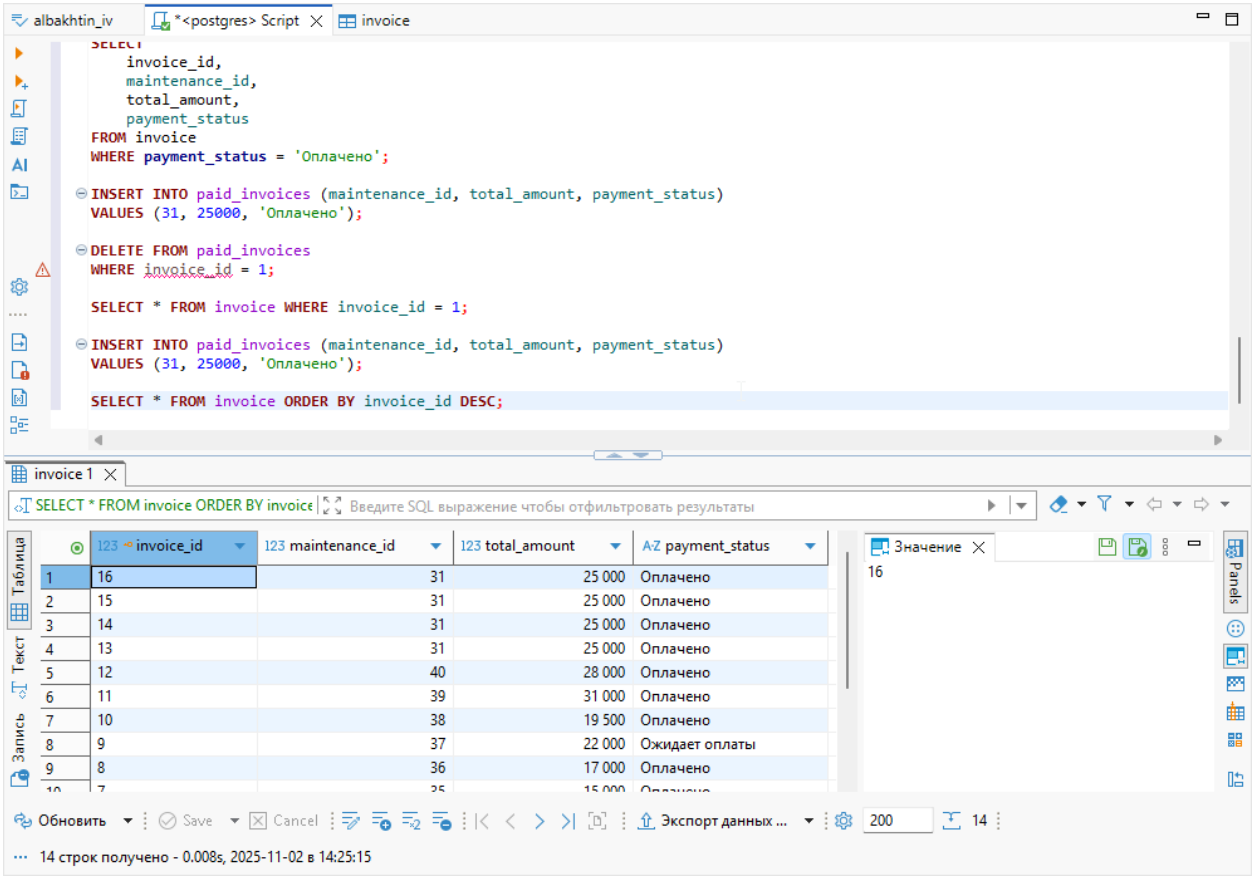


Рисунок 5 - Проверка

Задание 3. Создание немодифицируемого аналитического представления

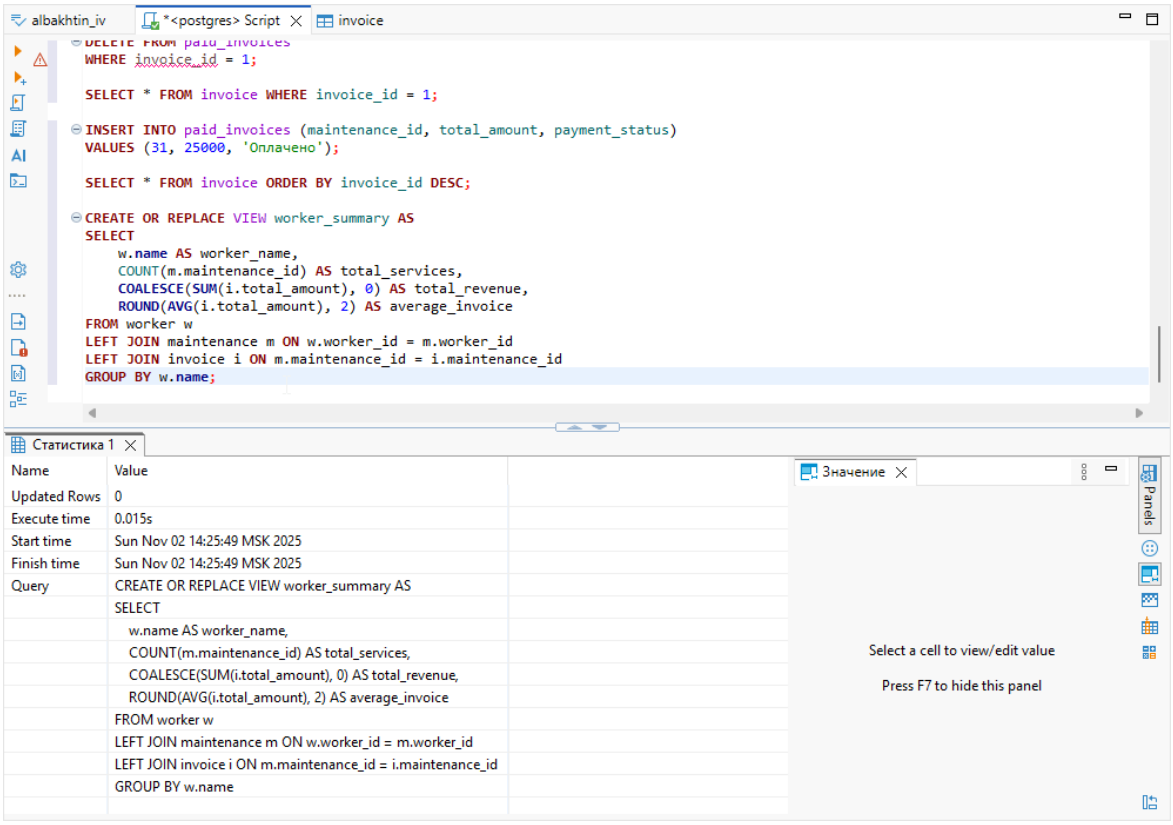


Рисунок 6 – Создание сводки по каждому сотруднику

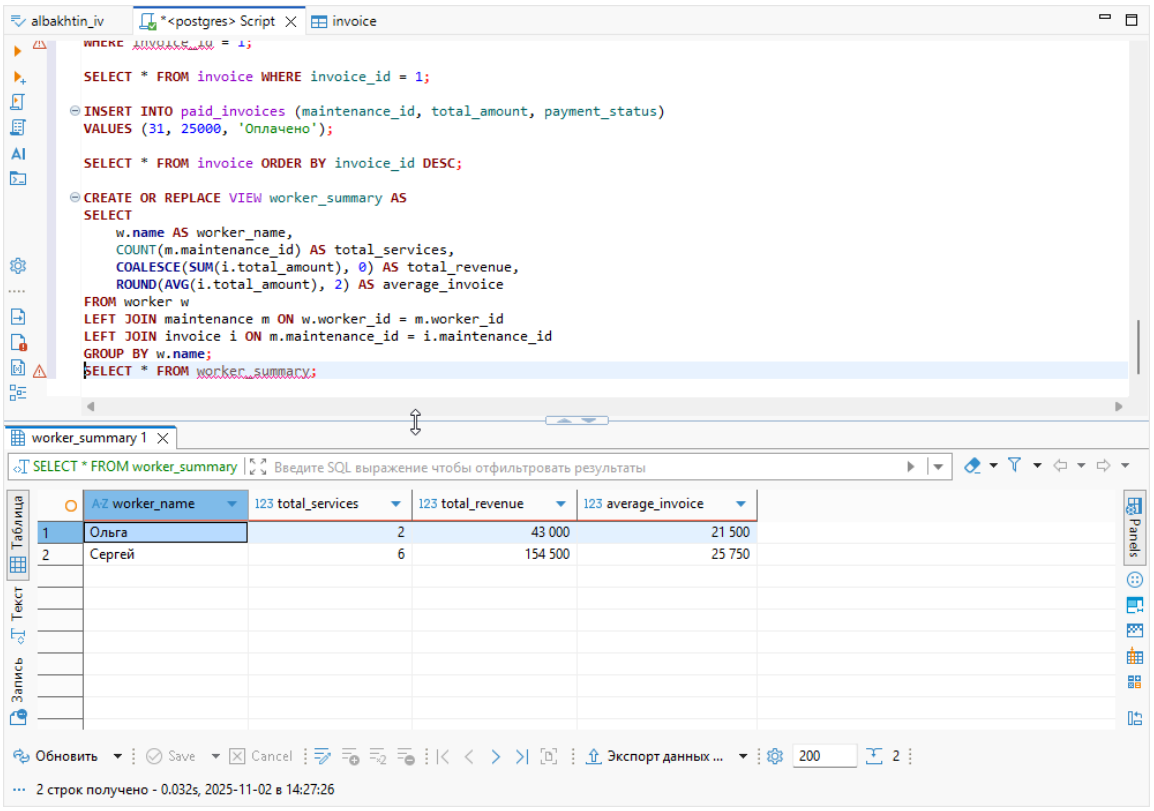


Рисунок 7 - Проверка

Задание 4. Использование аналитического представления в запросах

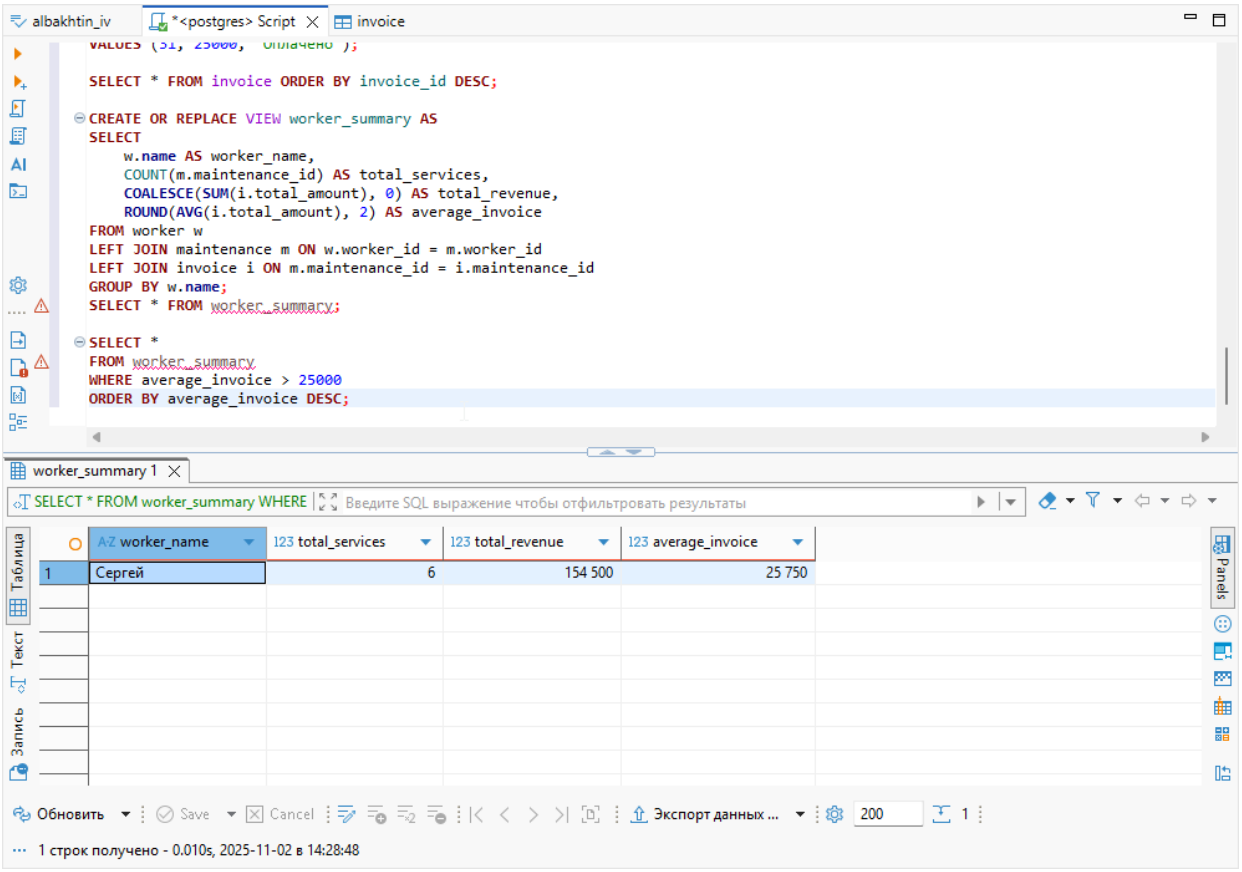


Рисунок 8 - Выведем всех работников, у которых средний счёт больше 25 000

Задание 5. Создание и обновление материализованного Представления

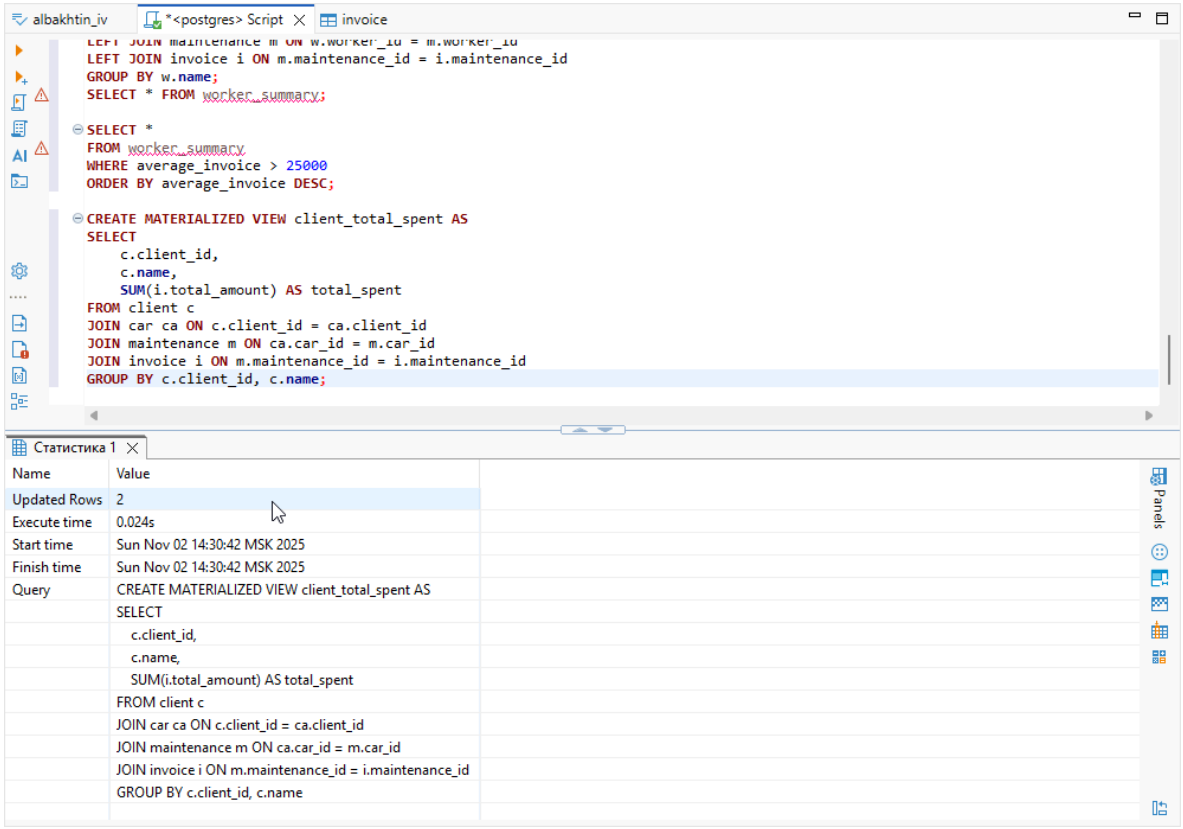


Рисунок 9 - Создадим быстрое сводное представление по клиентам: сколько они потратили на ТО

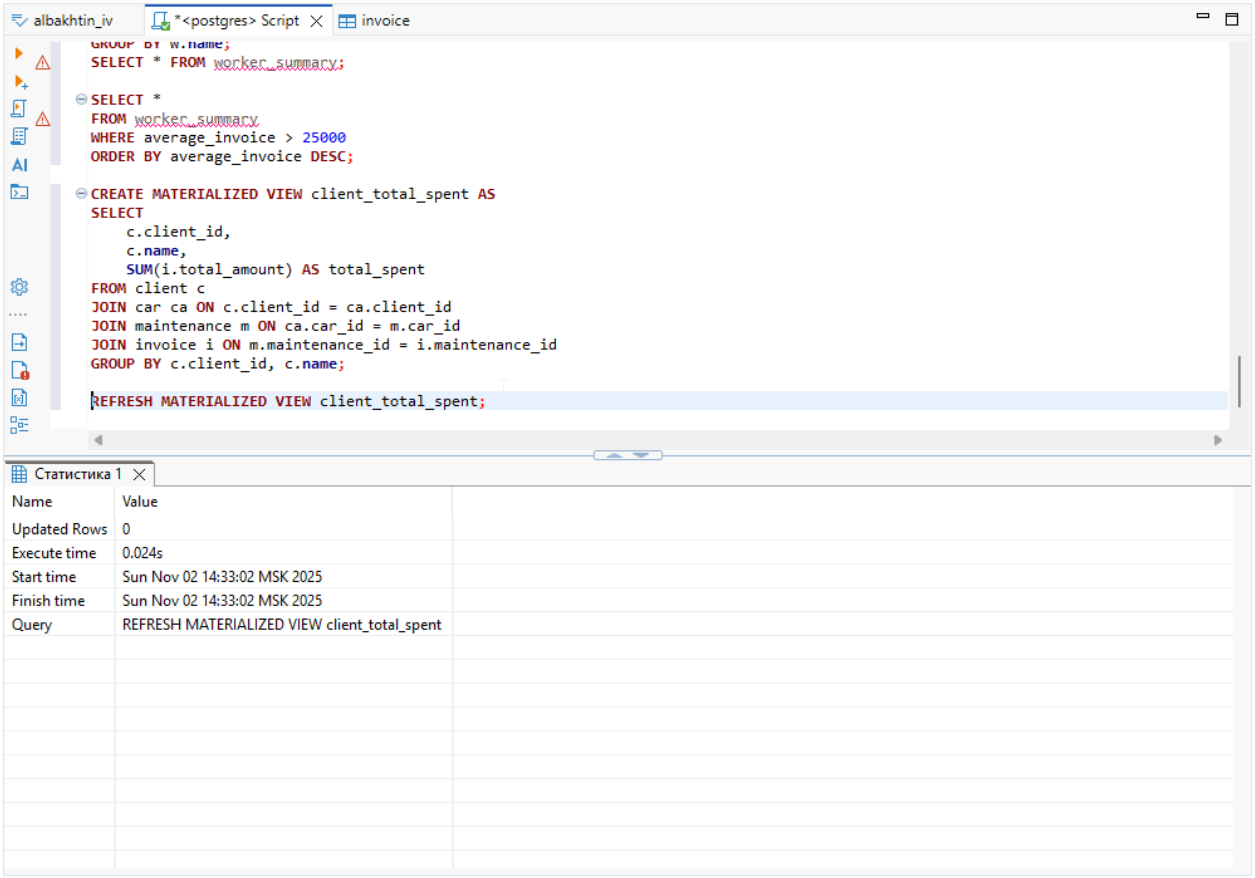


Рисунок 10 – Обновим данные

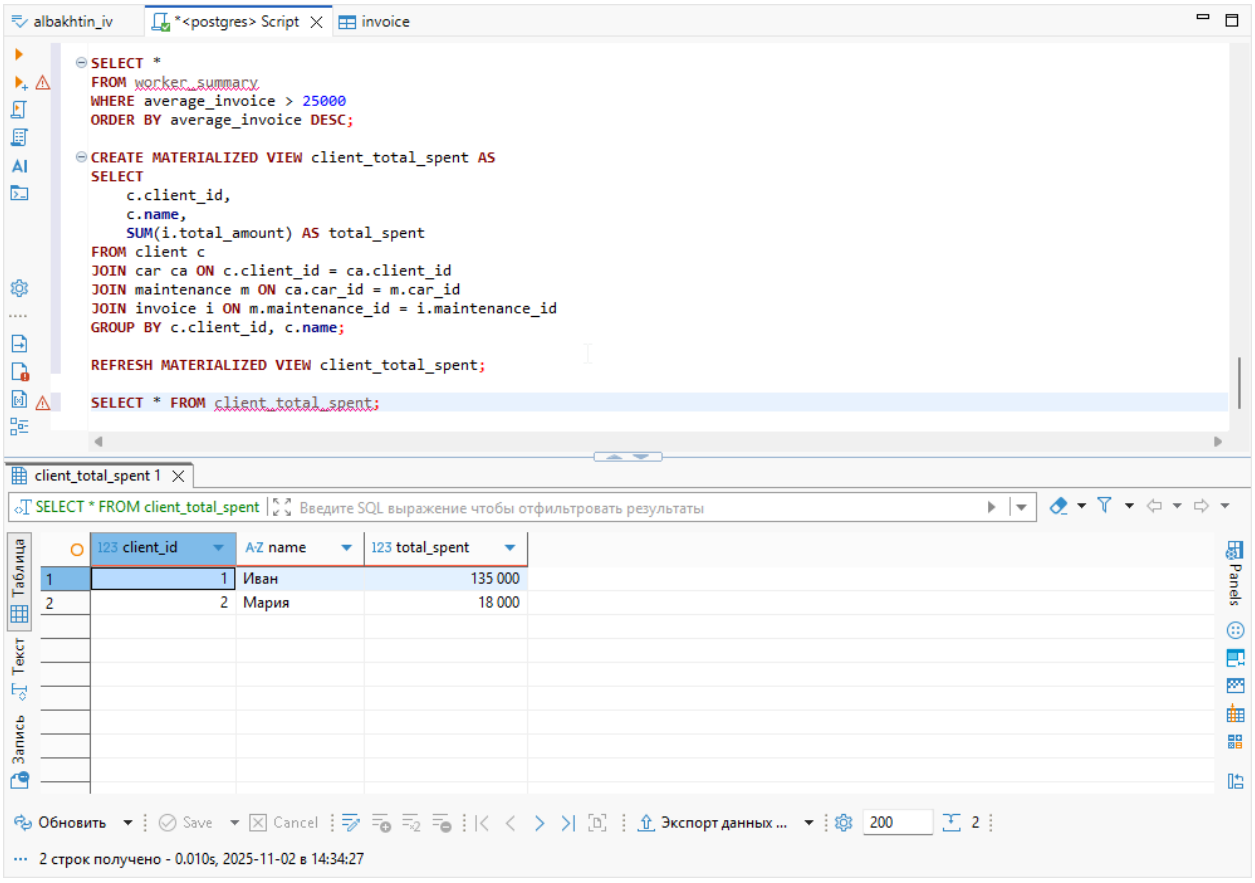


Рисунок 11 – Проверка

Задание 6. Разработка пользовательской функции для аналитических вычислений

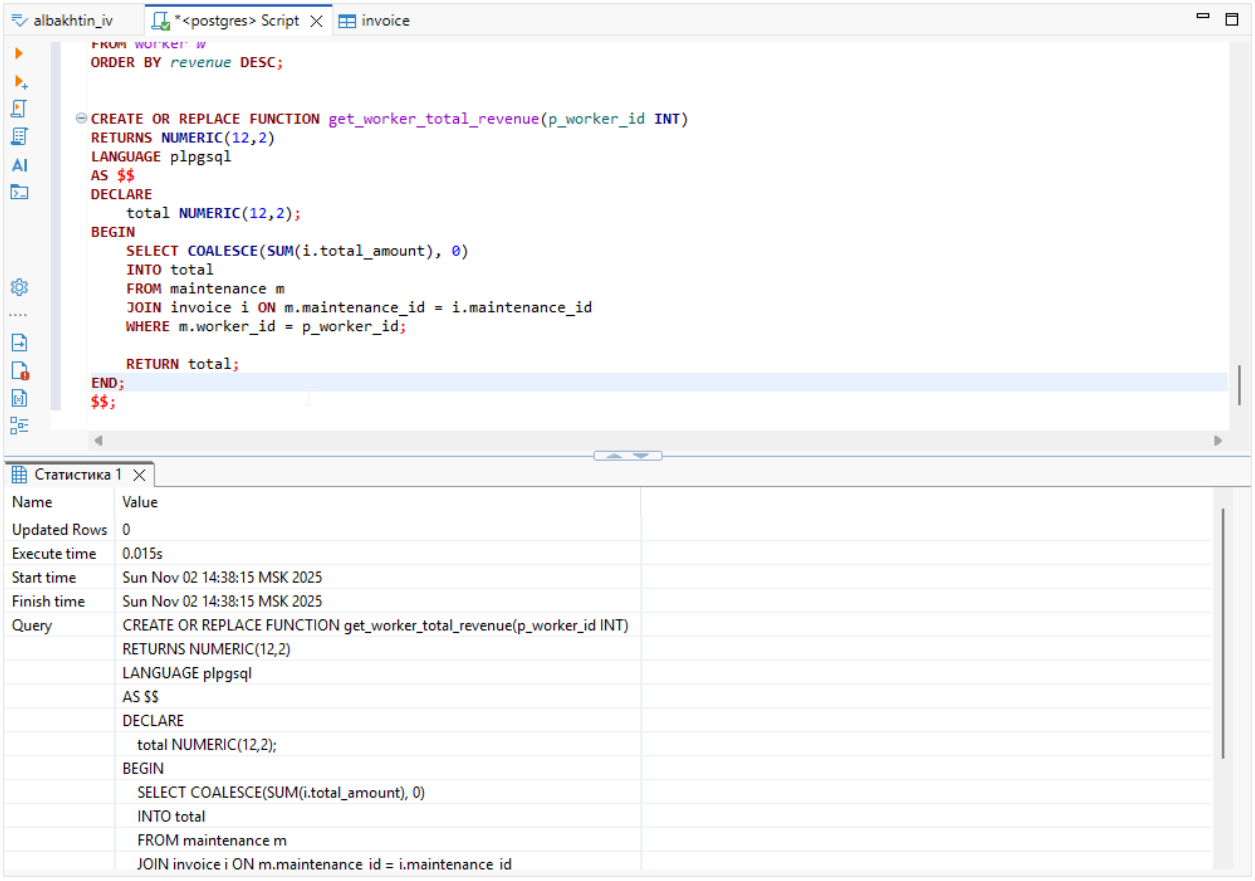


Рисунок 12 –Возвращаем общую сумму по конкретному работнику

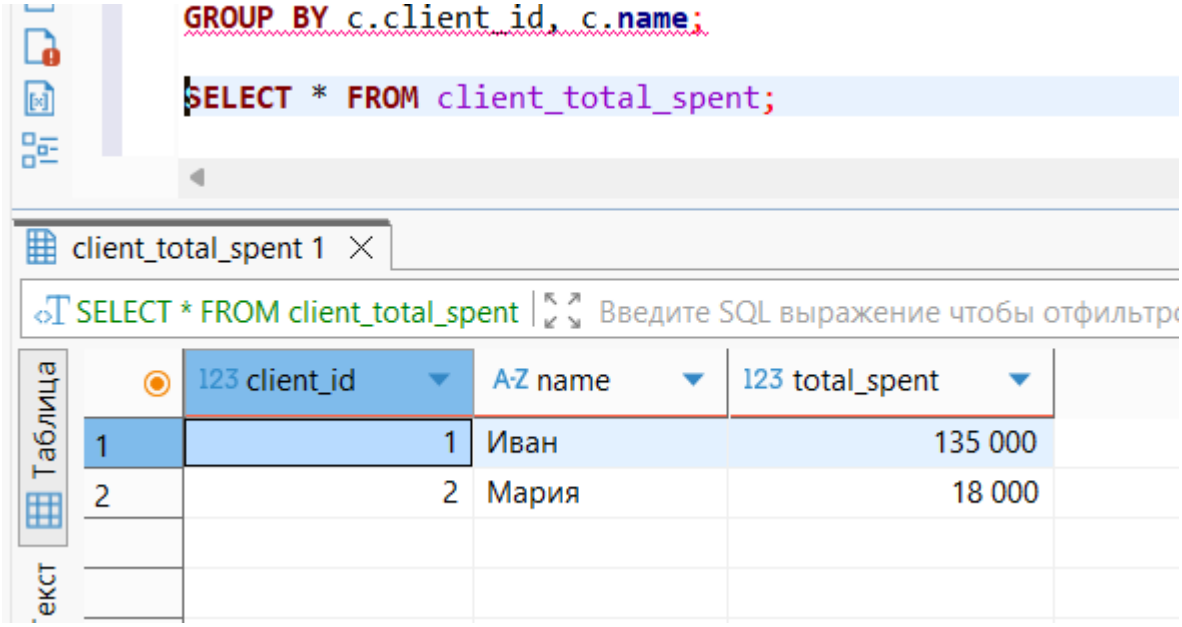


Рисунок 13 - Текущее состояние

```
INSERT INTO invoice (maintenance_id, total_amount, payment_status)
VALUES (32, 7000, 'Оплачено');
```

Статистика 1

Name	Value
Updated Rows	1
Execute time	0,026s
Start time	Sun Dec 21 13:40:57 MSK 2025
Finish time	Sun Dec 21 13:40:58 MSK 2025
Query	INSERT INTO invoice (maintenance_id, total_amount, payment_status) VALUES (32, 7000, 'Оплачено')

Рисунок 14 – Меняем исходные данные

```
INSERT INTO invoice (maintenance_id, total_amount, payment_status)
VALUES (32, 7000, 'Оплачено');

SELECT * FROM client_total_spent;
```

client_total_spent 1

SELECT * FROM client_total_spent | Введите SQL выражение чтобы отфильтровать результат

Таблица

	123 client_id	A-Z name	123 total_spent
1	1	Иван	135 000
2	2	Мария	18 000

Рисунок 15 – Ничего не поменялось

```
REFRESH MATERIALIZED VIEW client_total_spent;
```

Статистика 1

Name	Value
Updated Rows	0
Execute time	0,06s
Start time	Sun Dec 21 13:43:15 MSK 2025
Finish time	Sun Dec 21 13:43:15 MSK 2025
Query	REFRESH MATERIALIZED VIEW client_total_spent

Рисунок 16 – Обновляем представление

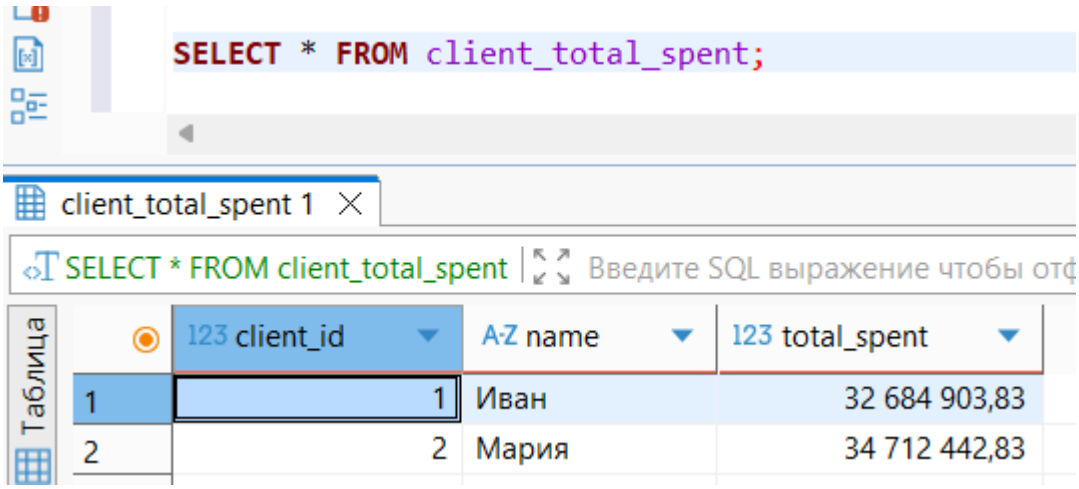


Рисунок 17 – После обновления (данные поменялись слишком сильно, так как переделываю я эту работу после всех последующих)

Задание 7. Разработка хранимой процедуры для выполнения сложной операции

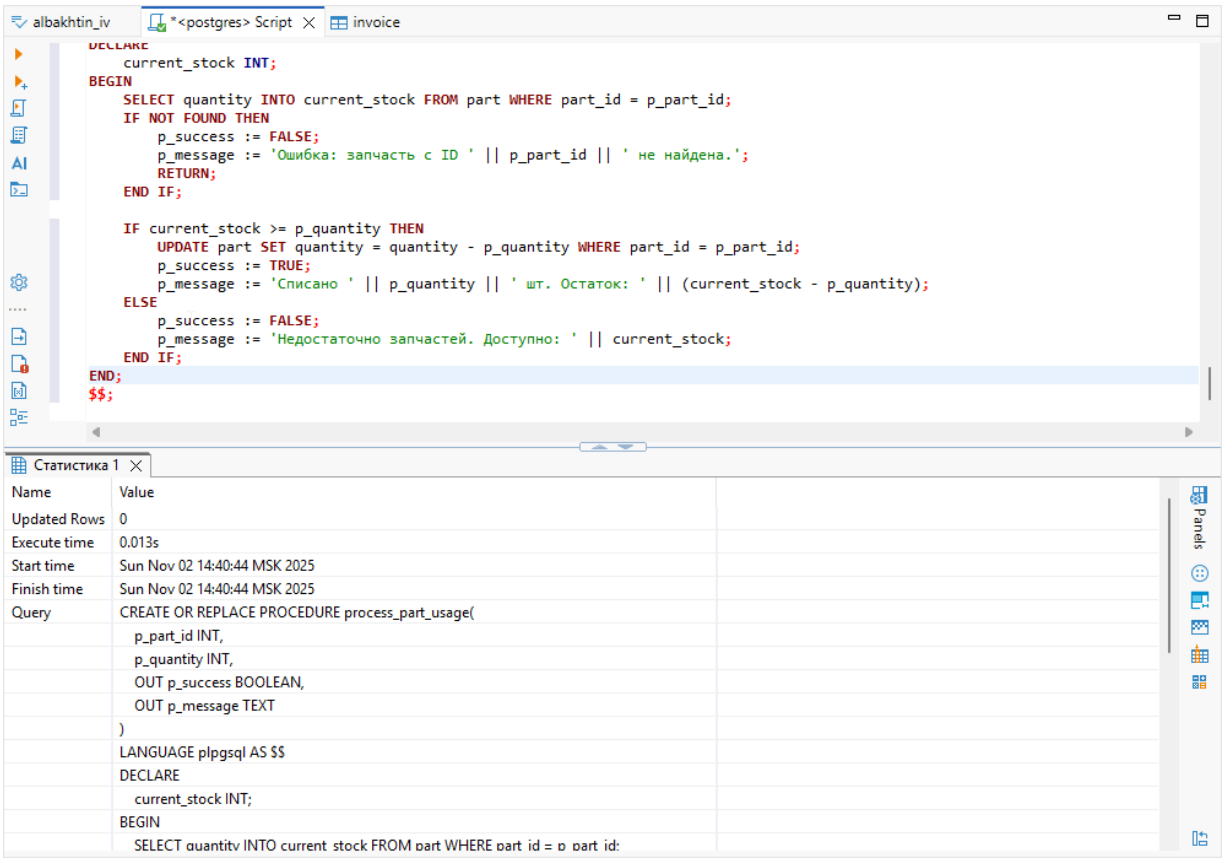


Рисунок 18 – Списываем запас запчастей (part), если их хватает, и записывает в таблицу maintenance_work.

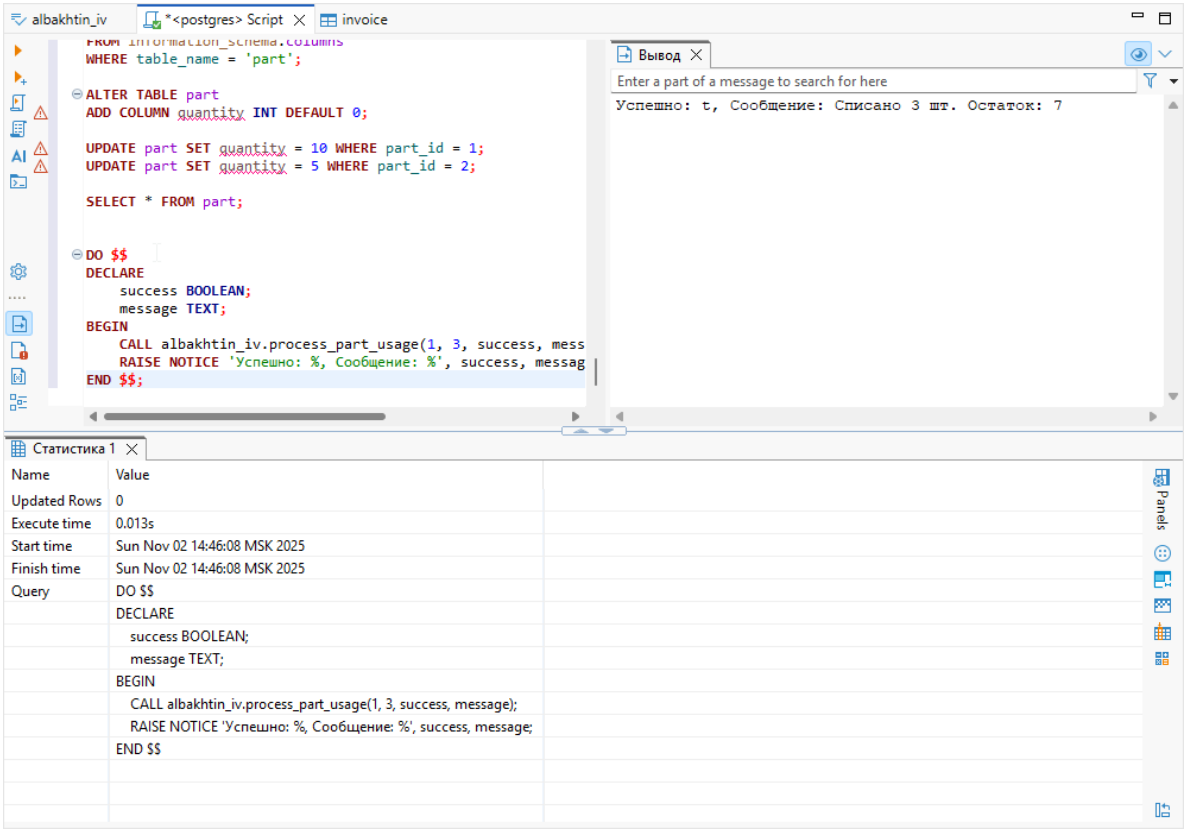


Рисунок 19 - Проверка

Задание 8: Демонстрация вызова хранимой процедуры

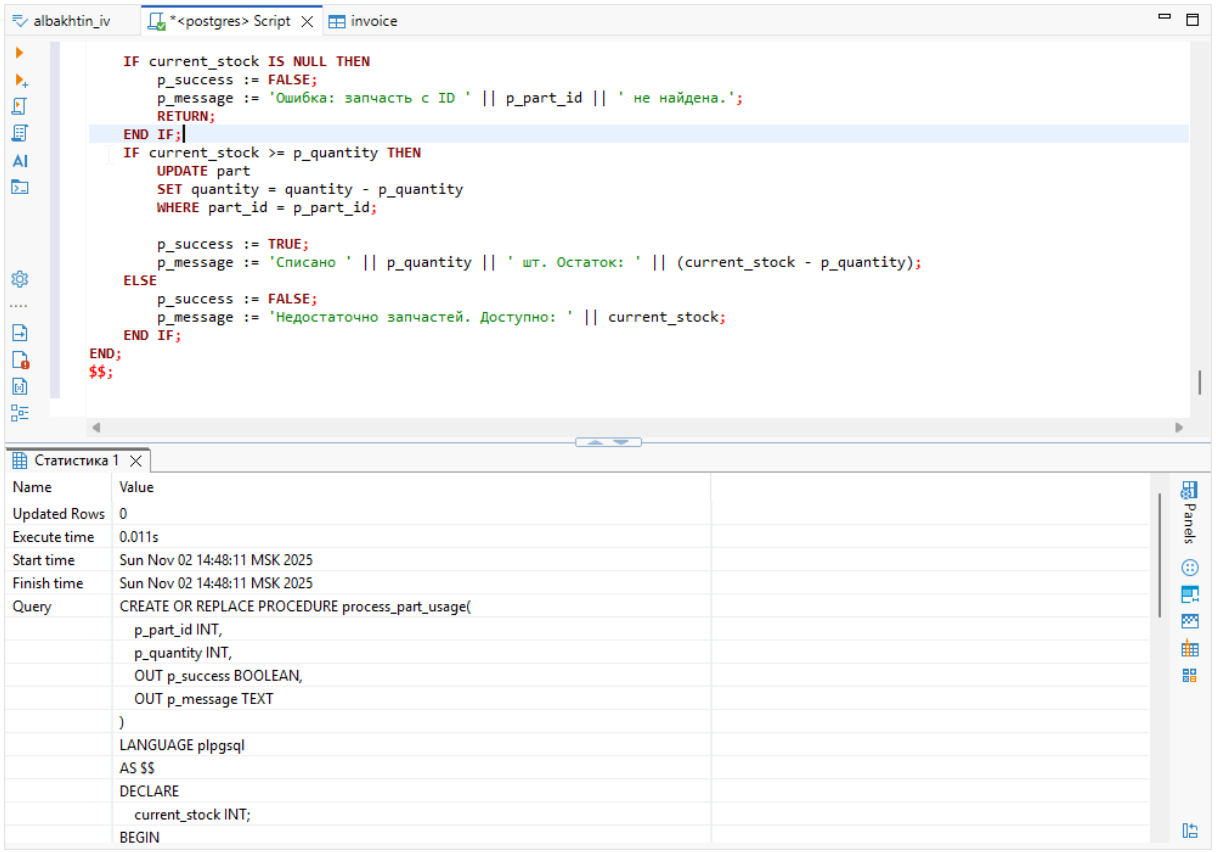


Рисунок 20 – Процедура списания запчастей со склада

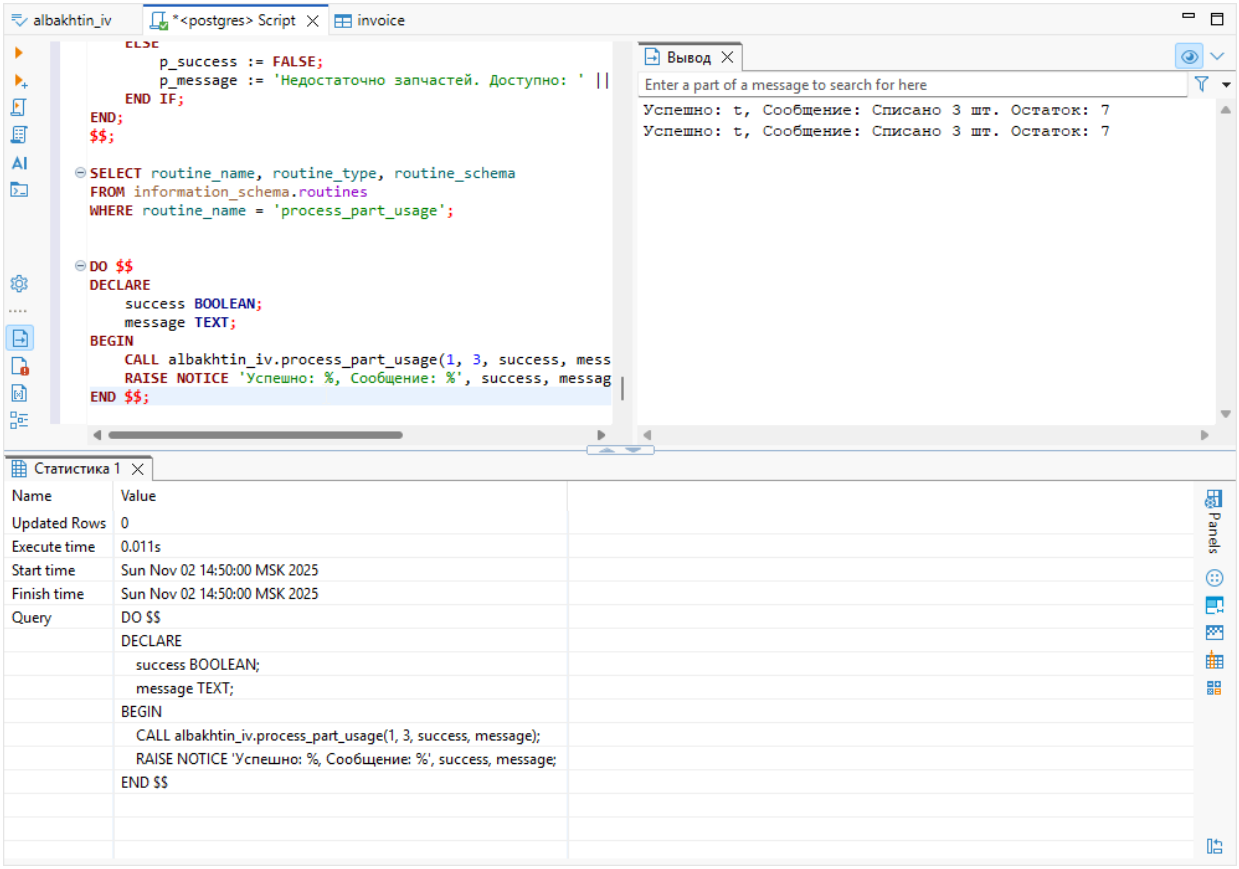


Рисунок 21 – Проверка

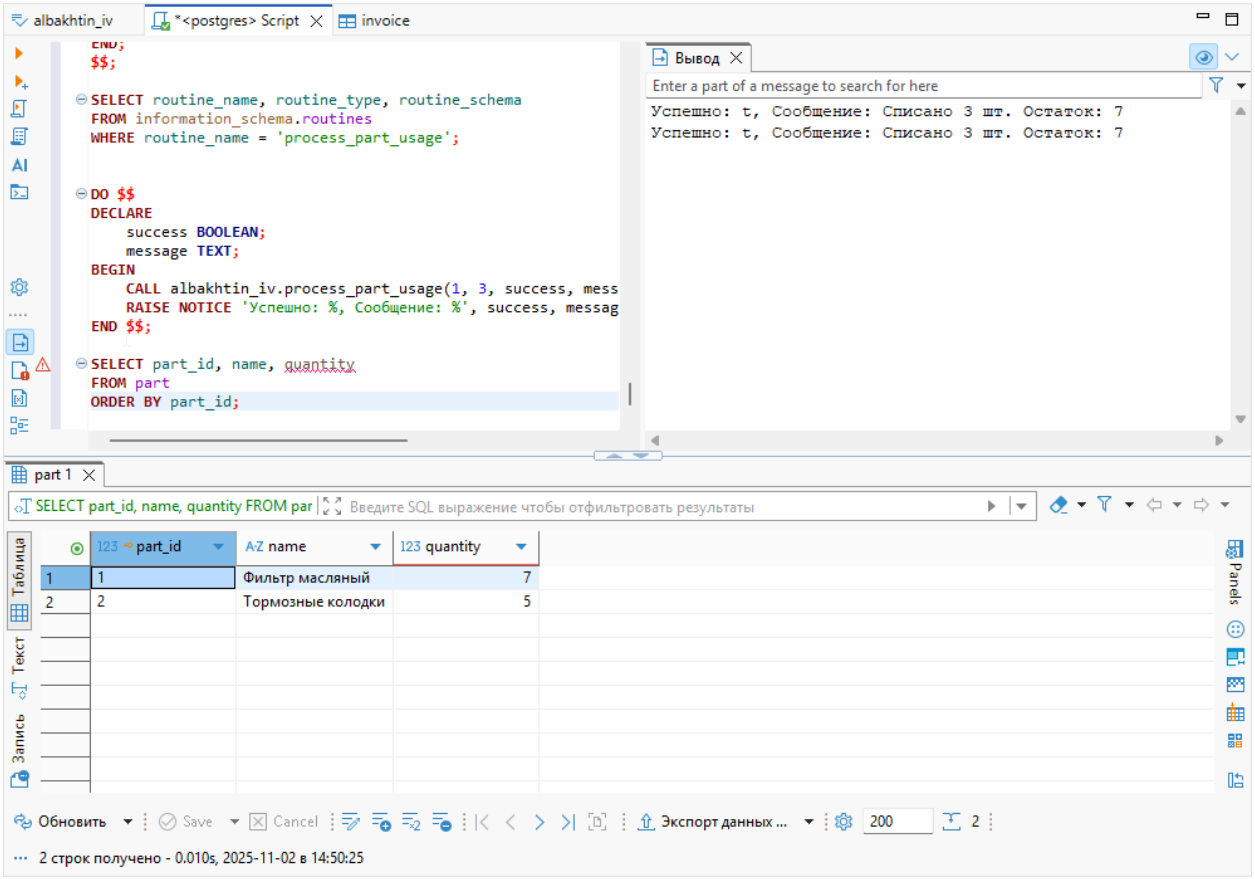


Рисунок 22 – Проверка остатков

ВЫВОД

В ходе выполнения работы была создана процедура process_part_usage, выполняющая автоматическое списание запчастей со склада. Процедура корректно обрабатывает случаи успешного списания, нехватки деталей и отсутствия запчасти. Работа подтверждает успешное применение процедур для автоматизации учёта в базе данных.