

МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет
«Московский институт электронной техники»

Лабораторная работа №6
по дисциплине «Системы управления базами данных»
Язык программирования PL/pgSQL. Процедуры, функции, триггеры

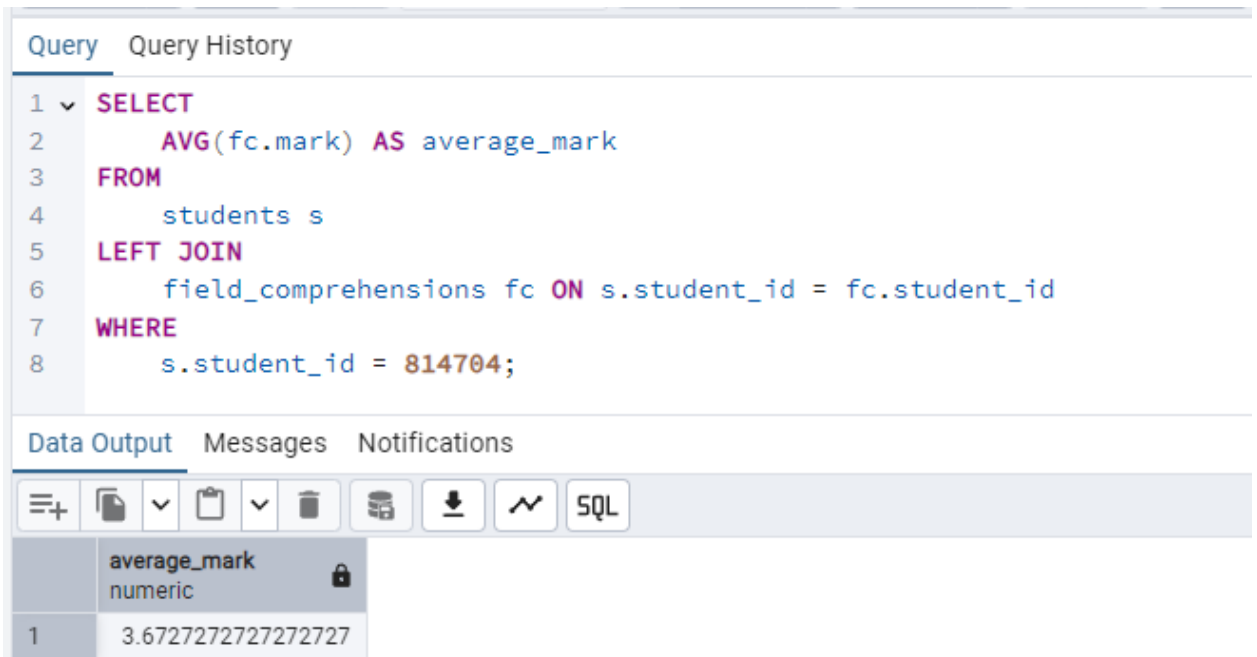
Подготовил:

Студент группы ПИН-32
Трусов М.П.

Москва 2025

Задание 1

Напишите скрипт на языке PL/pgSQL, вычисляющий среднюю оценку студента. Аналогичный запрос напишите на языке SQL. Сравните время выполнения работы в обоих случаях. Для расчета времени выполнения скрипта, запустите его в терминале psql, перед этим запустив таймер с помощью команды \timing. Для того, чтобы отключить таймер после окончания работы, выполните команду \timing off.



The screenshot shows a SQL IDE interface. The top tab is 'Query', and the bottom tab is 'Data Output'. The query editor contains the following SQL code:

```
1 SELECT
2     AVG(fc.mark) AS average_mark
3 FROM
4     students s
5 LEFT JOIN
6     field_comprehensions fc ON s.student_id = fc.student_id
7 WHERE
8     s.student_id = 814704;
```

The 'Data Output' tab shows the result of the query. It has a table with one column, 'average_mark', and one row with the value '3.67272727272727'.

	average_mark
1	3.67272727272727

```
students=# \timing
Timing is on.
students=# SELECT
students-#     AVG(fc.mark) AS average_mark
students-# FROM
students-#     students s
students-# LEFT JOIN
students-#     field_comprehensions fc ON s.student_id = fc.student_id
students-# WHERE
students-#     s.student_id = 814704;
         average_mark
-----
 3.67272727272727
(1 row)

Time: 1,775 ms
students=# \timing off
Timing is off.
students=#
```

```

1 CREATE OR REPLACE FUNCTION calculate_average_mark(p_student_id integer)
2 RETURNS numeric AS $$
3 DECLARE
4     v_avg_mark numeric;
5 BEGIN
6     SELECT AVG(mark) INTO v_avg_mark
7     FROM field_comprehensions
8     WHERE student_id = p_student_id;
9
10    RETURN v_avg_mark;
11 END;
12 $$ LANGUAGE plpgsql;

```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 57 msec.

```

students=# \timing
Timing is on.
students=# SELECT calculate_average_mark(814704);
 calculate_average_mark
-----
          3.67272727272727
(1 row)

Time: 1,360 ms
students=# \timing off
Timing is off.
students=# █

```

Задание 2

Напишите SQL запросы к учебной базе данных в соответствии с вариантом. Вариант к практической части выбирается по формуле: $V = (N \% 10) + 1$, где N – номер в списке группы, % - остаток от деления.

№ варианта	№ запросов
9	9, 19, 29, 39, 49, 59, 69

9. Напишите скрипт определяющий знак зодиака каждого студента. Выведите Фамилию, имя, дату рождения и знак зодиака.

```

1 CREATE OR REPLACE FUNCTION get_zodiac_sign(birth_date date)
2 RETURNS text AS $$
3 BEGIN
4     RETURN CASE
5         WHEN (EXTRACT(MONTH FROM birth_date) = 3 AND EXTRACT(DAY FROM birth_date) >= 21) OR
6              (EXTRACT(MONTH FROM birth_date) = 4 AND EXTRACT(DAY FROM birth_date) <= 19) THEN 'Овен'
7         WHEN (EXTRACT(MONTH FROM birth_date) = 4 AND EXTRACT(DAY FROM birth_date) >= 20) OR
8              (EXTRACT(MONTH FROM birth_date) = 5 AND EXTRACT(DAY FROM birth_date) <= 20) THEN 'Телец'
9         WHEN (EXTRACT(MONTH FROM birth_date) = 5 AND EXTRACT(DAY FROM birth_date) >= 21) OR
10              (EXTRACT(MONTH FROM birth_date) = 6 AND EXTRACT(DAY FROM birth_date) <= 20) THEN 'Близнецы'
11        WHEN (EXTRACT(MONTH FROM birth_date) = 6 AND EXTRACT(DAY FROM birth_date) >= 21) OR
12              (EXTRACT(MONTH FROM birth_date) = 7 AND EXTRACT(DAY FROM birth_date) <= 22) THEN 'Рак'
13        WHEN (EXTRACT(MONTH FROM birth_date) = 7 AND EXTRACT(DAY FROM birth_date) >= 23) OR
14              (EXTRACT(MONTH FROM birth_date) = 8 AND EXTRACT(DAY FROM birth_date) <= 22) THEN 'Лев'
15        WHEN (EXTRACT(MONTH FROM birth_date) = 8 AND EXTRACT(DAY FROM birth_date) >= 23) OR
16              (EXTRACT(MONTH FROM birth_date) = 9 AND EXTRACT(DAY FROM birth_date) <= 22) THEN 'Дева'
17        WHEN (EXTRACT(MONTH FROM birth_date) = 9 AND EXTRACT(DAY FROM birth_date) >= 23) OR
18              (EXTRACT(MONTH FROM birth_date) = 10 AND EXTRACT(DAY FROM birth_date) <= 22) THEN 'Весы'
19        WHEN (EXTRACT(MONTH FROM birth_date) = 10 AND EXTRACT(DAY FROM birth_date) >= 23) OR
20              (EXTRACT(MONTH FROM birth_date) = 11 AND EXTRACT(DAY FROM birth_date) <= 21) THEN 'Скорпион'
21        WHEN (EXTRACT(MONTH FROM birth_date) = 11 AND EXTRACT(DAY FROM birth_date) >= 22) OR
22              (EXTRACT(MONTH FROM birth_date) = 12 AND EXTRACT(DAY FROM birth_date) <= 21) THEN 'Стрелец'
23        WHEN (EXTRACT(MONTH FROM birth_date) = 12 AND EXTRACT(DAY FROM birth_date) >= 22) OR
24              (EXTRACT(MONTH FROM birth_date) = 1 AND EXTRACT(DAY FROM birth_date) <= 19) THEN 'Козерог'
25        WHEN (EXTRACT(MONTH FROM birth_date) = 1 AND EXTRACT(DAY FROM birth_date) >= 20) OR
26              (EXTRACT(MONTH FROM birth_date) = 2 AND EXTRACT(DAY FROM birth_date) <= 18) THEN 'Водолей'
27        WHEN (EXTRACT(MONTH FROM birth_date) = 2 AND EXTRACT(DAY FROM birth_date) >= 19) OR
28              (EXTRACT(MONTH FROM birth_date) = 3 AND EXTRACT(DAY FROM birth_date) <= 20) THEN 'Рыбы'
29        ELSE 'Не определен'
30    END;
31 END;
32 $$ LANGUAGE plpgsql;

```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 91 msec.

```

1 SELECT
2     last_name AS "Фамилия",
3     first_name AS "Имя",
4     birthday AS "Дата рождения",
5     get_zodiac_sign(birthday) AS "Знак зодиака"
6 FROM
7     public.students
8 ORDER BY
9     last_name, first_name;

```

Data Output Messages Notifications

SQL

	Фамилия character varying (30)	Имя character varying (30)	Дата рождения date	Знак зодиака text
1	Абрамов	Платон	2004-03-14	Рыбы
2	Адуев	Александр	2003-10-16	Весы
3	Адуев	Пётр	2003-01-12	Козерог
4	Адуева	Анна	2003-01-08	Козерог
5	Адуева	Лизавета	2003-01-05	Козерог
6	Алексеев	Глеб	2003-03-23	Овен
7	Алексеев	Иван	2003-07-26	Лев
8	Алексеев	Тимофей	2003-09-28	Весы
9	Алехин	Андрей	2003-12-22	Козерог
10	Алешин	Дамир	2001-03-24	Овен
11	Андреев	Николай	2003-12-19	Стрелец

19. Напишите скрипт имитирующий шахматный турнир. Выберите случайным образом 10 студентов. Каждый играет с каждым по одному разу. Результат игры между двумя участниками выбирается рандомно из победы (победивший получает 2 очка, проигравший 0) и ничьи (каждому добавляется по одному баллу). После проведения первого тура отсеиваются 2 участника набравшие наименьшее количество очков. Проводится 2 тур между оставшимися 8 студентами. И так продолжать до тех пор, пока не останутся два победителя. Вывести результаты каждой игры каждого тура (фамилии участников и результат игры) и также для каждого тура итоговые таблицы всех участников с суммой полученных баллов в порядке убывания очков.

```
ЗАМЕЧАНИЕ: --- Начало раунда 1 ---
ЗАМЕЧАНИЕ: Результаты игр раунда 1:
ЗАМЕЧАНИЕ: Бодаев vs Гурмыжская: Победил Гурмыжская
ЗАМЕЧАНИЕ: Бодаев vs Карамазов: Победил Бодаев
ЗАМЕЧАНИЕ: Бодаев vs Трусоцкая: Победил Бодаев
ЗАМЕЧАНИЕ: Бодаев vs Фарафонов: Победил Фарафонов
ЗАМЕЧАНИЕ: Виноградов vs Бодаев: Победил Бодаев
ЗАМЕЧАНИЕ: Виноградов vs Гурмыжская: Победил Гурмыжская
ЗАМЕЧАНИЕ: Виноградов vs Карамазов: Победил Карамазов
ЗАМЕЧАНИЕ: Виноградов vs Кирсанов: Победил Виноградов
ЗАМЕЧАНИЕ: Виноградов vs Красоткин: Победил Виноградов
ЗАМЕЧАНИЕ: Виноградов vs Маслобоев: Победил Маслобоев
ЗАМЕЧАНИЕ: Виноградов vs Огудалова: Победил Виноградов
ЗАМЕЧАНИЕ: Виноградов vs Трусоцкая: Победил Виноградов
ЗАМЕЧАНИЕ: Виноградов vs Фарафонов: Победил Фарафонов
ЗАМЕЧАНИЕ: Гурмыжская vs Карамазов: Победил Гурмыжская
ЗАМЕЧАНИЕ: Гурмыжская vs Трусоцкая: Победил Трусоцкая
ЗАМЕЧАНИЕ: Карамазов vs Трусоцкая: Победил Карамазов
ЗАМЕЧАНИЕ: Кирсанов vs Бодаев: Победил Кирсанов
ЗАМЕЧАНИЕ: Кирсанов vs Гурмыжская: Победил Кирсанов
ЗАМЕЧАНИЕ: Кирсанов vs Карамазов: Победил Кирсанов
ЗАМЕЧАНИЕ: Кирсанов vs Красоткин: Победил Красоткин
ЗАМЕЧАНИЕ: Кирсанов vs Огудалова: Победил Кирсанов
ЗАМЕЧАНИЕ: Кирсанов vs Трусоцкая: Победил Трусоцкая
ЗАМЕЧАНИЕ: Кирсанов vs Фарафонов: Победил Кирсанов
ЗАМЕЧАНИЕ: Красоткин vs Бодаев: Победил Бодаев
ЗАМЕЧАНИЕ: Красоткин vs Гурмыжская: Победил Красоткин
ЗАМЕЧАНИЕ: Красоткин vs Карамазов: Победил Красоткин
ЗАМЕЧАНИЕ: Красоткин vs Огудалова: Победил Огудалова
ЗАМЕЧАНИЕ: Красоткин vs Трусоцкая: Победил Трусоцкая
ЗАМЕЧАНИЕ: Красоткин vs Фарафонов: Победил Красоткин
ЗАМЕЧАНИЕ: Маслобоев vs Бодаев: Победил Бодаев
ЗАМЕЧАНИЕ: Маслобоев vs Гурмыжская: Победил Маслобоев
ЗАМЕЧАНИЕ: Маслобоев vs Карамазов: Победил Маслобоев
ЗАМЕЧАНИЕ: Маслобоев vs Кирсанов: Победил Кирсанов
ЗАМЕЧАНИЕ: Маслобоев vs Красоткин: Победил Красоткин
ЗАМЕЧАНИЕ: Маслобоев vs Огудалова: Победил Маслобоев
ЗАМЕЧАНИЕ: Маслобоев vs Трусоцкая: Победил Трусоцкая
```

ЗАМЕЧАНИЕ:	Огудалова vs Фарафонов: Победил Фарафонов			
ЗАМЕЧАНИЕ:	Фарафонов vs Гурмыжская: Победил Фарафонов			
ЗАМЕЧАНИЕ:	Фарафонов vs Карамазов: Победил Карамазов			
ЗАМЕЧАНИЕ:	Фарафонов vs Трусоцкая: Победил Фарафонов			
ЗАМЕЧАНИЕ:	Итоговая таблица раунда 1:			
ЗАМЕЧАНИЕ:	Место	Фамилия	Имя	Очки
ЗАМЕЧАНИЕ:	-----			
ЗАМЕЧАНИЕ:	1	Кирсанов	Аркадий	12
ЗАМЕЧАНИЕ:	2	Бодаев	Уар	10
ЗАМЕЧАНИЕ:	2	Красоткин	Николай	10
ЗАМЕЧАНИЕ:	2	Маслобоев	Филипп	10
ЗАМЕЧАНИЕ:	2	Трусоцкая	Наталья	10
ЗАМЕЧАНИЕ:	2	Фарафонов	Степан	10
ЗАМЕЧАНИЕ:	7	Виноградов	Иван	8
ЗАМЕЧАНИЕ:	7	Гурмыжская	Раиса	8
ЗАМЕЧАНИЕ:	7	Карамазов	Федор	8
ЗАМЕЧАНИЕ:	10	Огудалова	Лариса	4
ЗАМЕЧАНИЕ:	--- Начало раунда 2 ---			
ЗАМЕЧАНИЕ:	Результаты игр раунда 2:			
ЗАМЕЧАНИЕ:	Бодаев vs Гурмыжская: Победил Гурмыжская			
ЗАМЕЧАНИЕ:	Бодаев vs Карамазов: Победил Карамазов			
ЗАМЕЧАНИЕ:	Бодаев vs Трусоцкая: Победил Бодаев			
ЗАМЕЧАНИЕ:	Бодаев vs Фарафонов: Победил Фарафонов			
ЗАМЕЧАНИЕ:	Гурмыжская vs Карамазов: Победил Гурмыжская			
ЗАМЕЧАНИЕ:	Гурмыжская vs Трусоцкая: Победил Трусоцкая			
ЗАМЕЧАНИЕ:	Карамазов vs Трусоцкая: Победил Карамазов			
ЗАМЕЧАНИЕ:	Кирсанов vs Бодаев: Победил Кирсанов			
ЗАМЕЧАНИЕ:	Кирсанов vs Гурмыжская: Победил Кирсанов			
ЗАМЕЧАНИЕ:	Кирсанов vs Карамазов: Победил Карамазов			
ЗАМЕЧАНИЕ:	Кирсанов vs Красоткин: Победил Кирсанов			
ЗАМЕЧАНИЕ:	Кирсанов vs Трусоцкая: Победил Кирсанов			
ЗАМЕЧАНИЕ:	Кирсанов vs Фарафонов: Победил Кирсанов			
ЗАМЕЧАНИЕ:	Красоткин vs Бодаев: Победил Красоткин			
ЗАМЕЧАНИЕ:	Красоткин vs Гурмыжская: Победил Красоткин			
ЗАМЕЧАНИЕ:	Красоткин vs Карамазов: Победил Красоткин			
ЗАМЕЧАНИЕ:	Красоткин vs Трусоцкая: Победил Трусоцкая			
ЗАМЕЧАНИЕ:	Красоткин vs Фарафонов: Победил Красоткин			
ЗАМЕЧАНИЕ:	Маслобоев vs Бодаев: Победил Маслобоев			
ЗАМЕЧАНИЕ:	Маслобоев vs Гурмыжская: Победил Гурмыжская			
ЗАМЕЧАНИЕ:	Маслобоев vs Карамазов: Победил Карамазов			
ЗАМЕЧАНИЕ:	Фарафонов vs Карамазов: Победил Карамазов			
ЗАМЕЧАНИЕ:	Фарафонов vs Трусоцкая: Победил Фарафонов			
ЗАМЕЧАНИЕ:	Итоговая таблица раунда 2:			
ЗАМЕЧАНИЕ:	Место	Фамилия	Имя	Очки
ЗАМЕЧАНИЕ:	-----			
ЗАМЕЧАНИЕ:	1	Кирсанов	Аркадий	12
ЗАМЕЧАНИЕ:	2	Карамазов	Федор	10
ЗАМЕЧАНИЕ:	3	Гурмыжская	Раиса	8
ЗАМЕЧАНИЕ:	3	Красоткин	Николай	8
ЗАМЕЧАНИЕ:	3	Маслобоев	Филипп	8
ЗАМЕЧАНИЕ:	6	Трусоцкая	Наталья	4
ЗАМЕЧАНИЕ:	6	Фарафонов	Степан	4
ЗАМЕЧАНИЕ:	8	Бодаев	Уар	2
ЗАМЕЧАНИЕ:	--- Начало раунда 3 ---			
ЗАМЕЧАНИЕ:	Результаты игр раунда 3:			
ЗАМЕЧАНИЕ:	Карамазов vs Гурмыжская: Победил Гурмыжская			
ЗАМЕЧАНИЕ:	Карамазов vs Кирсанов: Победил Кирсанов			
ЗАМЕЧАНИЕ:	Карамазов vs Красоткин: Победил Карамазов			
ЗАМЕЧАНИЕ:	Карамазов vs Маслобоев: Победил Маслобоев			
ЗАМЕЧАНИЕ:	Карамазов vs Фарафонов: Победил Фарафонов			
ЗАМЕЧАНИЕ:	Кирсанов vs Гурмыжская: Победил Гурмыжская			
ЗАМЕЧАНИЕ:	Кирсанов vs Красоткин: Победил Красоткин			
ЗАМЕЧАНИЕ:	Кирсанов vs Маслобоев: Победил Кирсанов			
ЗАМЕЧАНИЕ:	Кирсанов vs Фарафонов: Победил Кирсанов			
ЗАМЕЧАНИЕ:	Красоткин vs Гурмыжская: Победил Гурмыжская			
ЗАМЕЧАНИЕ:	Красоткин vs Маслобоев: Победил Маслобоев			
ЗАМЕЧАНИЕ:	Маслобоев vs Гурмыжская: Победил Маслобоев			
ЗАМЕЧАНИЕ:	Фарафонов vs Гурмыжская: Победил Фарафонов			
ЗАМЕЧАНИЕ:	Фарафонов vs Красоткин: Победил Фарафонов			
ЗАМЕЧАНИЕ:	Фарафонов vs Маслобоев: Победил Фарафонов			
ЗАМЕЧАНИЕ:	Итоговая таблица раунда 3:			
ЗАМЕЧАНИЕ:	Место	Фамилия	Имя	Очки
ЗАМЕЧАНИЕ:	-----			
ЗАМЕЧАНИЕ:	1	Фарафонов	Степан	8
ЗАМЕЧАНИЕ:	2	Гурмыжская	Раиса	6
ЗАМЕЧАНИЕ:	2	Кирсанов	Аркадий	6
ЗАМЕЧАНИЕ:	2	Маслобоев	Филипп	6
ЗАМЕЧАНИЕ:	5	Карамазов	Федор	2
ЗАМЕЧАНИЕ:	5	Красоткин	Николай	2

```

ЗАМЕЧАНИЕ: Итоговая таблица раунда 3:
ЗАМЕЧАНИЕ: Место | Фамилия          | Имя          | Очки
ЗАМЕЧАНИЕ: -----+-----+-----+-----
ЗАМЕЧАНИЕ:      1 | Фарафонов      | Степан       | 8
ЗАМЕЧАНИЕ:      2 | Гурмыжская     | Раиса        | 6
ЗАМЕЧАНИЕ:      2 | Кирсанов       | Аркадий      | 6
ЗАМЕЧАНИЕ:      2 | Маслобоев      | Филипп       | 6
ЗАМЕЧАНИЕ:      5 | Карамазов      | Федор        | 2
ЗАМЕЧАНИЕ:      5 | Красоткин      | Николай      | 2
ЗАМЕЧАНИЕ: --- Начало раунда 4 ---
ЗАМЕЧАНИЕ: Результаты игр раунда 4:
ЗАМЕЧАНИЕ: Кирсанов vs Гурмыжская: Победил Гурмыжская
ЗАМЕЧАНИЕ: Кирсанов vs Маслобоев: Победил Маслобоев
ЗАМЕЧАНИЕ: Кирсанов vs Фарафонов: Победил Фарафонов
ЗАМЕЧАНИЕ: Маслобоев vs Гурмыжская: Победил Гурмыжская
ЗАМЕЧАНИЕ: Фарафонов vs Гурмыжская: Победил Фарафонов
ЗАМЕЧАНИЕ: Фарафонов vs Маслобоев: Победил Маслобоев
ЗАМЕЧАНИЕ: Итоговая таблица раунда 4:
ЗАМЕЧАНИЕ: Место | Фамилия          | Имя          | Очки
ЗАМЕЧАНИЕ: -----+-----+-----+-----
ЗАМЕЧАНИЕ:      1 | Гурмыжская     | Раиса        | 4
ЗАМЕЧАНИЕ:      1 | Маслобоев      | Филипп       | 4
ЗАМЕЧАНИЕ:      1 | Фарафонов      | Степан       | 4
ЗАМЕЧАНИЕ:      4 | Кирсанов       | Аркадий      | 0
ЗАМЕЧАНИЕ: --- ФИНАЛЬНЫЙ МАТЧ ---
ЗАМЕЧАНИЕ: Степан Фарафонов vs Филипп Маслобоев: Победил Маслобоев
ЗАМЕЧАНИЕ: --- ФИНАЛЬНАЯ ТАБЛИЦА ---
ЗАМЕЧАНИЕ: Место | Фамилия          | Имя          | Очки
ЗАМЕЧАНИЕ: -----+-----+-----+-----
ЗАМЕЧАНИЕ:      1 | Маслобоев      | Филипп       | 2
ЗАМЕЧАНИЕ:      2 | Фарафонов      | Степан       | 0
ЗАМЕЧАНИЕ: ПОБЕДИТЕЛЬ ТУРНИРА: Филипп Маслобоев!
DO

```

29. Создайте процедуру продления студенческих билетов у определенной группы на 1 год. Входной параметр - номер группы.

```

Query  Query History
1  CREATE OR REPLACE PROCEDURE extend_student_ids_for_group(
2      group_number VARCHAR(7))
3  LANGUAGE plpgsql
4  AS $$
5  DECLARE
6      updated_count INTEGER;
7  BEGIN
8      -- Обновляем дату истечения для всех студентов указанной группы
9      UPDATE public.student_ids
10     SET expiration_date = expiration_date + INTERVAL '1 year'
11     WHERE student_id IN (
12         SELECT student_id
13         FROM public.students
14         WHERE students_group_number = group_number
15     );
16 END;
17 $$;

```

Data Output Messages Notifications

CREATE PROCEDURE

Query returned successfully in 59 msec.

Query
Query History

```

1 SELECT * FROM public.student_ids
2 ORDER BY student_id ASC LIMIT 100
3

```

Data Output
Messages
Notifications

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	student_id [PK] integer	issue_date date	expiration_date date
1	812507	2021-08-31	2028-08-31
2	812513	2022-08-31	2026-08-31
3	812640	2023-08-31	2027-08-31
4	812850	2020-08-31	2024-08-31

	student_id [PK] integer	last_name character varying (30)	first_name character varying (30)	patronymic character varying (30)	students_group_number character varying (7)	birthday date	email character varying (30)
1	812507	Мамаева	Клеопатра	Львовна	ИТД-31	2003-03-28	MamaevaKleopatra@miet.ru
2	812513	Бубнова	Анна	Трифоновна	ИБТ-23	2004-02-17	BubnovaAnna@miet.ru

```

students=# CALL extend_student_ids_for_group(U&'\'0418\0422\0414\002D\0033\0031');
CALL

```

Query
Query History

```

1 SELECT * FROM public.student_ids
2 ORDER BY student_id ASC LIMIT 100
3

```

Data Output
Messages
Notifications

+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

	student_id [PK] integer	issue_date date	expiration_date date
1	812507	2021-08-31	2029-08-31
2	812513	2022-08-31	2026-08-31
3	812640	2023-08-31	2027-08-31

39. Создайте функцию, которая считает количество преподавателей, в структурном подразделении с номером N. N вводится в качестве параметра.

Query Query History

```

1 CREATE OR REPLACE FUNCTION count_professors_in_unit(unit_id INTEGER)
2 RETURNS INTEGER AS $$
3 DECLARE
4     professor_count INTEGER;
5     unit_exists BOOLEAN;
6 BEGIN
7     -- Проверяем существование структурного подразделения
8     SELECT EXISTS(
9         SELECT 1 FROM public.structural_units
10        WHERE structural_unit_id = unit_id
11    ) INTO unit_exists;
12
13     IF NOT unit_exists THEN
14         RAISE EXCEPTION 'Структурное подразделение с ID % не существует', unit_id;
15     END IF;
16
17     -- Считаем количество преподавателей в подразделении
18     SELECT COUNT(*) INTO professor_count
19     FROM public.employments
20     WHERE structural_unit_id = unit_id;
21
22     RETURN professor_count;
23 END;
24 $$ LANGUAGE plpgsql;

```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 424 msec.

```

1 SELECT count_professors_in_unit(1) AS professor_count;
2

```

Data Output Messages Notifications

+

📄

▼

📋

▼

🗑️

🗑️

⬇️

⤴️

SQL

	professor_count integer
1	19

```

1 SELECT count_professors_in_unit(1111) AS professor_count;
2

```

Data Output Messages Notifications

ERROR: Структурное подразделение с ID 1111 не существует

CONTEXT: функция PL/pgSQL count_professors_in_unit(integer), строка 13, оператор RAISE

ОШИБКА: Структурное подразделение с ID 1111 не существует

SQL state: P0001

49. Создайте функцию, выводящую всех преподавателей, преподающих в определенном структурном подразделении.

Query

Query History

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

```
CREATE OR REPLACE FUNCTION get_professors_by_unit(unit_id INTEGER)
RETURNS TABLE (
    professor_id INTEGER,
    last_name VARCHAR(30),
    first_name VARCHAR(30),
    patronymic VARCHAR(15)
) AS $$
BEGIN
    -- Проверяем существование структурного подразделения
    IF NOT EXISTS (
        SELECT 1 FROM public.structural_units
        WHERE structural_unit_id = unit_id
    ) THEN
        RAISE EXCEPTION 'Структурное подразделение с ID % не существует', unit_id;
    END IF;

    -- Возвращаем всех преподавателей подразделения
    RETURN QUERY
    SELECT
        p.professor_id,
        p.last_name,
        p.first_name,
        p.patronymic
    FROM
        public.professors p
    JOIN
        public employments e ON p.professor_id = e.professor_id
    WHERE
        e.structural_unit_id = unit_id
    ORDER BY
        p.last_name, p.first_name, p.patronymic;
END;
$$ LANGUAGE plpgsql;
```

Data Output

Messages

Notifications

CREATE FUNCTION

Query returned successfully in 81 msec.

1

SELECT * FROM get_professors_by_unit(1);

Data Output

Messages

Notifications

59. Создайте триггер, запрещающий ставить студенту больше 4 двоек.

Query	Query History
<pre> 1 CREATE OR REPLACE FUNCTION check_dvoiki_limit() 2 RETURNS TRIGGER AS \$\$ 3 DECLARE 4 dvoiki_count INTEGER; 5 BEGIN 6 -- Если оценка не 2, пропускаем проверку 7 IF NEW.mark <> 2 THEN 8 RETURN NEW; 9 END IF; 10 11 -- Считаем количество двоек у студента (включая новую) 12 SELECT COUNT(*) INTO dvoiki_count 13 FROM public.field_comprehensions 14 WHERE student_id = NEW.student_id AND mark = 2; 15 16 -- Если двоек уже 4 или больше (с учетом новой), отменяем операцию 17 IF dvoiki_count >= 4 THEN 18 RAISE EXCEPTION 'Студент с ID % уже имеет % двоек. Нельзя поставить более 4 двоек одному студенту.', 19 NEW.student_id, dvoiki_count; 20 END IF; 21 22 RETURN NEW; 23 END; 24 \$\$ LANGUAGE plpgsql; 25 26 CREATE TRIGGER prevent_too_many_dvoiki 27 BEFORE INSERT OR UPDATE ON public.field_comprehensions 28 FOR EACH ROW EXECUTE FUNCTION check_dvoiki_limit(); </pre>	
Data Output	Messages Notifications
<p>CREATE TRIGGER</p> <p>Query returned successfully in 151 msec.</p>	

```

1  INSERT INTO public.field_comprehensions (student_id, field, mark)
2  VALUES (812507, '015c5946-4ced-41a1-b13f-f1c2ae683972', 2);

```

Data Output [Messages](#) Notifications

ERROR: Студент с ID 812507 уже имеет 6 двоек. Нельзя поставить более 4 двоек одному студенту.
CONTEXT: функция PL/pgSQL check_dvoiki_limit(), строка 17, оператор RAISE

ОШИБКА: Студент с ID 812507 уже имеет 6 двоек. Нельзя поставить более 4 двоек одному студенту.
SQL state: P0001

69. Создайте триггер, сохраняющий информацию об изменениях оценок у студентов.

[Query](#) [Query History](#)

```

1  CREATE TABLE IF NOT EXISTS grade_change_log (
2      log_id SERIAL PRIMARY KEY,
3      student_id INTEGER NOT NULL,
4      field UUID NOT NULL,
5      old_mark INTEGER,
6      new_mark INTEGER,
7      change_time TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
8      changed_by TEXT DEFAULT current_user
9  );
10
11 CREATE OR REPLACE FUNCTION log_grade_changes()
12 RETURNS TRIGGER AS $$
13 BEGIN
14     -- Для операции UPDATE (изменение оценки)
15     IF TG_OP = 'UPDATE' AND OLD.mark IS DISTINCT FROM NEW.mark THEN
16         INSERT INTO grade_change_log (student_id, field, old_mark, new_mark)
17         VALUES (OLD.student_id, OLD.field, OLD.mark, NEW.mark);
18
19     -- Для операции INSERT (новая оценка)
20     ELSEIF TG_OP = 'INSERT' THEN
21         INSERT INTO grade_change_log (student_id, field, old_mark, new_mark)
22         VALUES (NEW.student_id, NEW.field, NULL, NEW.mark);
23
24     -- Для операции DELETE (удаление оценки)
25     ELSEIF TG_OP = 'DELETE' THEN
26         INSERT INTO grade_change_log (student_id, field, old_mark, new_mark)
27         VALUES (OLD.student_id, OLD.field, OLD.mark, NULL);
28     END IF;
29
30     RETURN NEW;
31 END;
32 $$ LANGUAGE plpgsql;

```

Data Output [Messages](#) Notifications

CREATE FUNCTION

Query returned successfully in 256 msec.

Query Query History

```
1  CREATE TRIGGER track_grade_changes
2  AFTER INSERT OR UPDATE OR DELETE ON public.field_comprehensions
3  FOR EACH ROW EXECUTE FUNCTION log_grade_changes();
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 81 msec.

Query Query History

```
1  INSERT INTO public.field_comprehensions (student_id, field, mark)
2  VALUES (812507, '2032f03a-1d84-4c9a-8201-3d8d064d7109', 5);
```

Data Output Messages Notifications

INSERT 0 1

Query Query History

```
1  UPDATE public.field_comprehensions
2  SET mark = 4
3  WHERE student_id = 812507 AND field = '02accd14-abf8-40f9-9a58-6820e94de810';
```

Data Output Messages Notifications

UPDATE 1

Query Query History

```
1  DELETE FROM public.field_comprehensions
2  WHERE student_id = 812507 AND field = '02accd14-abf8-40f9-9a58-6820e94de810';
```

Data Output Messages Notifications

DELETE 1

Query returned successfully in 85 msec.

Query

Query History

1

SELECT * FROM grade_change_log ORDER BY change_time DESC;

Data Output

Messages

Notifications

SQL

	log_id [PK] integer	student_id integer	field uuid	old_mark integer	new_mark integer	change_time timestamp with time zone	changed_by text
1	3	812507	02accd14-abf8-40f9-9a58-6820e94de810	4	[null]	2025-05-09 13:06:36.911696+03	postgres
2	2	812507	02accd14-abf8-40f9-9a58-6820e94de810	5	4	2025-05-09 13:06:23.767648+03	postgres
3	1	812507	2032f03a-1d84-4c9a-8201-3d8d064d7109	[null]	5	2025-05-09 13:05:39.902964+03	postgres

Задание 3.

Для добавленной в 4-й лабораторной работе таблицы создайте любой триггер.

Query Query History	
1	<code>-- Создание таблицы для логов</code>
2	<code>CREATE TABLE IF NOT EXISTS public.student_achievements_log (</code>
3	<code> log_id SERIAL PRIMARY KEY,</code>
4	<code> operation VARCHAR(10) NOT NULL,</code>
5	<code> achievement_id INTEGER NOT NULL,</code>
6	<code> changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,</code>
7	<code> changed_by VARCHAR(100) DEFAULT CURRENT_USER</code>
8	<code>);</code>
9	
10	<code>-- Функция триггера для логирования</code>
11	<code>CREATE OR REPLACE FUNCTION log_student_achievement_changes()</code>
12	<code>RETURNS TRIGGER AS \$\$</code>
13	<code>BEGIN</code>
14	<code> IF TG_OP = 'DELETE' THEN</code>
15	<code> INSERT INTO public.student_achievements_log (operation, achievement_id)</code>
16	<code> VALUES ('DELETE', OLD.achievement_id);</code>
17	<code> ELSIF TG_OP = 'UPDATE' THEN</code>
18	<code> INSERT INTO public.student_achievements_log (operation, achievement_id)</code>
19	<code> VALUES ('UPDATE', NEW.achievement_id);</code>
20	<code> ELSIF TG_OP = 'INSERT' THEN</code>
21	<code> INSERT INTO public.student_achievements_log (operation, achievement_id)</code>
22	<code> VALUES ('INSERT', NEW.achievement_id);</code>
23	<code> END IF;</code>
24	
25	<code> RETURN NULL;</code>
26	<code>END;</code>
27	<code>\$\$ LANGUAGE plpgsql;</code>
28	
29	<code>-- Создание триггера логирования</code>
30	<code>CREATE TRIGGER tr_log_student_achievement_changes</code>
31	<code>AFTER INSERT OR UPDATE OR DELETE ON public.student_achievements</code>
32	<code>FOR EACH ROW EXECUTE FUNCTION log_student_achievement_changes();</code>
Data Output Messages Notifications	
CREATE TRIGGER	
Query returned successfully in 105 msec.	

1	SELECT * FROM public.student_achievements
2	ORDER BY achievement_id ASC

Data Output	Messages	Notifications
-------------	----------	---------------

	achievement_id [PK] integer	portfolio_id integer	achievement_type_id integer
1	1	1	1
2	2	1	3
3	3	2	2
4	4	3	4
5	5	3	5
6	6	4	6
7	7	5	7
8	8	6	8
9	9	7	9
10	10	10	10

Query	Query History
1	INSERT INTO student_achievements (achievement_id, portfolio_id, achievement_type_id)
2	VALUES (11, 1, 3);
3	
4	UPDATE student_achievements
5	SET achievement_type_id = 5
6	WHERE achievement_id = 11;
7	
8	DELETE FROM student_achievements
9	WHERE achievement_id = 11;

Data Output	Messages	Notifications
-------------	----------	---------------

DELETE 1
Query returned successfully in 129 msec.

Query	Query History
1	SELECT * FROM public.student_achievements_log
2	ORDER BY log_id ASC

Data Output	Messages	Notifications
-------------	----------	---------------

	log_id [PK] integer	operation character varying (10)	achievement_id integer	changed_at timestamp without time zone	changed_by character varying (100)
1	1	INSERT	11	2025-05-09 13:33:13.856603	postgres
2	2	UPDATE	11	2025-05-09 13:33:13.856603	postgres
3	3	DELETE	11	2025-05-09 13:33:13.856603	postgres

Приложение. Задание 19.

-- Создаем временные таблицы перед началом блока

```
CREATE TEMP TABLE IF NOT EXISTS tournament_results (  
    round INT,  
    player_id INT,  
    last_name VARCHAR(30),  
    first_name VARCHAR(30),  
    points INT DEFAULT 0,  
    PRIMARY KEY (round, player_id)  
);
```

```
CREATE TEMP TABLE IF NOT EXISTS game_history (  
    round INT,  
    player1_id INT,  
    player1_last_name VARCHAR(30),  
    player2_id INT,  
    player2_last_name VARCHAR(30),  
    result VARCHAR(50),  
    PRIMARY KEY (round, player1_id, player2_id)  
);
```

```
TRUNCATE TABLE tournament_results;  
TRUNCATE TABLE game_history;
```

DO \$\$

DECLARE

```
    tournament_round INT := 1;  
    current_players INT := 10;  
    players_to_eliminate INT := 2;  
    player1 RECORD;  
    player2 RECORD;  
    game_result INT;  
    remaining_players INT[];  
    all_players INT[];  
    player_score INT;  
    pos INT;  
    game_rec RECORD;  
    player_data RECORD;
```

BEGIN


```

-- Выбираем 10 случайных студентов для участия в турнире
INSERT INTO tournament_results (round, player_id, last_name, first_name, points)
SELECT
    1,
    student_id,
    last_name,
    first_name,
    0
FROM public.students
ORDER BY random()
LIMIT 10;

-- Получаем список всех игроков первого раунда
SELECT array_agg(player_id) INTO all_players
FROM tournament_results
WHERE round = 1;

-- Основной цикл турнира
WHILE current_players > 2 LOOP
    RAISE NOTICE '--- Начало раунда % ---', tournament_round;

    -- Проводим игры между всеми участниками текущего раунда
    FOR i IN 1..array_length(all_players, 1) LOOP
        FOR j IN (i+1)..array_length(all_players, 1) LOOP
            -- Получаем данные игроков
            SELECT INTO player1 * FROM tournament_results
            WHERE round = tournament_round AND player_id = all_players[i];

            SELECT INTO player2 * FROM tournament_results
            WHERE round = tournament_round AND player_id = all_players[j];

            -- Определяем результат игры случайным образом
            game_result := floor(random() * 2); -- 0 - победа 1го, 1 - победа 2го

            -- Записываем результат игры в историю
            IF game_result = 0 THEN
                INSERT INTO game_history VALUES (
                    tournament_round,
                    player1.player_id,

```

```

        player1.last_name,
        player2.player_id,
        player2.last_name,
        'Победил ' || player1.last_name
    );

    -- Обновляем очки
    UPDATE tournament_results
    SET points = points + 2
    WHERE round = tournament_round AND player_id = player1.player_id;
ELSE
    INSERT INTO game_history VALUES (
        tournament_round,
        player1.player_id,
        player1.last_name,
        player2.player_id,
        player2.last_name,
        'Победил ' || player2.last_name
    );

    UPDATE tournament_results
    SET points = points + 2
    WHERE round = tournament_round AND player_id = player2.player_id;
END IF;
END LOOP;
END LOOP;

-- Выводим результаты игр текущего раунда
RAISE NOTICE 'Результаты игр раунда %:', tournament_round;
FOR game_rec IN SELECT * FROM game_history WHERE round = tournament_round ORDER
BY player1_last_name, player2_last_name LOOP
    RAISE NOTICE '% vs %: %',
        game_rec.player1_last_name,
        game_rec.player2_last_name,
        game_rec.result;
END LOOP;

-- Выводим таблицу результатов текущего раунда с правильным порядком при равных
очках
RAISE NOTICE 'Итоговая таблица раунда %:', tournament_round;

```

```

RAISE NOTICE 'Место | Фамилия          | Имя          | Очки';
RAISE NOTICE '-----+-----+-----+-----';

pos := 1;
FOR player_data IN
    SELECT
        last_name,
        first_name,
        points,
        rank() OVER (ORDER BY points DESC) as position
    FROM tournament_results
    WHERE round = tournament_round
    ORDER BY points DESC, last_name, first_name
LOOP
    RAISE NOTICE '% | % | % | %',
        lpad(player_data.position::text, 5),
        rpad(player_data.last_name, 13),
        rpad(player_data.first_name, 12),
        player_data.points;
END LOOP;

-- Определяем игроков, которые выбывают
players_to_eliminate := 2;

-- Получаем список игроков для следующего раунда (исключаем худших)
WITH eliminated AS (
    SELECT player_id
    FROM tournament_results
    WHERE round = tournament_round
    ORDER BY points, last_name, first_name
    LIMIT players_to_eliminate
)
SELECT array_agg(player_id) INTO remaining_players
FROM tournament_results
WHERE round = tournament_round AND player_id NOT IN (SELECT player_id FROM
eliminated);

-- Уменьшаем количество текущих игроков
current_players := current_players - players_to_eliminate;

```

```

-- Если осталось больше 2 игроков, готовим следующий раунд
IF current_players >= 2 THEN
    tournament_round := tournament_round + 1;

    -- Копируем оставшихся игроков в следующий раунд с нулевыми очками
    INSERT INTO tournament_results (round, player_id, last_name, first_name,
points)
    SELECT
        tournament_round,
        tr.player_id,
        tr.last_name,
        tr.first_name,
        0
    FROM tournament_results tr
    WHERE tr.round = tournament_round - 1 AND tr.player_id =
ANY(remaining_players);

    -- Обновляем список игроков для следующего раунда
    all_players := remaining_players;
END IF;
END LOOP;

-- Проводим финальный матч между последними двумя игроками
IF current_players = 2 THEN
    tournament_round := tournament_round + 1;

    -- Копируем финалистов в новый раунд
    INSERT INTO tournament_results (round, player_id, last_name, first_name, points)
    SELECT
        tournament_round,
        tr.player_id,
        tr.last_name,
        tr.first_name,
        0
    FROM tournament_results tr
    WHERE tr.round = tournament_round - 1
    ORDER BY tr.points DESC, tr.last_name, tr.first_name;

    -- Получаем данные финалистов
    SELECT INTO player1 * FROM tournament_results

```

```

WHERE round = tournament_round
ORDER BY player_id LIMIT 1;

SELECT INTO player2 * FROM tournament_results
WHERE round = tournament_round
ORDER BY player_id DESC LIMIT 1;

-- Проводим финальный матч
game_result := floor(random() * 2);

IF game_result = 0 THEN
    INSERT INTO game_history VALUES (
        tournament_round,
        player1.player_id,
        player1.last_name,
        player2.player_id,
        player2.last_name,
        'ФИНАЛ: Победил ' || player1.last_name
    );

    UPDATE tournament_results
    SET points = points + 2
    WHERE round = tournament_round AND player_id = player1.player_id;
ELSE
    INSERT INTO game_history VALUES (
        tournament_round,
        player1.player_id,
        player1.last_name,
        player2.player_id,
        player2.last_name,
        'ФИНАЛ: Победил ' || player2.last_name
    );

    UPDATE tournament_results
    SET points = points + 2
    WHERE round = tournament_round AND player_id = player2.player_id;
END IF;

-- Выводим результат финального матча

```

```

RAISE NOTICE '--- ФИНАЛЬНЫЙ МАТЧ ---';
IF game_result = 0 THEN
    RAISE NOTICE '% % vs % %: Победил %',
        player1.first_name, player1.last_name,
        player2.first_name, player2.last_name,
        player1.last_name;
ELSE
    RAISE NOTICE '% % vs % %: Победил %',
        player1.first_name, player1.last_name,
        player2.first_name, player2.last_name,
        player2.last_name;
END IF;
END IF;

-- Выводим финальную таблицу
RAISE NOTICE '--- ФИНАЛЬНАЯ ТАБЛИЦА ---';
RAISE NOTICE 'Место | Фамилия          | Имя          | Очки';
RAISE NOTICE '-----+-----+-----+-----';

pos := 1;
FOR player_data IN
    WITH final_results AS (
        SELECT
            last_name,
            first_name,
            points,
            rank() OVER (ORDER BY points DESC) as position
        FROM tournament_results
        WHERE round = tournament_round
        ORDER BY points DESC, last_name, first_name
    )
    SELECT
        position,
        last_name,
        first_name,
        points
    FROM final_results
LOOP
    RAISE NOTICE '% | % | % | %',

```

```

        lpad(player_data.position::text, 5),
        rpad(player_data.last_name, 13),
        rpad(player_data.first_name, 12),
        player_data.points;
END LOOP;

-- Выводим победителя
SELECT INTO player1 * FROM tournament_results
WHERE round = tournament_round
ORDER BY points DESC, last_name, first_name
LIMIT 1;

RAISE NOTICE 'ПОБЕДИТЕЛЬ ТУРНИРА: % %!', player1.first_name, player1.last_name;
END $$;
```