Министерство науки и высшего образования РФ

федеральное государственное автономное  
образовательное учреждение высшего образования

«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Прикладная математика и фундаментальная информатика»

|  |
| --- |
| **Лабораторная работа №6** |
| по дисциплине **Системы управления базами данных**  **Тема: ПРЕДСТАВЛЕНИЯ** |

|  |  |  |  |
| --- | --- | --- | --- |
| Студента | Кузиной Варвары Михайловны | | |
| Курс | 3 | Группа | МО-231 |
| Направление | 02.03.03 Математическое обеспечение и администрирование информационных систем | | |
| Руководитель | доц., к.н. | | |
|  | Моисеева Н.А. | | |
| Выполнил | 31.10 | | |
| Проверил |  | | |

Омск 2025

**Цель работы:**

Создать в БД rgr\_store несколько простых и материализованных представлений, выполнить к ним три типа запросов: с вложенным подзапросом, с группировкой и простой сортировкой, с группировкой и многоуровневой сортировкой. Результаты подтвердить SQL-скриптами и нумерованными скриншотами.

**Исходные данные:**

БД rgr\_store: employees, suppliers, product\_catalog, orders, order\_composition, receipts, check\_composition, invoices, invoice\_composition, а также расширенные таблицы product\_media (массивы) и receipt\_payments (JSONB). Все связи реализованы внешними ключами с каскадами; заданы ограничения CHECK/NOT NULL/PK.

**1. Простое представление: active\_receipts**

Смысл: Показать чеки за последние 30 дней с именем кассира.

SQL:

-- простое представление

CREATE OR REPLACE VIEW public.active\_receipts AS

SELECT

r.receipt\_number,

r.receipt\_date,

r.cash\_register\_number,

e.emp\_name AS cashier\_name

FROM public.receipts r

JOIN public.employees e ON e.employee\_id = r.employee\_id

WHERE r.receipt\_date >= CURRENT\_DATE - INTERVAL '30 days';

--Проверка и откат

SELECT \* FROM public.product\_sales\_summary;

DROP MATERIALIZED VIEW IF EXISTS public.product\_sales\_summary;

Пояснение:

Представление объединяет чеки и сотрудников, фильтруя по актуальной дате. Используется JOIN и условие WHERE. Результат обновляется при каждом запросе (рисунок 1).

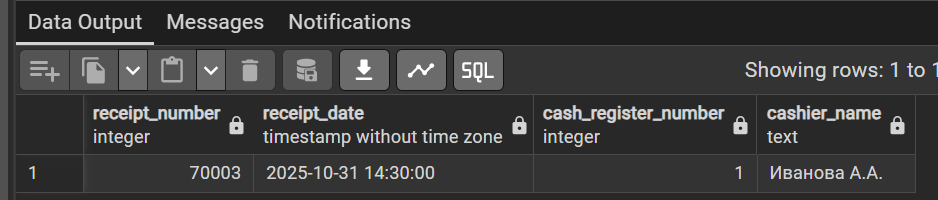


Рисунок 1— Чеки за последние 30 дней

**2. Материализованное представление: product\_sales\_summary**

Смысл: Сводка по товарам: общее количество продано и выручка.

SQL:

--материализованоое представление

CREATE MATERIALIZED VIEW public.product\_sales\_summary AS

SELECT

p.product\_code,

p.product\_name,

p.product\_category,

p.product\_price,

COALESCE(SUM(cc.quantity\_in\_check), 0) AS total\_sold,

COALESCE(SUM(cc.quantity\_in\_check \* p.product\_price), 0) AS total\_revenue

FROM public.product\_catalog p

LEFT JOIN public.check\_composition cc ON cc.product\_code = p.product\_code

GROUP BY p.product\_code, p.product\_name, p.product\_category, p.product\_price;

--Проверка и откат

SELECT \* FROM public.product\_sales\_summary;

DROP MATERIALIZED VIEW IF EXISTS public.product\_sales\_summary;

Пояснение:

Данные агрегируются один раз при создании. Для актуализации требуется REFRESH. Используется LEFT JOIN, чтобы включить даже не продававшиеся товары (рисунок 2).

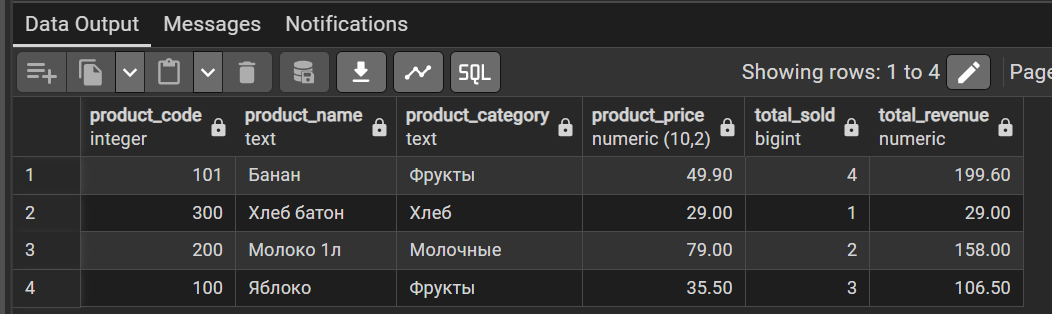


Рисунок 2—Количество проданных товаров(по товарам) и выручка

**3. Материализованное представление: supplier\_category\_stats**

Смысл: Общее количество поставленных товаров по поставщикам и категориям.

SQL:

--материализованное представление

CREATE MATERIALIZED VIEW public.supplier\_invoice\_totals AS

SELECT

s.supplier\_code,

s.supplier\_name,

SUM(ic.product\_quantity) AS total\_supplied

FROM public.suppliers s

JOIN public.invoices i ON i.supplier\_code = s.supplier\_code

JOIN public.invoice\_composition ic ON ic.invoice\_number = i.invoice\_number

GROUP BY s.supplier\_code, s.supplier\_name;

--Проверка и откат

SELECT \* FROM public.supplier\_invoice\_totals;

DROP MATERIALIZED VIEW IF EXISTS public.supplier\_invoice\_totals;

Пояснение:

Представление объединяет 4 таблицы и агрегирует данные по двум измерениям. Используется для многоуровневой сортировки (рисунок 3).

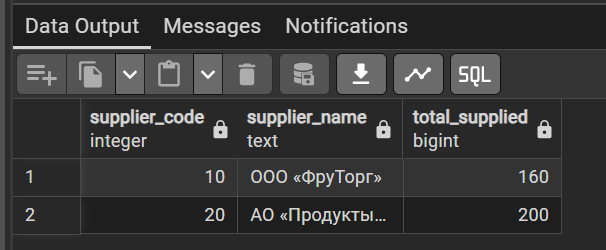


Рисунок 3— Количество товаров(по поставщикам и категориям)

**4. Запрос с вложенным подзапросом**

Смысл: Найти товары, которые продавались чаще среднего.

SQL:

SELECT product\_name, total\_sold

FROM public.product\_sales\_summary

WHERE total\_sold > (

SELECT AVG(total\_sold)

FROM public.product\_sales\_summary

);

Пояснение:

Вложенный скалярный подзапрос вычисляет среднее количество продаж. Внешний запрос отбирает только «популярные» товары (рисунок 4).

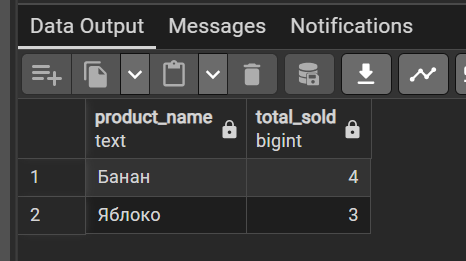


Рисунок 4—Товары, продающиеся больше всего

**5. Запрос с группировкой и простой сортировкой**

Смысл: Показать общую выручку по категориям, отсортированную по убыванию.

SQL:

SELECT

product\_category,

SUM(total\_revenue) AS category\_revenue

FROM public.product\_sales\_summary

GROUP BY product\_category

ORDER BY category\_revenue DESC;

Пояснение:

Группировка по product\_category, агрегация через SUM, сортировка — одна колонка (рисунок 5).

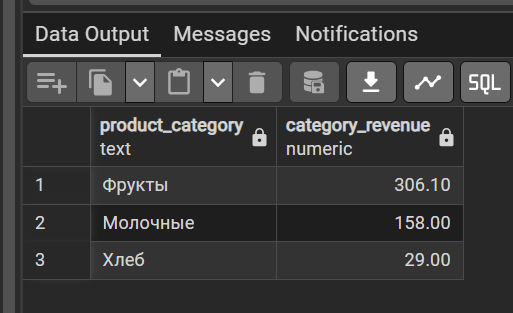


Рисунок 5— Общая выручка по категориям (по убыванию)

**6. Запрос с группировкой и многоуровневой сортировкой**

Смысл: Показать поставщиков и категории товаров с суммарным количеством, отсортировав сначала по имени поставщика (A→Z), затем по количеству (убывание).

SQL:

--запрос с группировкой и многоуровневой сортировкой

SELECT supplier\_name, supplier\_code, total\_supplied

FROM public.supplier\_invoice\_totals

ORDER BY supplier\_name ASC,total\_supplied DESC;

Пояснение:

Запрос к материализованному представлению с двумя уровнями сортировки (рисунок 6).

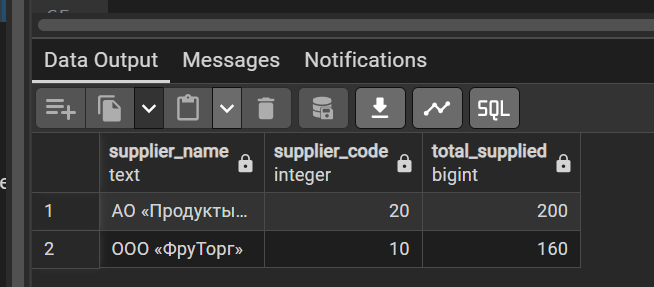


Рисунок 6— Количество товаров по поставщикам и коду товара

**Заключение:**

Для базы rgr\_store созданы одно простое и два материализованных представления. Выполнены три типа запросов: с вложенным подзапросом, с группировкой и простой сортировкой, с группировкой и многоуровневой сортировкой. Все объекты успешно работают в pgAdmin; результаты подтверждены нумерованными скриншотами.