# Задание 4. Создайте хранимые процедуры:

#### 1. Для повышения стипендии отличникам на 10%

Сначала создадим функцию, которая поможет нам посчитать этот средний балл для каждого студента, и занесем его в столбец average\_grade:

```
CREATE OR REPLACE FUNCTION calculate average grade()
RETURNS void AS $$
DECLARE
  student id INTEGER;
  avg_grade NUMERIC;
BEGIN
  FOR student_id IN (SELECT id_student FROM lab.student)
  LOOP
    SELECT AVG(grade) INTO avg_grade
    FROM lab.attestation
    WHERE id is studying = (SELECT id is studying FROM lab.is studying WHERE id student =
student_id);
    UPDATE lab.student
    SET average grade = avg grade
    WHERE id student = student id;
  END LOOP;
END;
$$ LANGUAGE plpgsql;
SELECT calculate_average_grade();
```

## Видим актуальные данные:

	id_student [PK] integer	full_name character varying	average_grade numeric	scholarship numeric
1	1	Иван Иванов	4.25	[null]
2	2	Мария Смирнова	2.5	[null]
3	3	Георгий Константинов	3	[null]
4	4	Дмитрий Рубежкин	5	[null]

#### Затем приступаем к функции увеличения стипендии:

```
CREATE OR REPLACE PROCEDURE increase_scholarship()

AS $$

DECLARE

avg_grade numeric;

BEGIN

IF NOT EXISTS (SELECT 1 FROM information_schema.columns

WHERE table_name = 'student' AND column_name = 'average_grade')

THEN

ALTER TABLE lab.student ADD COLUMN average_grade numeric;

END IF;

IF NOT EXISTS (SELECT 1 FROM information_schema.columns
```

```
WHERE table_name = 'student' AND column_name = 'scholarship')

THEN

ALTER TABLE lab.student ADD COLUMN scholarship numeric;
END IF;

SELECT

UPDATE lab.student

SET scholarship = type_scholarship.amount_sch * 1.1

FROM lab.type_scholarship, lab.receiving_sch

WHERE lab.receiving_sch.id_sch = lab.type_scholarship.id_sch

AND lab.student.average_grade >= 4.5;

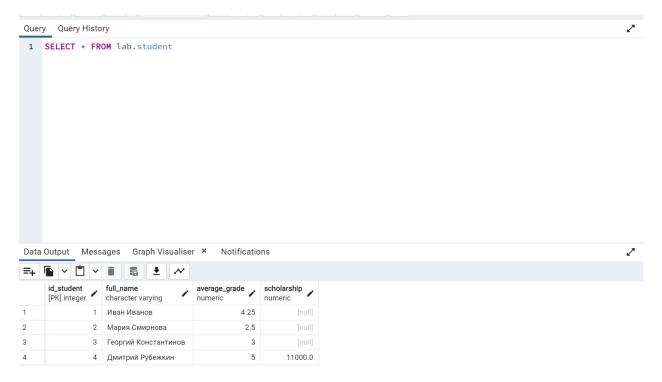
SELECT AVG(grade) INTO avg_grade

FROM lab.attestation;

END;

$$ LANGUAGE plpgsql;

CALL increase_scholarship();
```



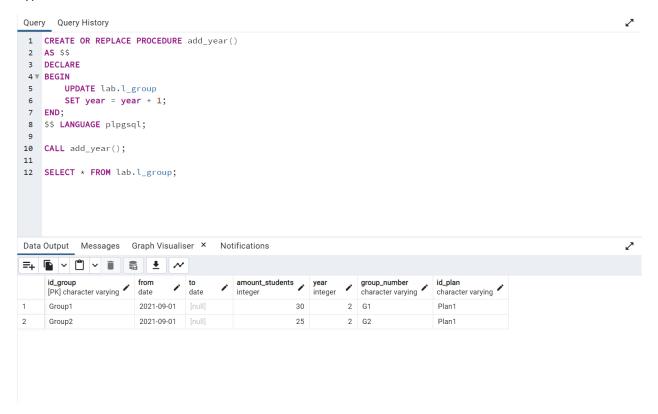
Новая стипендия для отличника Дмитрия Рубежкина – 11000 рублей.

### 2. Для перевода студентов на следующий курс.

```
CREATE OR REPLACE PROCEDURE add_year()
AS $$
DECLARE
BEGIN
UPDATE lab.l_group
SET year = year + 1;
END;
$$ LANGUAGE plpgsql;

CALL add_year();
```

Смотрим таблицу и видим, что первокурсники действительно повзрослели, и вся группа теперь 2 курс.



#### 3. Для изменения оценки при успешной пересдаче экзамена.

```
CREATE OR REPLACE PROCEDURE update_grade(student_id VARCHAR, id_subject VARCHAR, new_grade INTEGER)

AS $$
BEGIN

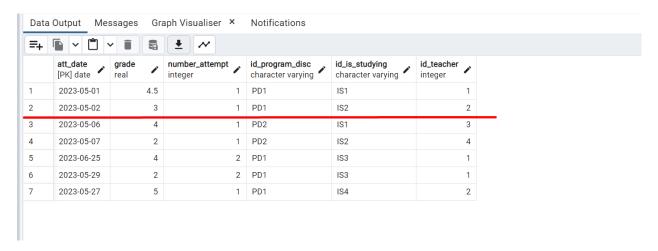
UPDATE lab.attestation

SET grade = new_grade,

number_attempt = number_attempt + 1

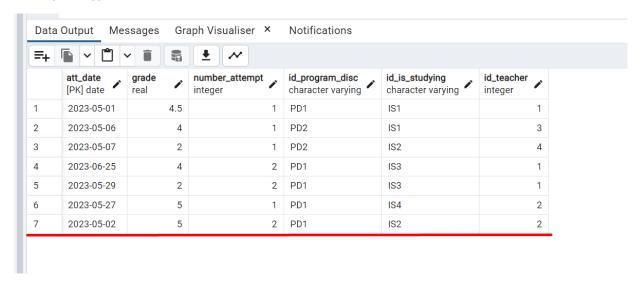
WHERE id_is_studying = student_id AND id_program_disc = id_subject;
END;

$$ LANGUAGE plpgsql;
```



Наблюдаем за троечником IS2.

#### После процедуры:



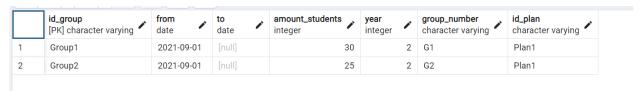
Мы UPDATEнули номер попытки и оценку учащегося.

# Задание 5. Триггеры

Задаем следующий триггер:

## Триггер для автоматического перевода студентов на следующий курс.

Скриншот до срабатывания триггера.



```
CREATE OR REPLACE FUNCTION update_year_trigger()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.year <= 4 THEN

NEW.year = NEW.year + 1;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER update_year_trigger

AFTER INSERT OR UPDATE ON lab.l_group

FOR EACH ROW

EXECUTE FUNCTION update_year_trigger();
```

Делаем UPDATE нашей таблицы:

UPDATE lab.l\_group

SET year = year;

Получаем следующее:

	id_group [PK] character varying	from date	to date	amount_students integer	year integer	group_number character varying	id_plan character varying
1	Group1	2021-09-01	[null]	30	3	G1	Plan1
2	Group2	2021-09-01	[null]	25	3	G2	Plan1

# Вывод:

В ходе лабораторной работы №3 мы изучили работу с процедурами и функциями, а также задали триггер для нашей базы данных, улучшающий работу с ней.