sem10_Nemkov

Task 1

Для каждой процедуры отобразите следующую информацию: дата процедуры, фамилия и имя доктора (в формате Иванов Иван), категория, название, оценка и средний балл по процедурам в той же категории (средний балл процедуры = средний балл среди проведенных процедур за 3 предыдущих дня) (Для решения задания используйте оконные функции)

```
SELECT
  P.PROCEDURE_DATE,
  D.LAST_NAME || ' ' || D.FIRST_NAME,
  P.CATEGORY,
  P.NAME,
  P.SCORE,
  ROUND(
    AVG(P.SCORE) OVER (
      PARTITION BY
        P.CATEGORY
      ORDER BY
        P.PROCEDURE_DATE RANGE BETWEEN INTERVAL '3 days' PRECEDING
        AND INTERVAL '1 day' PRECEDING
    ),
    2
  ) AS avg_score_at_last_3_days
FROM
  PROCEDURES P
  JOIN DOCTORS D ON P.DOCTOR_ID = D.ID
ORDER BY
  P.PROCEDURE_DATE;
```

	procedure_date date	?column? text	category character varying (100)	name character varying (255)	score integer	avg_score_at_last_3_days numeric
1	2025-04-01	Трешкин Никита	Терапия	Общий осмотр	10	[null]
2	2025-04-02	Хрущев Дмитрий	Психиатрия	Психиатрическая консультация	6	[null]
3	2025-04-03	Цветкова Ольга	Терапия	Лечение ОРВИ	10	10.00
4	2025-04-03	Хрущев Дмитрий	Психиатрия	Терапия депрессии	8	6.00
5	2025-04-03	Сергеев Сергей	Эндокринология	Консультация по щитовидной железе	6	[null]
6	2025-04-04	Иванов Сергей	Хирургия	Удаление аппендикса	7	[null]
7	2025-04-05	Маркова Мария	Гинекология	УЗИ органов малого таза	10	[null]
8	2025-04-06	Петров Александр	Офтальмология	Очковая коррекция	6	[null]
9	2025-04-07	Петров Александр	Офтальмология	Очковая коррекция	9	6.00
10	2025-04-08	Селезнева Елена	Хирургия	Удаление аппендикса	7	[null]
11	2025-04-09	Борисов Юрий	Кардиология	УЗИ сердца	7	[null]
12	2025-04-10	Лыкова Светлана	Гинекология	Удаление кисты	10	[null]
13	2025-04-11	Трешкин Никита	Терапия	Контроль хронических заболеваний	6	[null]
14	2025-04-12	Лыкова Светлана	Гинекология	Гинекологический осмотр	9	10.00
15	2025-04-13	Цветкова Ольга	Терапия	Лечение ОРВИ	10	6.00
16	2025-04-14	Иванов Сергей	Хирургия	Операция на плечевом суставе	8	[null]
17	2025-04-15	Цветкова Ольга	Терапия	Контроль хронических заболеваний	9	10.00
18	2025-04-16	Маркова Мария	Гинекология	Лечение миомы	10	[null]
19	2025-04-17	Селезнева Елена	Хирургия	Операция на колене	6	8.00
20	2025-04-18	Иванов Сергей	Хирургия	Удаление опухоли	9	6.00

Total rows: 62 of 62 Query complete 00:00:00.154 Ln 18, Col 1

image-297.png

Task 2

Необходимо создать представление, которое выводит информацию о том, сколько процедур было выполнено каждым врачом, а также общую стоимость процедур, выполненных каждым врачом.

```
CREATE VIEW task_2 AS
SELECT
  D.ID,
  D.LAST_NAME || ' ' || D.FIRST_NAME,
  COUNT(P.ID) as procedures_count,
  SUM(P.PRICE) as procedures_price
FROM
  DOCTORS D
  LEFT JOIN PROCEDURES P ON D.ID = P.DOCTOR_ID
GROUP BY
  D.ID,
  D.LAST_NAME,
  D.FIRST_NAME
ORDER BY
  procedures_count DESC,
  procedures_price DESC;
```

```
SELECT * FROM task_2;
```

	id integer	?column? text	procedures_count bigint	procedures_price numeric
1	6	Хрущев Дмитрий	8	54613.76
2	7	Селезнева Елена	7	59549.25
3	4	Маркова Мария	6	63131.93
4	5	Лыкова Светлана	6	62129.88
5	3	Петров Александр	6	61906.19
6	8	Сергеев Сергей	6	59966.54
7	1	Иванов Сергей	6	54717.66
8	10	Трешкин Никита	6	50531.21
9	9	Цветкова Ольга	6	46104.00
10	2	Борисов Юрий	5	45395.96

Total rows: 10 of 10 Query complete 00:00:00.155 Ln 17, Col 1

image-299.png

Task 3

В клинике акция в честь дня здоровья. Создайте триггер, который будет срабатывать при добавлении записей в таблицу procedures. Каждый раз, при добавлении новой записи в таблицу procedures, для **новых** клиентов должна применяться скидка в 20% на процедуру. Так же должна проводиться запись в таблицу clients_audit (patient_id, procedure_id, old_price, new_price).

```
CREATE TABLE clients_audit (
    id SERIAL PRIMARY KEY,
    patient_id INT NOT NULL,
    procedure_id INT NOT NULL,
    old_price NUMERIC(10, 2) NOT NULL,
    new_price NUMERIC(10, 2) NOT NULL,
    change_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    discount_applied BOOLEAN DEFAULT TRUE
);

CREATE OR REPLACE FUNCTION apply_new_client_discount()
```

```
RETURNS TRIGGER AS $$
DECLARE
    is_new_client BOOLEAN;
    discount_price NUMERIC(10, 2);
BEGIN
    -- Проверяем, является ли пациент новым (нет других процедур)
    SELECT NOT EXISTS (
        SELECT 1 FROM procedures
        WHERE patient_id = NEW.patient_id AND id != NEW.id
    ) INTO is_new_client;

    -- Если пациент новый, применяем скидку 20%
    IF is_new_client THEN
        discount_price := NEW.price * 0.8;

        -- Записываем в аудит-таблицу
        INSERT INTO clients_audit (patient_id, procedure_id, old_price,
new_price)
        VALUES (NEW.patient_id, NEW.id, NEW.price, discount_price);

        -- Обновляем цену в новой записи
        NEW.price := discount_price;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER new_client_discount_trigger
BEFORE INSERT ON procedures
FOR EACH ROW
EXECUTE FUNCTION apply_new_client_discount();
```

Task 4

Расскажите про уровень изоляции repeatable read что такое аномалия «фантомного» чтения.
Почему repeatable read не предотвращает его? Приведите пример.

- REPEATABLE READ *исключает проблему неповторяющегося чтения*. В этом режиме видны только те данные, которые были зафиксированы до начала

транзакции, но не видны незафиксированные данные и изменения, произведённые другими транзакциями в процессе выполнения данной транзакции.

- Но запрос будет видеть эффекты предыдущих изменения в своей транзакции, несмотря на то, что они еще не зафиксированы
- Аномалия фантомного чтения - это аномалия, суть которой заключается в том, что определённая транзакция может *при повторном вызове возвращает какие-то "фантомные" строки, которые появились после выполнения какой-то другой транзакции*
- REPEATABLE READ не предотвращает эту аномалию, поскольку, как я уже написал выше, *в этом режиме не видны незафиксированные данные и изменения, произведённые другими транзакциями в процессе выполнения данной транзакции
- Пример: в базе данных текущего семинара можно выполнить транзакцию, которая выведет количество строк таблицы. Например, она вернет N строк. Далее выполнится другая транзакция, которая выполнит вставку новых P строк. И при следующем выполнении первой транзакции будет получено уже N+P строк. Эти фантомные P строк как раз будут являться последствием фантомного чтения

В чем особенность реализации данного уровня изоляции в Postgres?
Почему в этом случае гарантируется предотвращение аномалий этого рода?

- Особенность реализации в Postgres:

1. Каждая транзакция видит "фотографию" данных на момент старта
2. Система запоминает не только что мы прочитали, но и по каким условиям (например "цена > 1000")
3. При сохранении проверяет:
   - Кто-то изменил прочитанные вами данные? тогда отмена
   - Кто-то добавил строки, подходящие под ваши условия? тогда отмена
   Работает быстро без блокировок, но иногда требует перезапуска транзакции

Task 5

В чем разница между уровнями изоляции repeatable read и serializable?
В каких ситуациях предпочтительнее использовать каждый из уровней?

Приведите пример ситуаций, где использование первого уровня может привести к логическим ошибкам, и способ решения проблемы.

- **REPEATABLE READ** – защищает от изменений уже прочитанных данных, но не видит новые записи "фантомы"
- **SERIALIZABLE** – полная изоляция, которая видит все изменения, но медленнее работает.
  Когда что брать:
- **REPEATABLE READ** – если появление новых строк не ломает логику (например, чтение логов).
- **SERIALIZABLE** – если критично учитывать все изменения (деньги, бронирования).

Пример ошибки в REPEATABLE READ:
Два кассира видят "10 товаров", продают по 8. И получается так, что остаток уходит в минус.
Как исправить:

1. Перейти на SERIALIZABLE.
2. Или добавить FOR UPDATE при чтении.