



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 №5**  
по дисциплине «Разработка баз данных»

Студент группы

*ИКБО-50-23. Павлов Н.С..*

\_\_\_\_\_  
(подпись)

Преподаватель

*Мажей Я. В.*

\_\_\_\_\_  
(подпись)

Москва 2025 г.

## СОДЕРЖАНИЕ

1. ПОСТАНОВКА ЗАДАЧИ .....	3
2 ВЫПОЛНЕНИЕ РАБОТЫ .....	6
2.1 ИСХОДНЫЕ ТАБЛИЦЫ .....	6
2.2 ЗАДАНИЕ 1 .....	10
2.3 ЗАДАНИЕ 2 .....	11
2.4 ЗАДАНИЕ 3 .....	12
2.5 ЗАДАНИЕ 4 .....	13
2.6 ЗАДАНИЕ 5 .....	14
2.7 ЗАДАНИЕ 6 .....	16
2.8 ЗАДАНИЕ 7 .....	19
2.9 ЗАДАНИЕ 8 .....	21

## 1. ПОСТАНОВКА ЗАДАЧИ

**Цель:** 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.

### **Задачи:**

#### **Задание №1: создание модифицируемого представления**

Для вашей базы данных создать простое модифицируемое представление, которое отбирает строки из одной таблицы по определенному критерию.

#### **Задание №2: модификация данных через представление**

Продemonстрировать возможность изменения данных в базовой таблице через представление, созданное в Задании №1. Для этого необходимо выполнить два запроса:

1. Добавить новую запись с помощью оператора INSERT.
2. Удалить существующую запись с помощью оператора DELETE.

#### **Задание №3: создание немодифицируемого аналитического представления**

Для вашей базы данных создать единое немодифицируемое представление для аналитических целей. Представление должно объединять данные как минимум из двух таблиц и содержать агрегирующие функции (COUNT, SUM, AVG и т.д.) и группировку (GROUP BY).

#### **Задание №4: использование аналитического представления в запросах**

Написать SELECT-запрос, который использует созданное в Задании №3 аналитическое представление в качестве источника данных для дальнейшей фильтрации или анализа.

### **Задание №5: Создание и обновление материализованного представления**

1. Создать материализованное представление для ускорения выполнения ресурсоемкого аналитического запроса.
2. Продемонстрировать процесс обновления данных в представлении с помощью команды `REFRESH MATERIALIZED VIEW viewName;`

### **Задание №6: разработка пользовательской функции для аналитических вычислений**

1. Разработать пользовательскую функцию, которая инкапсулирует комплексный аналитический расчет. Функция должна принимать на вход идентификатор (например, `manufacturer_id`) и возвращать одно скалярное значение (например, общую сумму продаж продукции данного производителя), вычисленное на основе соединения нескольких таблиц и применения агрегатных функций.
2. Продемонстрировать вызов функции в составе `SELECT`-запроса.

### **Задание №7: разработка хранимой процедуры для выполнения сложной операции**

Разработайте хранимую процедуру, которая выполняет безопасную операцию по изменению данных. Процедура должна принимать на вход ID какой-либо записи и числовое значение (например, количество).

Внутри процедуры необходимо проверить, достаточно ли текущего значения в числовом поле одной таблицы для выполнения операции.

- Если да – уменьшите это значение и добавьте новую запись в другую, связанную таблицу.
- Если нет – операция должна полностью прерваться, не внося никаких изменений в данные.

Для сообщения о результате используйте выходной параметр, который вернёт статус успеха или неудачи.

### **Задание №8: демонстрация вызова хранимой процедуры**

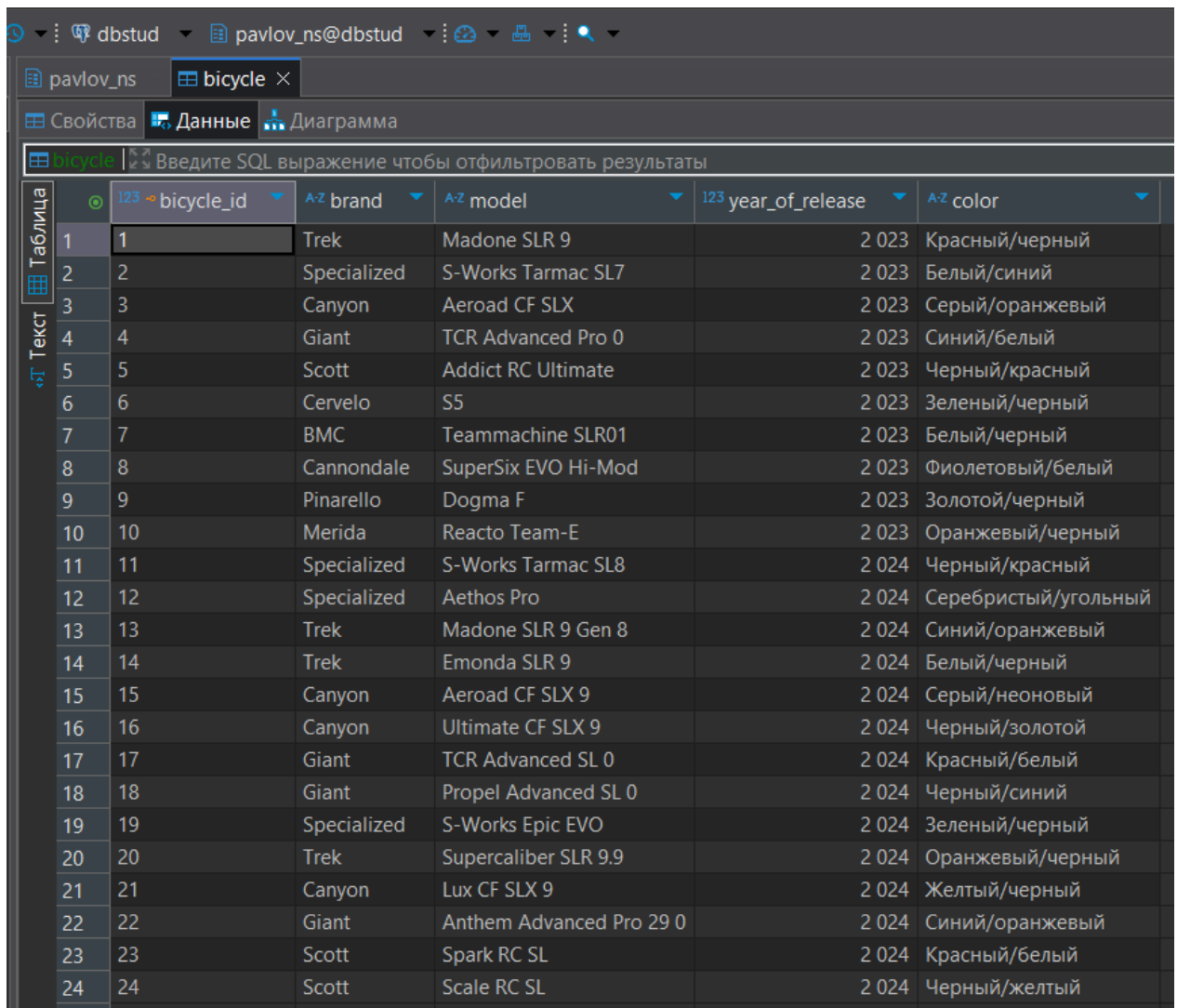
Привести два примера вызова процедуры, созданной в Задании №7:

- Успешный вызов, который добавляет в вашу базу данных уникальную запись.
- Неудачный вызов, который демонстрирует срабатывание реализованной проверки целостности и возврат пользовательской ошибки.

Каждый SQL-запрос сопровождать комментарием, объясняющим его назначение и логику работы с учетом специфики вашей базы данных.

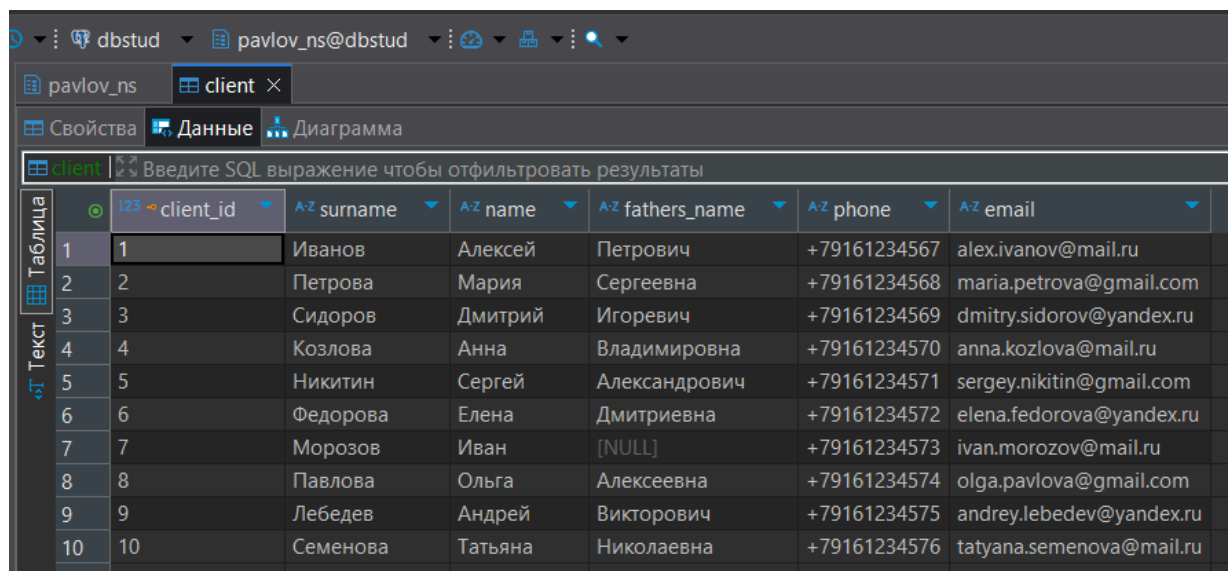
## 2 ВЫПОЛНЕНИЕ РАБОТЫ

### 2.1 ИСХОДНЫЕ ТАБЛИЦЫ



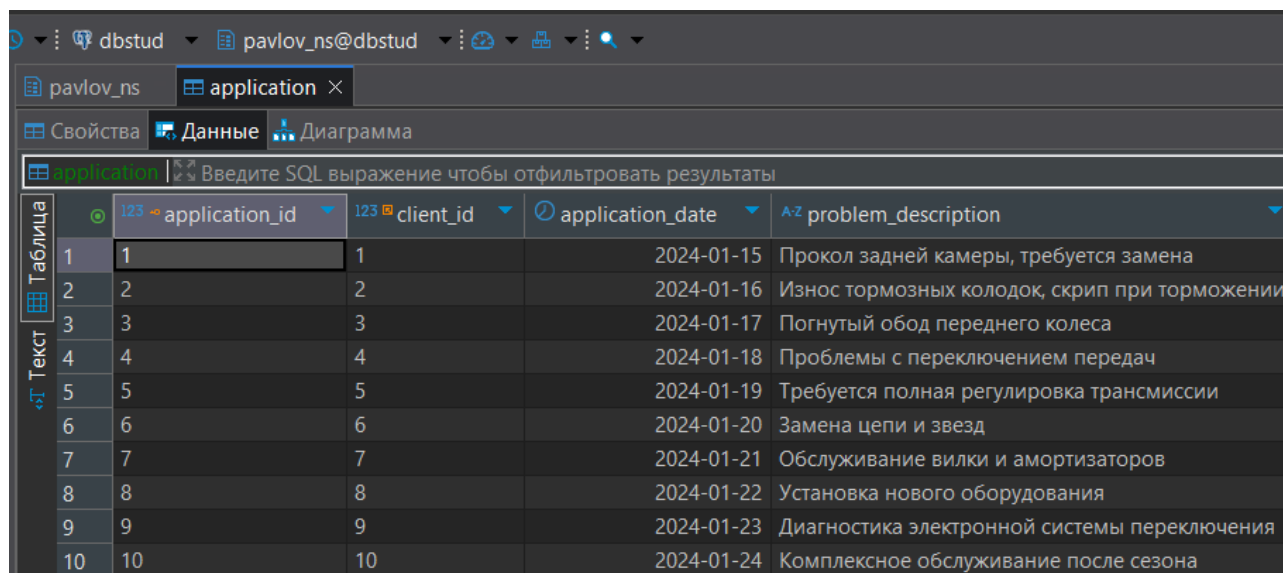
	bicycle_id	brand	model	year_of_release	color
1	1	Trek	Madone SLR 9	2 023	Красный/черный
2	2	Specialized	S-Works Tarmac SL7	2 023	Белый/синий
3	3	Canyon	Aeroad CF SLX	2 023	Серый/оранжевый
4	4	Giant	TCR Advanced Pro 0	2 023	Синий/белый
5	5	Scott	Addict RC Ultimate	2 023	Черный/красный
6	6	Cervelo	S5	2 023	Зеленый/черный
7	7	BMC	Teammachine SLR01	2 023	Белый/черный
8	8	Cannondale	SuperSix EVO Hi-Mod	2 023	Фиолетовый/белый
9	9	Pinarello	Dogma F	2 023	Золотой/черный
10	10	Merida	Reacto Team-E	2 023	Оранжевый/черный
11	11	Specialized	S-Works Tarmac SL8	2 024	Черный/красный
12	12	Specialized	Aethos Pro	2 024	Серебристый/угольный
13	13	Trek	Madone SLR 9 Gen 8	2 024	Синий/оранжевый
14	14	Trek	Emonda SLR 9	2 024	Белый/черный
15	15	Canyon	Aeroad CF SLX 9	2 024	Серый/неоновый
16	16	Canyon	Ultimate CF SLX 9	2 024	Черный/золотой
17	17	Giant	TCR Advanced SL 0	2 024	Красный/белый
18	18	Giant	Propel Advanced SL 0	2 024	Черный/синий
19	19	Specialized	S-Works Epic EVO	2 024	Зеленый/черный
20	20	Trek	Supercaliber SLR 9.9	2 024	Оранжевый/черный
21	21	Canyon	Lux CF SLX 9	2 024	Желтый/черный
22	22	Giant	Anthem Advanced Pro 29 0	2 024	Синий/оранжевый
23	23	Scott	Spark RC SL	2 024	Красный/белый
24	24	Scott	Scale RC SL	2 024	Черный/желтый

Рисунок 1 – Таблица bicycle



	client_id	surname	name	fathers_name	phone	email
1	1	Иванов	Алексей	Петрович	+79161234567	alex.ivanov@mail.ru
2	2	Петрова	Мария	Сергеевна	+79161234568	maria.petrova@gmail.com
3	3	Сидоров	Дмитрий	Игоревич	+79161234569	dmitry.sidorov@yandex.ru
4	4	Козлова	Анна	Владимировна	+79161234570	anna.kozlova@mail.ru
5	5	Никитин	Сергей	Александрович	+79161234571	sergey.nikitin@gmail.com
6	6	Федорова	Елена	Дмитриевна	+79161234572	elena.fedorova@yandex.ru
7	7	Морозов	Иван	[NULL]	+79161234573	ivan.morozov@mail.ru
8	8	Павлова	Ольга	Алексеевна	+79161234574	olga.pavlova@gmail.com
9	9	Лебедев	Андрей	Викторович	+79161234575	andrey.lebedev@yandex.ru
10	10	Семенова	Татьяна	Николаевна	+79161234576	tatyana.semenova@mail.ru

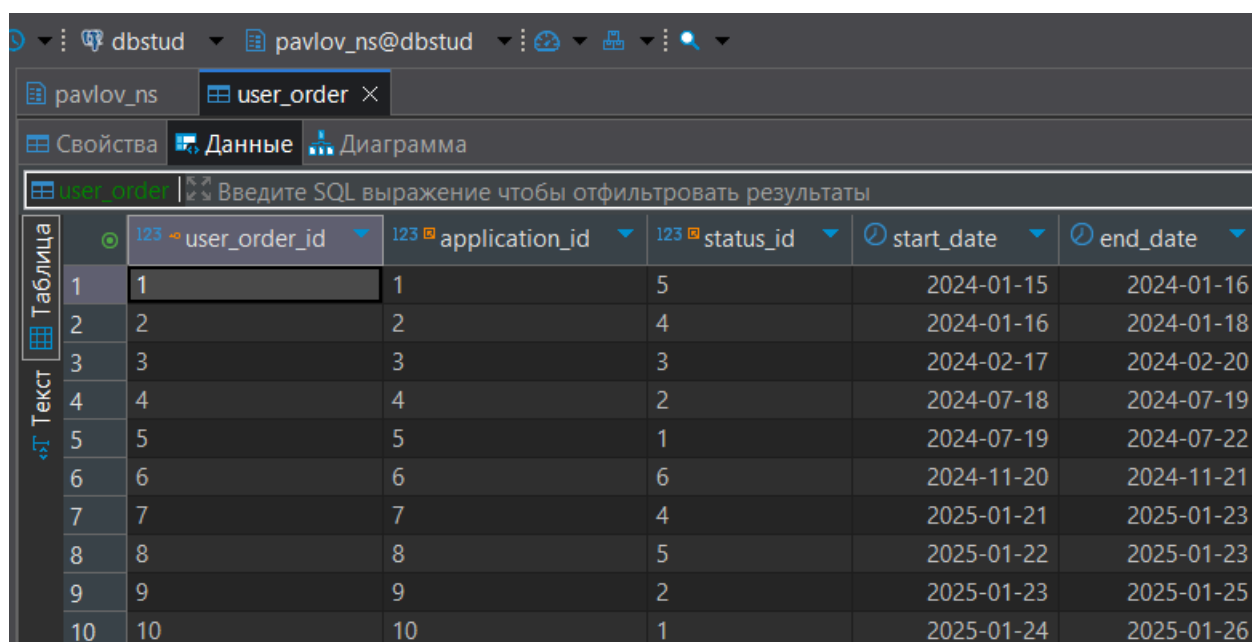
Рисунок 2 – Таблица client



The screenshot shows the 'application' table in the 'pavlov\_ns' database. The table has five columns: 'application\_id', 'client\_id', 'application\_date', and 'problem\_description'. The 'application\_id' column is highlighted with a blue background. The table contains 10 rows of data, numbered 1 to 10. The 'problem\_description' column contains text descriptions of vehicle issues in Russian.

	application_id	client_id	application_date	problem_description
1	1	1	2024-01-15	Прокол задней камеры, требуется замена
2	2	2	2024-01-16	Износ тормозных колодок, скрип при торможении
3	3	3	2024-01-17	Погнутый обод переднего колеса
4	4	4	2024-01-18	Проблемы с переключением передач
5	5	5	2024-01-19	Требуется полная регулировка трансмиссии
6	6	6	2024-01-20	Замена цепи и звезд
7	7	7	2024-01-21	Обслуживание вилки и амортизаторов
8	8	8	2024-01-22	Установка нового оборудования
9	9	9	2024-01-23	Диагностика электронной системы переключения
10	10	10	2024-01-24	Комплексное обслуживание после сезона

Рисунок 3 – Таблица application



The screenshot shows the 'user\_order' table in the 'pavlov\_ns' database. The table has five columns: 'user\_order\_id', 'application\_id', 'status\_id', 'start\_date', and 'end\_date'. The 'user\_order\_id' column is highlighted with a blue background. The table contains 10 rows of data, numbered 1 to 10. The 'start\_date' and 'end\_date' columns contain date values.

	user_order_id	application_id	status_id	start_date	end_date
1	1	1	5	2024-01-15	2024-01-16
2	2	2	4	2024-01-16	2024-01-18
3	3	3	3	2024-02-17	2024-02-20
4	4	4	2	2024-07-18	2024-07-19
5	5	5	1	2024-07-19	2024-07-22
6	6	6	6	2024-11-20	2024-11-21
7	7	7	4	2025-01-21	2025-01-23
8	8	8	5	2025-01-22	2025-01-23
9	9	9	2	2025-01-23	2025-01-25
10	10	10	1	2025-01-24	2025-01-26

Рисунок 4 – Таблица user\_order

dbstud pavlov\_ns@dbstud

pavlov\_ns estimate X

Свойства Данные Диаграмма

estimate Введите SQL выражение чтобы отфильтровать результаты

	123 estimate_id	123 user_order_id	123 price_of_work	123 price_of_spare_parts	123 total_price
1	1	1	500	850	1 350
2	2	2	800	1 200	2 000
3	3	3	1 500	4 500	6 000
4	4	4	1 000	8 500	9 500
5	5	5	2 000	2 500	4 500
6	6	6	1 200	3 800	5 000
7	7	7	3 000	25 000	28 000
8	8	8	1 000	4 800	5 800
9	9	9	1 500	1 800	3 300
10	10	10	4 000	45 000	49 000

Рисунок 5 – Таблица estimate

dbstud pavlov\_ns@dbstud

pavlov\_ns user\_order\_work X

Свойства Данные Диаграмма

user\_order\_work Введите SQL выражение чтобы отфильтровать результаты

	123 user_order_work_id	123 user_order_id	123 work_id
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10

Рисунок 6 – Таблица user\_order\_work

dbstud pavlov\_ns@dbstud

pavlov\_ns work X

Свойства Данные Диаграмма

work Введите SQL выражение чтобы отфильтровать результаты

	123 work_id	A-z title	A-z description_of_the_execution	123 lead_time	123 price
1	1	Замена камеры	Демонтаж покрышки, замена камеры, монтаж покрышки	30	500
2	2	Замена тормозных колодок	Снятие старых колодок, установка новых, регулировка	45	800
3	3	Правка обода	Выравнивание обода на станке, регулировка спиц	90	1 500
4	4	Регулировка переключения	Настройка переднего и заднего переключателей	60	1 000
5	5	Полная регулировка трансмиссии	Чистка, смазка, регулировка всей трансмиссии	120	2 000
6	6	Замена цепи и звезд	Замена цепи и кассеты, регулировка	75	1 200
7	7	Обслуживание вилки	Разборка, чистка, замена масла, сборка	180	3 000
8	8	Установка оборудования	Установка и настройка нового оборудования	60	1 000
9	9	Диагностика электроники	Проверка электронной системы, перепрошивка	90	1 500
10	10	Комплексное обслуживание	Полная диагностика и обслуживание всех систем	240	4 000

Рисунок 7 – Таблица work



	work_spare_part_id	work_id	spare_part_id
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10

Рисунок 8 – Таблица work\_spare\_part

	spare_part_id	title	specifications	price	supplier_id
1	1	Камера велосипедная 700x25-32	Резиновая, Presta, 48mm	850	1
2	2	Тормозные колодки Shimano R55C4	Для дисковых тормозов, керамические	1 200	5
3	3	Обод Mavic Ksyrium Elite	Алюминиевый, 28 спиц, 700c	4 500	2
4	4	Манетка Shimano Ultegra R8050	Электронная, 11-скоростная	8 500	5
5	5	Цепь Shimano HG701	11-скоростная, 116 звеньев	2 500	5
6	6	Звезды Shimano 105 R7000	Кассета 11-34T, 11 скоростей	3 800	5
7	7	Вилка RockShox Sid SL Ultimate	Карбон, ход 100mm, дисковые тормоза	25 000	8
8	8	Батарея Shimano Di2	BT-DN110, для электронного переключения	4 800	5
9	9	Подшипники Enduro	Комплект для втулок, промышленные	1 800	3
10	10	Трансмиссия SRAM Force eTap AXS	Беспроводная, 12-скоростная	45 000	6
11	32	Вилка Fox 36 Factory	Ход 160mm, воздушная пружина, амортизатор GRIP2, 29"/27.5", дисковые тормоза	125 000	9
12	33	Амортизатор Fox Float X2 Factory	Воздушный, регулировка высоко/низкоскоростного отскока и компрессии, 230x65mm	85 000	9
13	34	Вилка Fox 34 Performance Elite	Ход 130mm, воздушная пружина, амортизатор FIT4, 29"/27.5", дисковые тормоза	75 000	9
14	35	Амортизатор Fox DHX2 Factory	Пружинный, регулировка высоко/низкоскоростного отскока, 250x75mm	92 000	9
15	36	Вилка Fox 40 Factory	Двухкоронная, ход 190-203mm, воздушная пружина, 27.5", для даунхилла	140 000	9
16	37	Группа SRAM RED eTap AXS	Беспроводная электронная группа, 12-скоростная, гидравлические дисковые тормоза	280 000	6
17	38	Группа SRAM XX SL Eagle AXS	Беспроводная электронная группа для MTB, 12-скоростная, 10-52T	220 000	6
18	39	Тормоза SRAM Level Ultimate	Гидравлические дисковые тормоза, 4-поршневые, с технологией SwingLink	45 000	6
19	40	Система SRAM XX1 Eagle Carbon	Шатуны из карбона, 32T, каретка DUB, 12-скоростная	68 000	6
20	41	Кассета SRAM XG-1299 Eagle	12-скоростная, 10-52T, титановые шестерни, вес 268g	55 000	6

Рисунок 9 – Таблица spare\_part

	supplier_id	organization_title	address	phone
1	1	Велокомплект	г. Москва, ул. Ленина, 25	+74951234567
2	2	СпортЗапчасти	г. Санкт-Петербург, Невский пр., 100	+78121234567
3	3	БайкТек	г. Екатеринбург, ул. Мира, 15	+73431234567
4	4	ВелоМир	г. Новосибирск, Красный пр., 50	+73831234567
5	5	Шимано Рус	г. Москва, Ленинградский пр., 80	+74959876543
6	6	СРАМ Дистрибьюшн	г. Казань, ул. Баумана, 30	+78431234567
7	7	Кампагноло	г. Москва, ул. Тверская, 45	+74957778899
8	8	РокШок Сервис	г. Краснодар, ул. Красная, 120	+78611234567
9	9	Фокс Раша	г. Сочи, ул. Курортная, 25	+86221234567
10	10	ДТ Швейцария	г. Москва, Кутузовский пр., 32	+74956667788

Рисунок 10 – Таблица supplier

## 2.2 ЗАДАНИЕ 1

The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a script titled 'Script-2' with the following SQL code:

```
-- Создание модифицируемого представления для велосипедов марки Canyon
CREATE OR REPLACE VIEW canyon_bicycles AS
SELECT *
FROM bicycle
WHERE brand = 'Canyon';

-- Проверка представления
SELECT * FROM canyon_bicycles
```

The bottom pane shows the results of the query 'SELECT \* FROM canyon\_bicycles'. The results are displayed in a table with the following columns: bicycle\_id, brand, model, year\_of\_release, and color. The table contains four rows of data for Canyon bicycles.

	bicycle_id	brand	model	year_of_release	color
1	3	Canyon	Aeroad CF SLX	2 023	Серый/оранжевый
2	15	Canyon	Aeroad CF SLX	2 024	Серый/неоновый
3	16	Canyon	Ultimate CF SLX	2 024	Черный/золотой
4	21	Canyon	Lux CF SLX 9	2 024	Желтый/черный

Рисунок 11 – Создание модифицированного представления

## 2.3 ЗАДАНИЕ 2

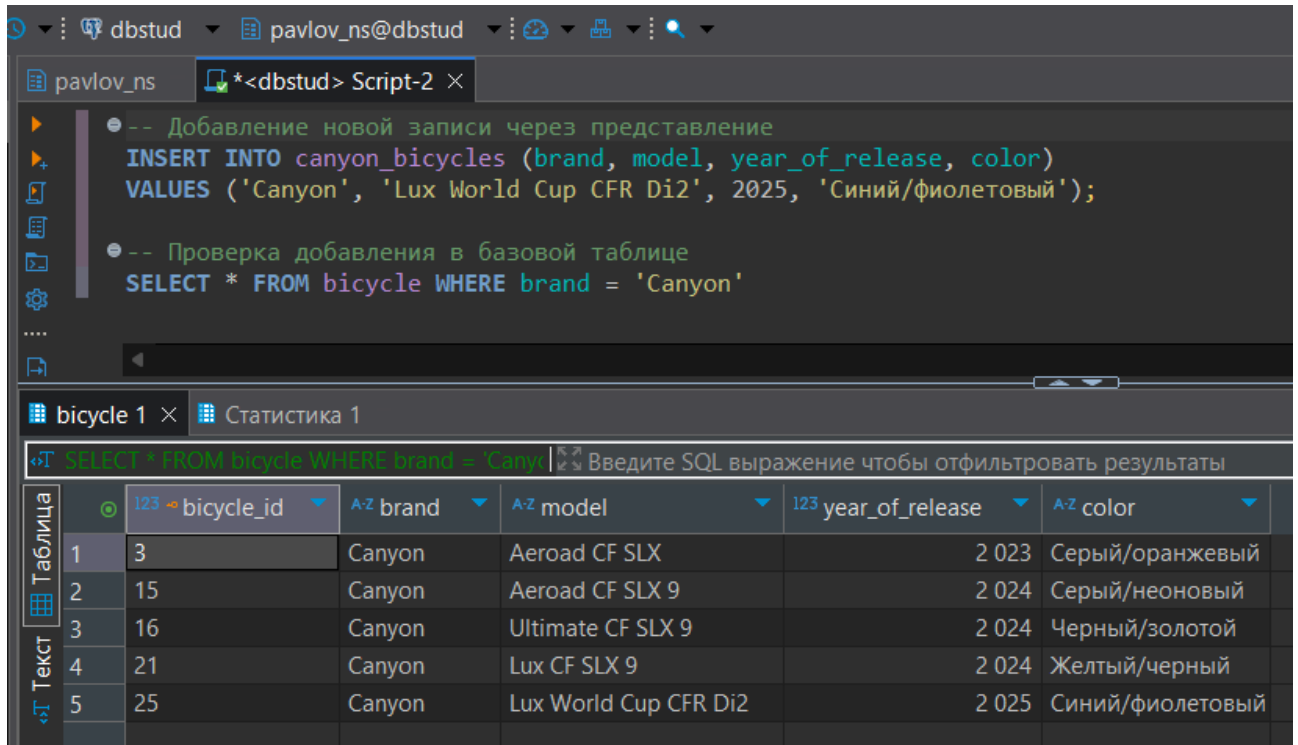


Рисунок 12 – Добавление записи через представление

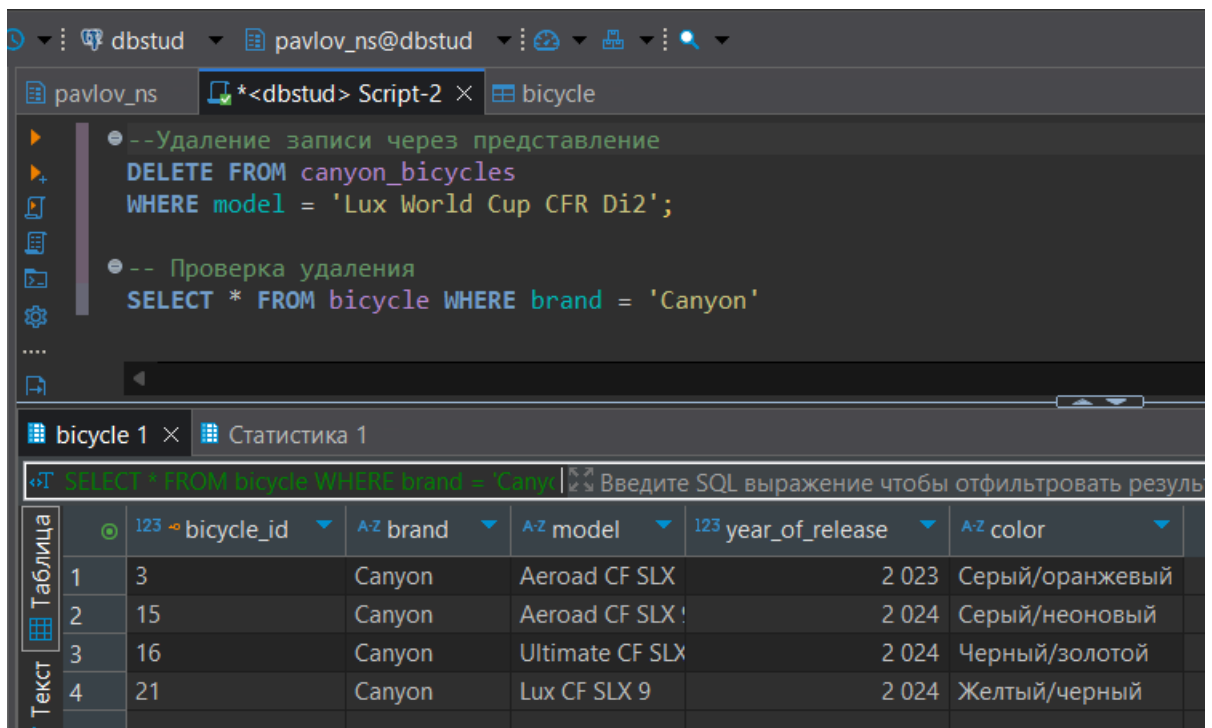


Рисунок 13 – Удаление записи через представление

## 2.4 ЗАДАНИЕ 3

The screenshot displays the dbstud interface. The top toolbar shows the database name 'pavlov\_ns' and the user 'pavlov\_ns@dbstud'. Below the toolbar, a script editor window titled '\*<dbstud> Script-2' contains the following SQL code:

```
-- Создание аналитического представления по клиентам и их заказам
CREATE OR REPLACE VIEW client_order_summary AS
SELECT
  c.surname || ' ' || c.name AS client_name,
  c.phone,
  c.email,
  COALESCE(SUM(e.total_price), 0) AS total_spent,
  MAX(uo.start_date) AS last_order_date
FROM client c
LEFT JOIN application a ON c.client_id = a.client_id
LEFT JOIN user_order uo ON a.application_id = uo.application_id
LEFT JOIN estimate e ON uo.user_order_id = e.user_order_id
GROUP BY c.client_id;

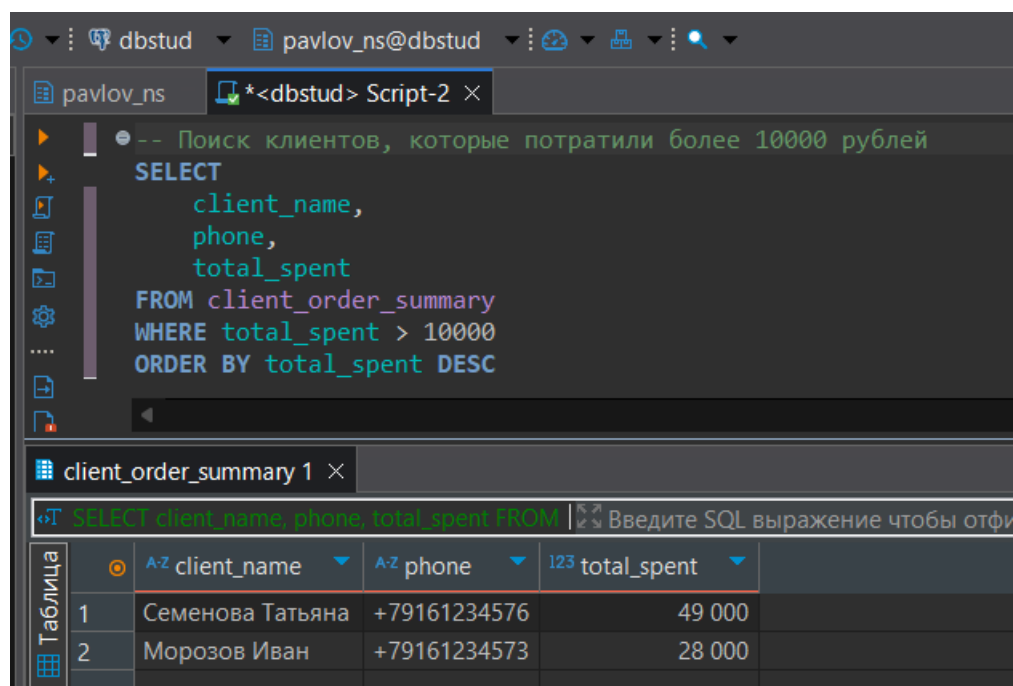
-- Проверка представления
SELECT * FROM client_order_summary
```

Below the script editor, the results of the query are displayed in a table view. The table is titled 'client\_order\_summary 1' and 'Статистика 1'. The columns are: client\_name, phone, email, total\_spent, and last\_order\_date. The results show 10 rows of data.

	client_name	phone	email	total_spent	last_order_date
1	Лебедев Андрей	+79161234575	andrey.lebedev@yandex.ru	3 300	2025-01-23
2	Сидоров Дмитрий	+79161234569	dmitry.sidorov@yandex.ru	6 000	2024-02-17
3	Никитин Сергей	+79161234571	sergey.nikitin@gmail.com	4 500	2024-07-19
4	Козлова Анна	+79161234570	anna.kozlova@mail.ru	9 500	2024-07-18
5	Семенова Татьяна	+79161234576	tatyana.semenova@mail.ru	49 000	2025-01-24
6	Федорова Елена	+79161234572	elena.fedorova@yandex.ru	5 000	2024-11-20
7	Петрова Мария	+79161234568	maria.petrova@gmail.com	2 000	2024-01-16
8	Морозов Иван	+79161234573	ivan.morozov@mail.ru	28 000	2025-01-21
9	Иванов Алексей	+79161234567	alex.ivanov@mail.ru	1 350	2024-01-15
10	Павлова Ольга	+79161234574	olga.pavlova@gmail.com	5 800	2025-01-22

Рисунок 14 – Создание немодифицируемого аналитического представления

## 2.5 ЗАДАНИЕ 4



The screenshot shows the dbstud interface. The top toolbar includes icons for connection, query, and other database functions. The main window has two tabs: 'pavlov\_ns' and '\*<dbstud> Script-2'. The 'Script-2' tab is active, displaying the following SQL query:

```
-- Поиск клиентов, которые потратили более 10000 рублей
SELECT
    client_name,
    phone,
    total_spent
FROM client_order_summary
WHERE total_spent > 10000
ORDER BY total_spent DESC
```

Below the query editor, there is a tab labeled 'client\_order\_summary 1'. Below this tab, the results of the query are displayed in a table view. The table has four columns: 'client\_name', 'phone', 'total\_spent', and an unlabeled column. The results are as follows:

	A-Z client_name	A-Z phone	123 total_spent	
1	Семенова Татьяна	+79161234576	49 000	
2	Морозов Иван	+79161234573	28 000	

Рисунок 15 – Запрос, использующий аналитическое представление

## 2.6 ЗАДАНИЕ 5

The screenshot shows a database client interface with a script editor and a table view. The script editor contains two SQL queries: one to create a materialized view and another to check it. The table view displays the data from the materialized view.

```
-- Создание материализованного представления
CREATE MATERIALIZED VIEW monthly_financial_report AS
SELECT
    DATE_TRUNC('month', uo.start_date)::DATE AS report_month,
    COUNT(DISTINCT uo.user_order_id) AS total_orders,
    SUM(e.total_price) AS total_revenue,
    SUM(e.price_of_work) AS total_work_revenue,
    SUM(e.price_of_spare_parts) AS total_parts_revenue,
    AVG(e.total_price) AS avg_order_amount
FROM user_order uo
JOIN application a ON uo.application_id = a.application_id
JOIN client c ON a.client_id = c.client_id
JOIN estimate e ON uo.user_order_id = e.user_order_id
GROUP BY DATE_TRUNC('month', uo.start_date)::DATE
ORDER BY report_month;

-- Проверка материализованного представления
SELECT * FROM monthly_financial_report
```

	report_month	total_orders	total_revenue	total_work_revenue	total_parts_revenue	avg_order_amount
1	2024-01-01	2	3 350	1 300	2 050	1 675
2	2024-02-01	1	6 000	1 500	4 500	6 000
3	2024-07-01	2	14 000	3 000	11 000	7 000
4	2024-11-01	1	5 000	1 200	3 800	5 000
5	2025-01-01	4	86 100	9 500	76 600	21 525

Рисунок 16 – Создание материализованного представления

The screenshot shows a database client interface with a script editor and a table view. The script editor contains three SQL queries: two to insert new data and one to check the materialized view. The table view displays the data from the materialized view.

```
-- Добавим новый заказ
INSERT INTO application (client_id, application_date, problem_description)
VALUES (1, '2024-02-01', 'Тестовый заказ');

INSERT INTO user_order (application_id, status_id, start_date, end_date)
VALUES (11, 1, '2024-02-01', '2024-02-05');

INSERT INTO estimate (user_order_id, price_of_work, price_of_spare_parts, total_price)
VALUES (11, 2000, 1500, 3500);

-- Проверим, что в материализованном представлении нет новых данных
SELECT * FROM monthly_financial_report WHERE report_month = '2024-02-01'
```

	report_month	total_orders	total_revenue	total_work_revenue	total_parts_revenue	avg_order_amount
1	2024-02-01	1	6 000	1 500	4 500	6 000

Рисунок 17 – Добавление новых данных

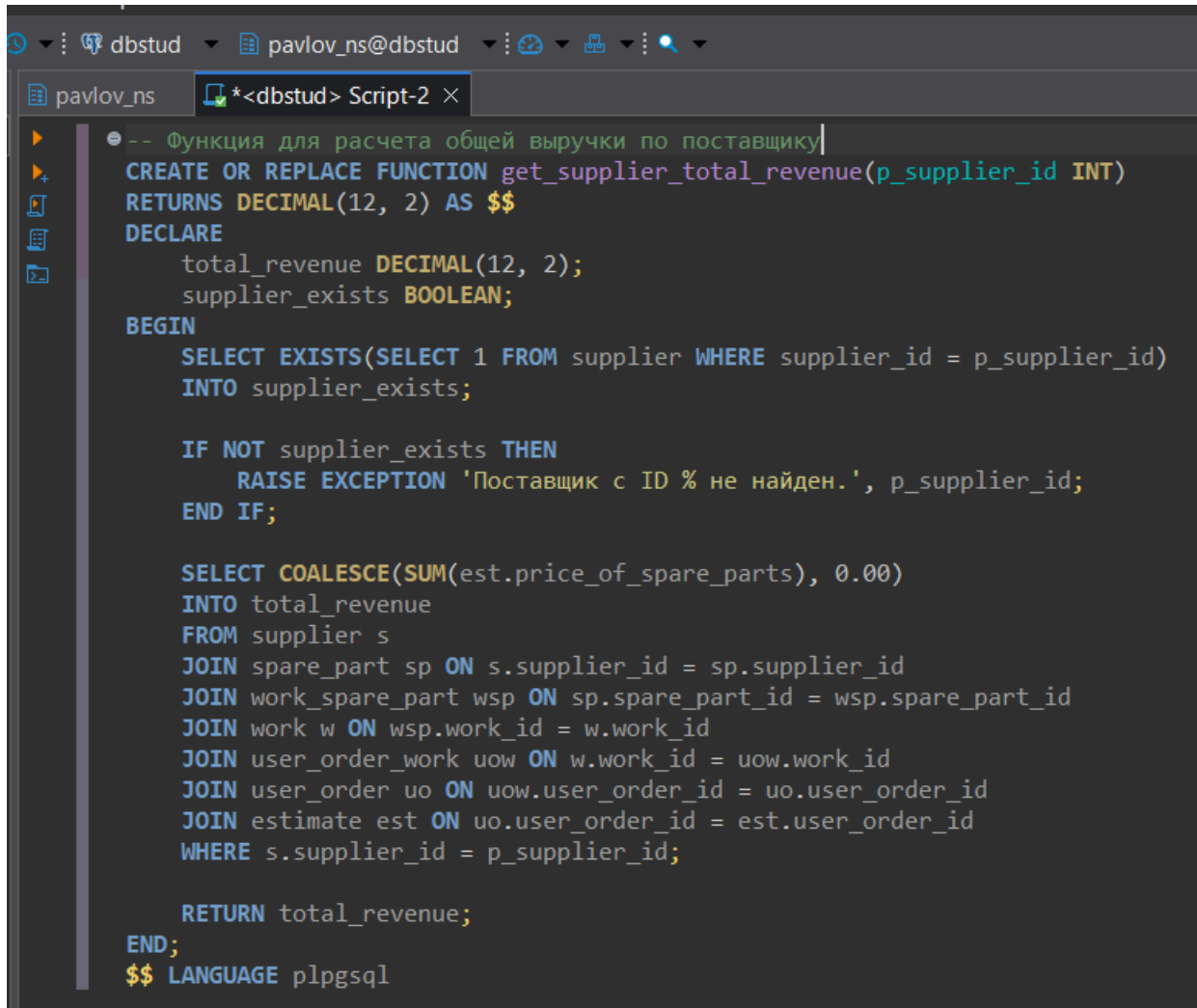
The screenshot shows a database client window with the following content:

- Script Editor:**
  - Comment: -- Обновляем материализованное представление
  - SQL: `REFRESH MATERIALIZED VIEW monthly_financial_report;`
  - Comment: -- Проверяем обновленные данные
  - SQL: `SELECT * FROM monthly_financial_report WHERE report_month = '2024-02-01'`
- Query Results:**
  - Tab: monthly\_financial\_report 1
  - Stat: Статистика 1
  - Filter: Введите SQL выражение чтобы отфильтровать результаты
- Data Table:**

	report_month	total_orders	total_revenue	total_work_revenue	total_parts_revenue	avg_order_amount
1	2024-02-01	2	9 500	3 500	6 000	4 750

Рисунок 18 – Обновление материализованного представления

## 2.7 ЗАДАНИЕ 6



The screenshot shows a database IDE window with a dark theme. The top bar displays the connection 'pavlov\_ns@dbstud'. The main editor area contains a PL/SQL script for creating a function named 'get\_supplier\_total\_revenue'. The script includes a comment in Russian, a function signature, a DECLARE section for variables, a BEGIN section with an EXISTS check and an exception handler, a complex JOIN query to calculate total revenue, and an END section with a LANGUAGE plpgsql directive.

```
-- Функция для расчета общей выручки по поставщику|
CREATE OR REPLACE FUNCTION get_supplier_total_revenue(p_supplier_id INT)
RETURNS DECIMAL(12, 2) AS $$
DECLARE
    total_revenue DECIMAL(12, 2);
    supplier_exists BOOLEAN;
BEGIN
    SELECT EXISTS(SELECT 1 FROM supplier WHERE supplier_id = p_supplier_id)
    INTO supplier_exists;

    IF NOT supplier_exists THEN
        RAISE EXCEPTION 'Поставщик с ID % не найден.', p_supplier_id;
    END IF;

    SELECT COALESCE(SUM(est.price_of_spare_parts), 0.00)
    INTO total_revenue
    FROM supplier s
    JOIN spare_part sp ON s.supplier_id = sp.supplier_id
    JOIN work_spare_part wsp ON sp.spare_part_id = wsp.spare_part_id
    JOIN work w ON wsp.work_id = w.work_id
    JOIN user_order_work uow ON w.work_id = uow.work_id
    JOIN user_order uo ON uow.user_order_id = uo.user_order_id
    JOIN estimate est ON uo.user_order_id = est.user_order_id
    WHERE s.supplier_id = p_supplier_id;

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

Рисунок 19 – Скрипт создания функции



The screenshot shows a database IDE with a script editor and a statistics window. The script editor contains a PL/SQL function definition for `get_supplier_total_revenue`. The statistics window, titled "Статистика 1", shows the execution details of the script.

**Script Editor Content:**

```

WHERE s.supplier_id = p_supplier_id;

RETURN total_revenue;
END;
$$ LANGUAGE plpgsql

```

**Statistics Window Content:**

Name	Value
Updated Rows	0
Execute time	0.025s
Start time	Fri Oct 31 09:21:12 MSK 2025
Finish time	Fri Oct 31 09:21:12 MSK 2025
Query	-- Функция для расчета общей выручки по поставщику CREATE OR REPLACE FUNCTION get_supplier_total_revenue(p_supplier_id INT) RETURNS DECIMAL(12, 2) AS \$\$ DECLARE total_revenue DECIMAL(12, 2); supplier_exists BOOLEAN; BEGIN SELECT EXISTS(SELECT 1 FROM supplier WHERE supplier_id = p_supplier_id) INTO supplier_exists; IF NOT supplier_exists THEN RAISE EXCEPTION 'Поставщик с ID % не найден.', p_supplier_id; END IF; SELECT COALESCE(SUM(est.price_of_spare_parts), 0.00) INTO total_revenue FROM supplier s JOIN spare_part sp ON s.supplier_id = sp.supplier_id JOIN work_spare_part wsp ON sp.spare_part_id = wsp.spare_part_id JOIN work w ON wsp.work_id = w.work_id JOIN user_order_work uow ON w.work_id = uow.work_id JOIN user_order uo ON uow.user_order_id = uo.user_order_id JOIN estimate est ON uo.user_order_id = est.user_order_id WHERE s.supplier_id = p_supplier_id; RETURN total_revenue; END; \$\$ LANGUAGE plpgsql

Рисунок 20 – Сообщение об успешном создании функции

dbstud   pavlov\_ns@dbstud   \*<dbstud> Script-2 ×

pavlov\_ns   -- Демонстрация вызова функции

```

SELECT
    organization_title AS "Поставщик",
    get_supplier_total_revenue(supplier_id) AS "Общая выручка"
FROM supplier
ORDER BY "Общая выручка" DESC

```

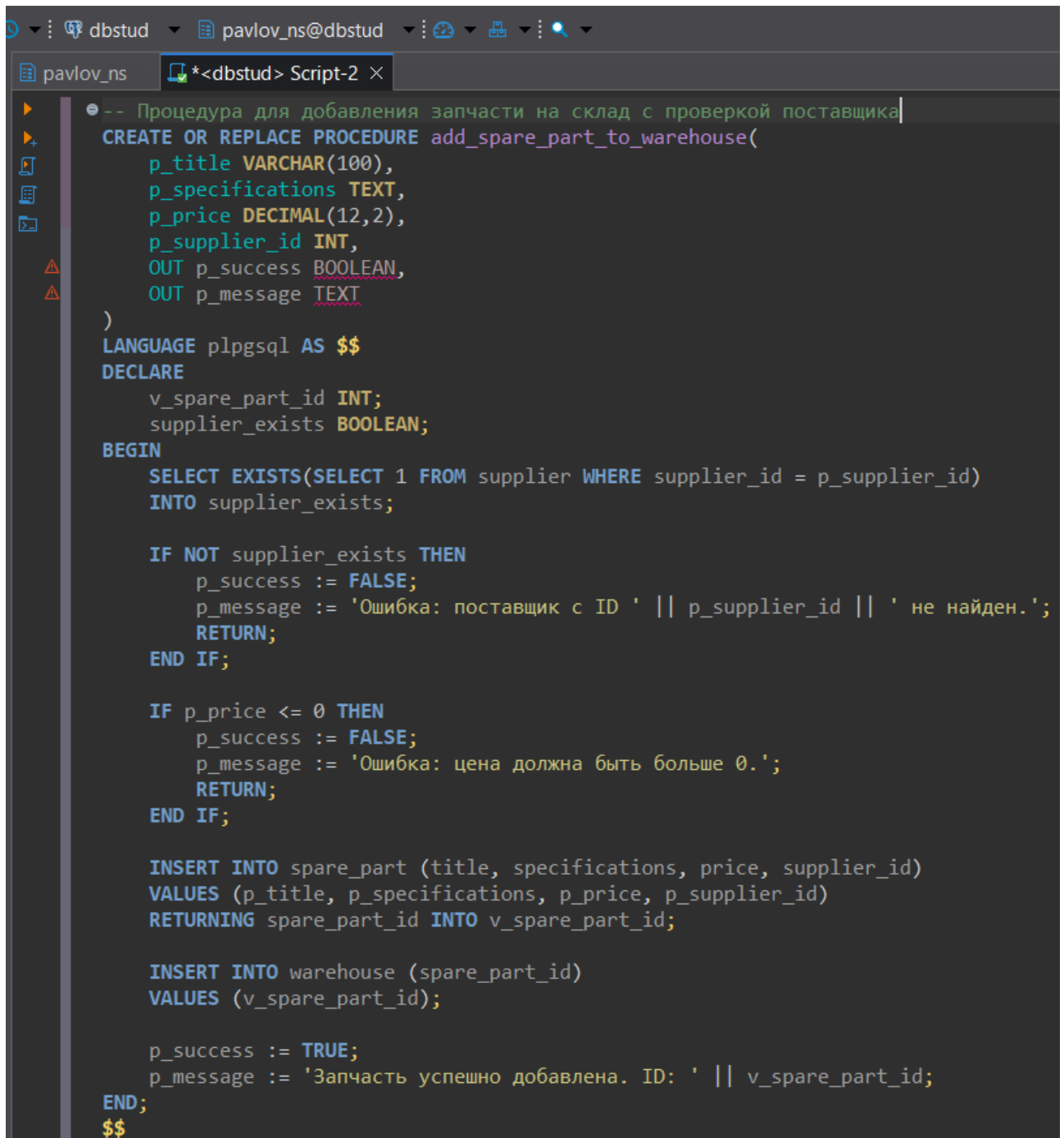
supplier 1 ×

SELECT organization\_title AS "Поставщик", ge Введите SQL выражение чтобы отфиль

	Поставщик	Общая выручка
1	СРАМ Дистрибьюшн	45 000
2	РокШок Сервис	25 000
3	Шимано Рус	20 800
4	СпортЗапчасти	4 500
5	БайкТек	1 800
6	Велокомплект	850
7	Кампагноло	0
8	ВелоМир	0
9	Фокс Раша	0
10	ДТ Швейцария	0

Рисунок 21 – Демонстрация вызова функции

## 2.8 ЗАДАНИЕ 7



```
-- Процедура для добавления запчастей на склад с проверкой поставщика
CREATE OR REPLACE PROCEDURE add_spare_part_to_warehouse(
    p_title VARCHAR(100),
    p_specifications TEXT,
    p_price DECIMAL(12,2),
    p_supplier_id INT,
    OUT p_success BOOLEAN,
    OUT p_message TEXT
)
LANGUAGE plpgsql AS $$
DECLARE
    v_spare_part_id INT;
    supplier_exists BOOLEAN;
BEGIN
    SELECT EXISTS(SELECT 1 FROM supplier WHERE supplier_id = p_supplier_id)
    INTO supplier_exists;

    IF NOT supplier_exists THEN
        p_success := FALSE;
        p_message := 'Ошибка: поставщик с ID ' || p_supplier_id || ' не найден.';
        RETURN;
    END IF;

    IF p_price <= 0 THEN
        p_success := FALSE;
        p_message := 'Ошибка: цена должна быть больше 0.';
        RETURN;
    END IF;

    INSERT INTO spare_part (title, specifications, price, supplier_id)
    VALUES (p_title, p_specifications, p_price, p_supplier_id)
    RETURNING spare_part_id INTO v_spare_part_id;

    INSERT INTO warehouse (spare_part_id)
    VALUES (v_spare_part_id);

    p_success := TRUE;
    p_message := 'Запчасть успешно добавлена. ID: ' || v_spare_part_id;
END;
$$
```

Рисунок 22 – Скрипт создания процедуры

END :	
Статистика 1	
Name	Value
Updated Rows	0
Execute time	0.016s
Start time	Fri Oct 31 09:44:14 MSK 2025
Finish time	Fri Oct 31 09:44:36 MSK 2025
Query	<pre>-- Процедура для добавления запчастей на склад с проверкой поставщика CREATE OR REPLACE PROCEDURE add_spare_part_to_warehouse(   p_title VARCHAR(100),   p_specifications TEXT,   p_price DECIMAL(12,2),   p_supplier_id INT,   OUT p_success BOOLEAN,   OUT p_message TEXT ) LANGUAGE plpgsql AS \$\$ DECLARE   v_spare_part_id INT;   supplier_exists BOOLEAN; BEGIN   SELECT EXISTS(SELECT 1 FROM supplier WHERE supplier_id = p_supplier_id)   INTO supplier_exists;   IF NOT supplier_exists THEN     p_success := FALSE;     p_message := 'Ошибка: поставщик с ID '    p_supplier_id    ' не найден.';     RETURN;   END IF;   IF p_price &lt;= 0 THEN     p_success := FALSE;     p_message := 'Ошибка: цена должна быть больше 0.';     RETURN;   END IF;   INSERT INTO spare_part (title, specifications, price, supplier_id)   VALUES (p_title, p_specifications, p_price, p_supplier_id)   RETURNING spare_part_id INTO v_spare_part_id;   INSERT INTO warehouse (spare_part_id)   VALUES (v_spare_part_id);   p_success := TRUE;   p_message := 'Запчасть успешно добавлена. ID: '    v_spare_part_id; END;</pre>

Рисунок 23 – Сообщение об успешном создании процедуры

## 2.9 ЗАДАНИЕ 8

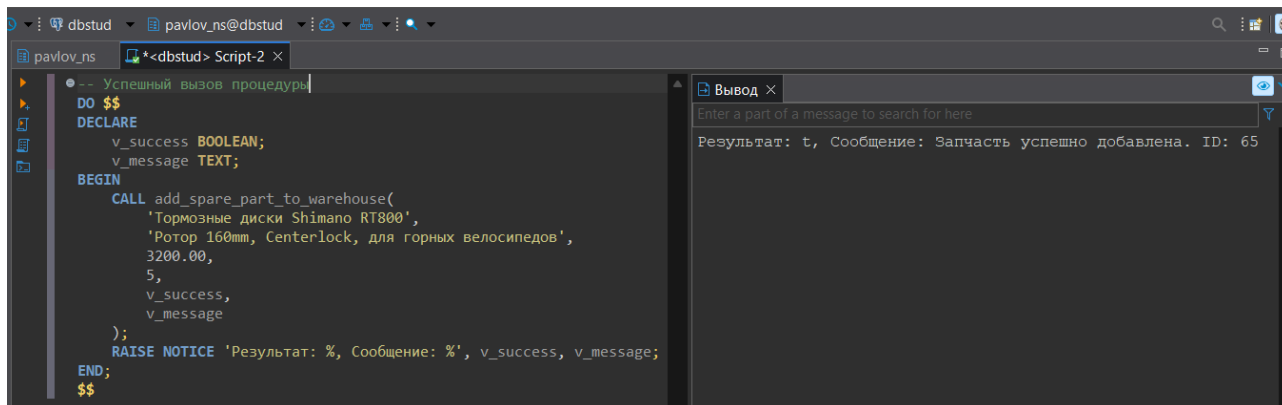


Рисунок 24 – Успешный вызов процедуры

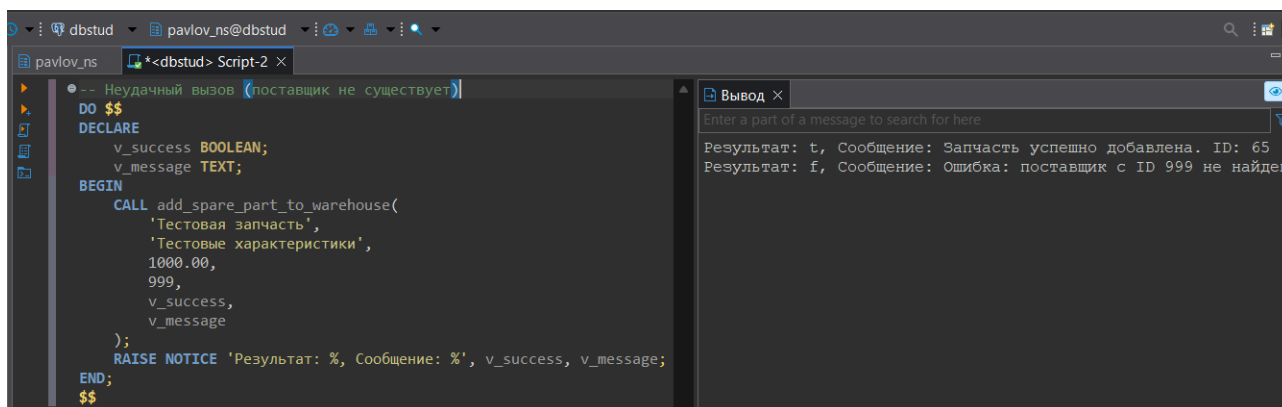


Рисунок 25 – Неудачный вызов процедуры