

Министерство образования и науки Республики Башкортостан  
Государственное автономное профессиональное образовательное учреждение  
Уфимский колледж статистики, информатики и вычислительной техники

## Отчет

Выполнил Студент гр. 22П-1  
Баранов Артём

## Оглавление

Представления MySql.....	3
Представления PostgreSQL.....	7
Представления по вариантам.....	11
Хранимые процедуры MySql.....	15
Хранимые процедуры PostgreSQL.....	21
ТРИГГЕРЫ My SQL.....	29
ТРИГГЕРЫ POSTGRE.....	34
Пользователи.....	40

## Представления MySQL

1 Создайте представление, которое показывает код поставки, наименование книги, дату поставки, наименование поставщика, стоимость поставки, объем поставки.

```
CREATE
ALGORITHM = UNDEFINED
DEFINER = `root`@`localhost`
SQL SECURITY DEFINER
VIEW `report` AS
SELECT
    `deliveries`.`Code_delivery` AS `Code_delivery`,
    `books`.`Title_book` AS `Title_book`,
    `purchases`.`Date_order` AS `Date_order`,
    `deliveries`.`Name_company` AS `Name_company`,
    (`purchases`.`Cost` * `purchases`.`Amount`) AS `totalCost`,
    `purchases`.`Amount` AS `Amount`
FROM
    ((`purchases`
    JOIN `books` ON ((`books`.`Code_book` = `purchases`.`Code_book`)))
    JOIN `deliveries` ON ((`deliveries`.`Code_delivery` = `purchases`.`Code_delivery`)))
```

С помощью select выбираем нужные нам поля из таблицы с поставками, книгами и соединяем с помощью join с таблицей книги по ключевому полю коду книги и с таблицей доставок по коду доставки

	Code_delivery	Title_book	Date_order	Name_company	totalCost	Amount
►	1	Евгений Онегин	2022-01-15	ОАО Книготорг	5000.00	10
	1	Война и мир	2022-06-10	ОАО Книготорг	7500.00	5
	2	Преступление и наказание	2023-03-01	ООО Библиосфера	7560.00	7
	2	Детство	2023-11-20	ООО Библиосфера	7200.00	15
	2	Мемуары	2023-12-12	ООО Библиосфера	150000.00	200
	3	Красная таблетка	2022-06-06	ООО Питерский	160.00	8

2 Создайте представление, которое показывает все сведения об издательствах из города Москва. (город был изменен на Санкт-Петербург в силу того что при заполнении бд Москвы в тот момент не было)

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = 'root'@'localhost'
  SQL SECURITY DEFINER
VIEW `about_publisher` AS
  SELECT
    `publishing_house`.`Code_publish` AS `Code_publish`,
    `publishing_house`.`Publish` AS `Publish`,
    `publishing_house`.`City` AS `City`,
    `books`.`Code_book` AS `Code_book`,
    `books`.`Title_book` AS `Title_book`
  FROM
    (`publishing_house`
    JOIN `books` ON ((`books`.`Code_publish` = `publishing_house`.`Code_publish`)))
  WHERE
    (`publishing_house`.`City` = 'Санкт-Петербург')
```

С помощью select выбираем поля код издательства, название, его город, код книги и ее название потом соединяем с книгами по ключевому полю.

	Code_publish	Publish	City	Code_book	Title_book
▶	2	Эксмо	Санкт-Петербург	2	Война и мир
	2	Эксмо	Санкт-Петербург	5	Мемуары
	2	Эксмо	Санкт-Петербург	10	Наука. Техника. Иновации
	4	Питер-Софт	Санкт-Петербург	6	Мальчик со шпагой

3 Создайте представление, которое показывает код книги, наименование книги, автора, количество книг на складе, стоимость книг (максимальная стоимость).

```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = 'root'@'localhost'
  SQL SECURITY DEFINER
VIEW `sklad_info` AS
  SELECT
```

```

`books`.`Code_book` AS `code_book`,
`books`.`Title_book` AS `title_book`,
`authors`.`name_author` AS `name_author`,
`purchases`.`Amount` AS `amount`,
`purchases`.`Cost` AS `cost`
FROM
((`books`
JOIN `authors` ON ((`authors`.`Code_author` = `books`.`Code_book`)))
JOIN `purchases` ON ((`purchases`.`Code_book` = `books`.`Code_book`)))

```

Выбираем код книги, название, автора из таблицы книги и из покупок количество и стоимость далее соединяем с авторами чтобы к каждой книге приписать ее автора и с покупками чтобы вытащить стоимость и количество

	code_book	title_book	name_author	amount	cost
▶	1	Евгений Онегин	Александр Пушкин	10	500.00
	2	Война и мир	Лев Толстой	5	1500.00
	3	Преступление и наказание	Федор Достоевский	7	1080.00
	4	Детство	Лев Толстой	15	480.00
	5	Мемуары	Лев Толстой	200	750.00
	16	Красная таблетка	Андрей курпатов	8	20.00

4 Создайте представление, которое показывает топ 5 книг с максимальным количеством на складе (используйте предыдущее представление).

```

CREATE
ALGORITHM = UNDEFINED
DEFINER = `root`@`localhost`
SQL SECURITY DEFINER
VIEW `top5` AS
SELECT
`sklad_info`.`code_book` AS `code_book`,
`sklad_info`.`title_book` AS `title_book`,
`sklad_info`.`name_author` AS `name_author`,
`sklad_info`.`amount` AS `amount`,
`sklad_info`.`cost` AS `cost`
FROM
`sklad_info`
ORDER BY `sklad_info`.`amount` DESC

```

LIMIT 5

С помощью select выбираем нужные поля сортируем с помощью Order by desc по убыванию, для выбора 5 записей устанавливаем Limit 5

	code_book	title_book	name_author	amount	cost
►	5	Мемуары	Лев Толстой	200	750.00
	4	Детство	Лев Толстой	15	480.00
	1	Евгений Онегин	Александр Пушкин	10	500.00
	16	Красная таблетка	Андрей курпатов	8	20.00
	3	Преступление и наказание	Федор Достоевский	7	1080.00

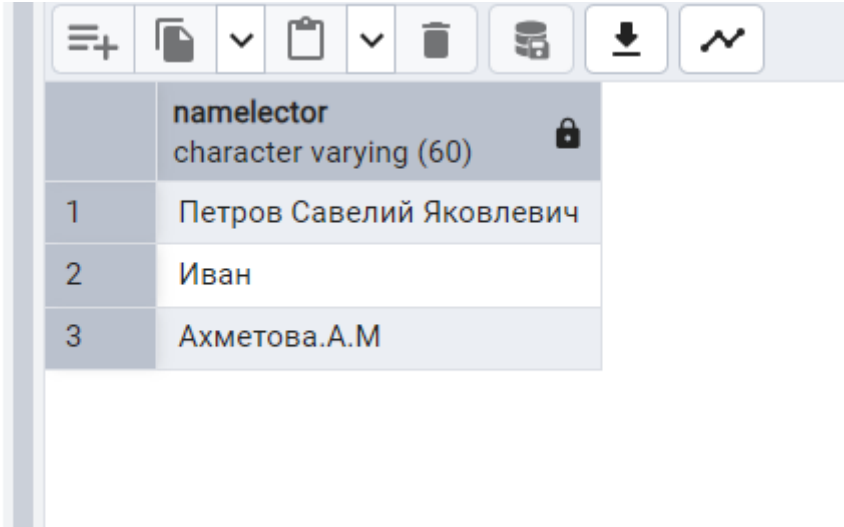
## Представления PostgreSQL

1 Создайте представление, которое показывает преподавателей, не принимающих экзамены.

```
CREATE OR REPLACE VIEW public.free_teachers  
AS  
SELECT lectors.namelector  
FROM lectors  
LEFT JOIN progress ON lectors.codelector = progress.codelector  
WHERE progress.codelector IS NULL;
```

```
ALTER TABLE public.free_teachers  
OWNER TO postgres;
```

Выбираем из таблицы лекторов имя лектора соединяем с таблицей прогресс где записи об экзаменах и делаем выборку где у лектора нету записей о экзамене



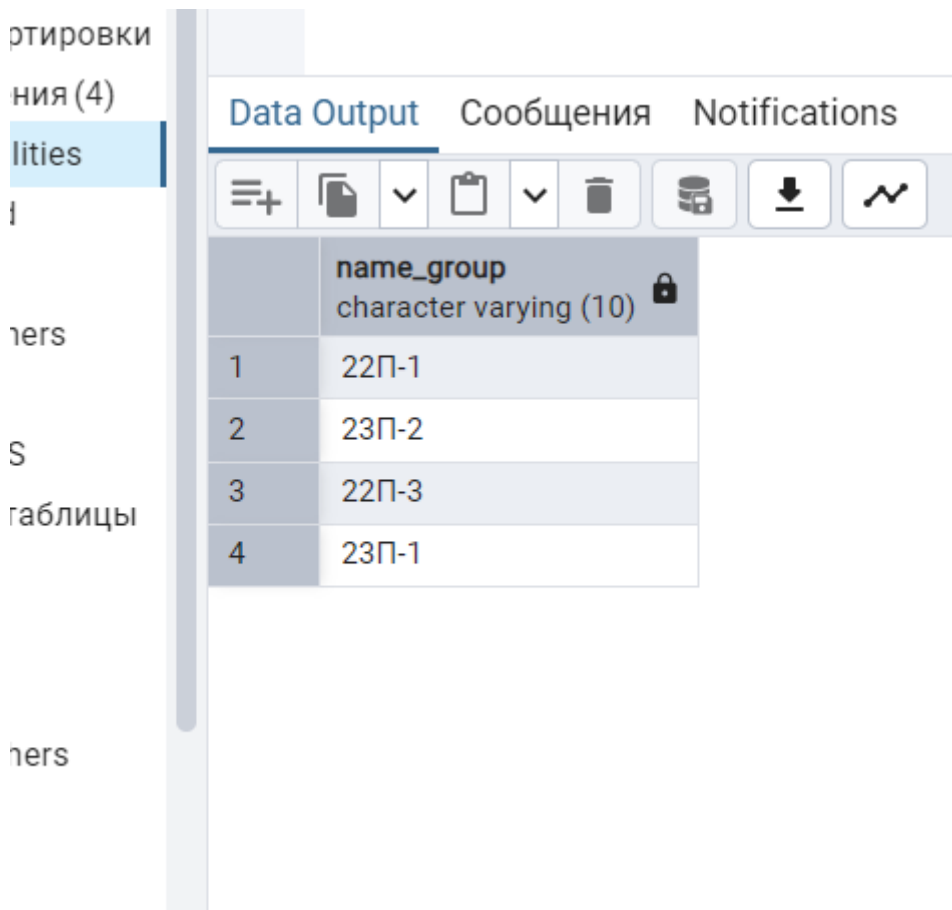
	namelector character varying (60)
1	Петров Савелий Яковлевич
2	Иван
3	Ахметова.А.М

2 Создайте представление, которое показывает список групп специальности Программирование в компьютерных системах.

```
CREATE OR REPLACE VIEW public.all_specialities
AS
SELECT name_group
FROM groups
WHERE name_specialty::text = 'Программист'::text;
```

```
ALTER TABLE public.all_specialities
OWNER TO postgres;
```

Выбираем название группы где название специальности = Программист



	name_group character varying (10)
1	22П-1
2	23П-2
3	22П-3
4	23П-1

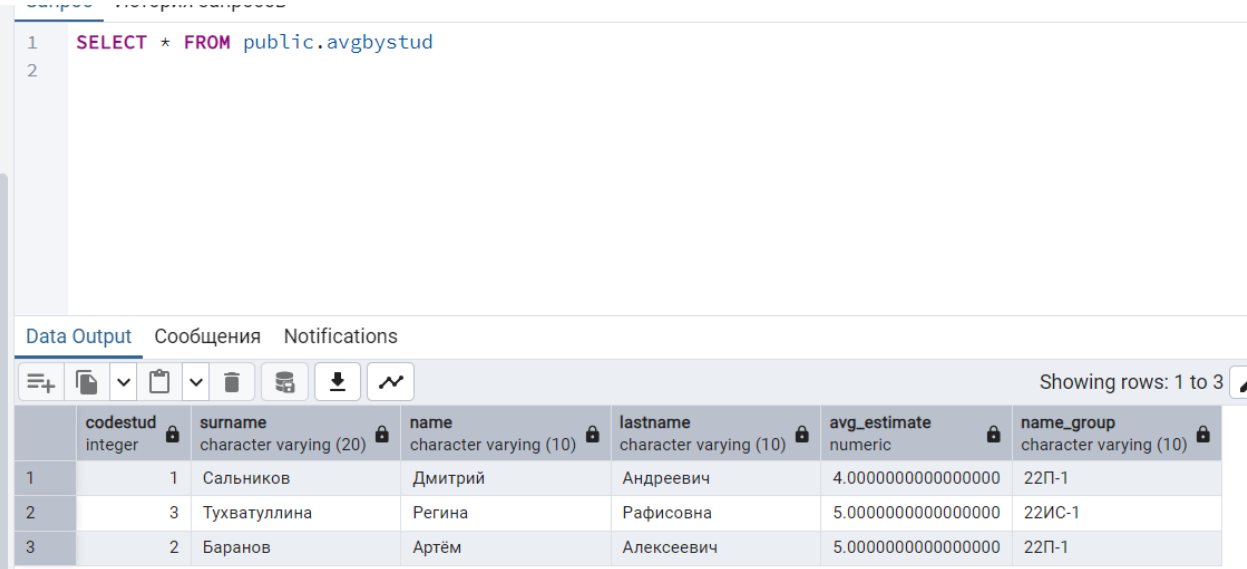


3 Создайте представление, которое показывает код студента, ФИО студента, наименование группы, средний балл за весь период обучения.

```
CREATE OR REPLACE VIEW public.avgbystud
AS
SELECT students.codestud,
       students.surname,
       students.name,
       students.lastname,
       avg(progress.estimate) AS avg_estimate,
       groups.name_group
FROM students
      JOIN groups ON groups.code_group = students.code_group
      JOIN progress ON progress.code_stud = students.codestud
GROUP BY students.codestud, students.surname, students.name, students.lastname,
groups.name_group;
```

```
ALTER TABLE public.avgbystud
OWNER TO postgres;
```

Выбираем ФИО код, группу и высчитываем средний балл из всех оценок



The screenshot shows a database client interface. At the top, a SQL query is entered in a text area: `SELECT * FROM public.avgbystud`. Below the query, there is a tabbed interface with 'Data Output' selected. The 'Data Output' tab displays a table with 7 columns: `codestud` (integer), `surname` (character varying (20)), `name` (character varying (10)), `lastname` (character varying (10)), `avg_estimate` (numeric), and `name_group` (character varying (10)). The table contains 3 rows of data. The interface also includes a toolbar with icons for various actions and a status bar indicating 'Showing rows: 1 to 3'.

	<code>codestud</code> integer	<code>surname</code> character varying (20)	<code>name</code> character varying (10)	<code>lastname</code> character varying (10)	<code>avg_estimate</code> numeric	<code>name_group</code> character varying (10)
1	1	Сальников	Дмитрий	Андреевич	4.0000000000000000	22П-1
2	3	Тухватуллина	Регина	Рафисовна	5.0000000000000000	22ИС-1
3	2	Баранов	Артём	Алексеевич	5.0000000000000000	22П-1

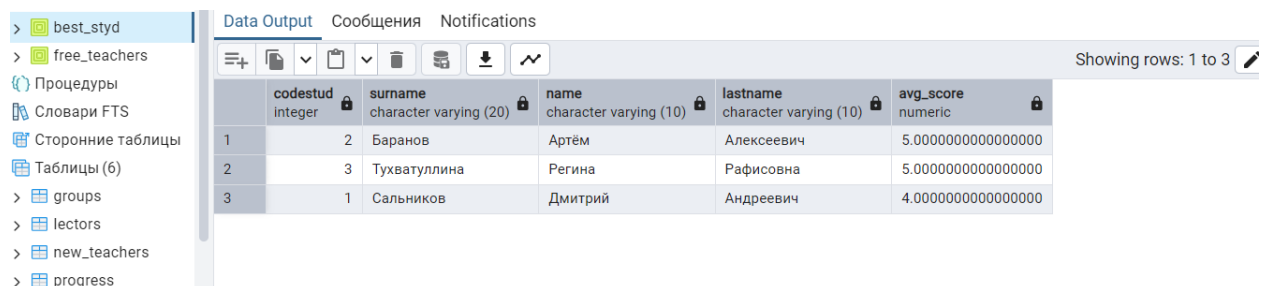
4 Создайте представление, которое показывает список из 5 студентов с наибольшим средним баллом за весь период обучения.

```
CREATE OR REPLACE VIEW public.best_styd
AS
SELECT students.codestud,
       students.surname,
       students.name,
       students.lastname,
       avg(progress.estimate) AS avg_score
FROM students
      JOIN progress ON students.codestud = progress.code_stud
GROUP BY students.codestud, students.surname, students.name, students.lastname
ORDER BY (avg(progress.estimate)) DESC
LIMIT 5
```

```
ALTER TABLE public.best_styd
```

```
OWNER TO postgres;
```

Выбираем код студента, фиио, так же вычисляем среднюю оценку из всех оценок студента с помощью avg и группируем по коду и фиио чтоб считались все записи оценок этого студента



	codestud integer	surname character varying (20)	name character varying (10)	lastname character varying (10)	avg_score numeric
1	2	Баранов	Артём	Алексеевич	5.0000000000000000
2	3	Тухватуллина	Регина	Рафисовна	5.0000000000000000
3	1	Сальников	Дмитрий	Андреевич	4.0000000000000000

## Представления по вариантам

1. Представление "Ведомость на получение зарплаты" (оклад-13%).

```
create view sallaty_pay as
```

```
select full_name, positions.salary * 0.87 as "salary" from employees
```

```
join positions on positions.position_name = employees.current_position
```

Выбираем фио и зарплату умножаем на 0,87 чтобы отнять 13%

	full_name character varying (40)	salary numeric
1	Сидорова Анна Викторовна	78300.0000
2	Кузнецов Сергей Алексеевич	56550.0000
3	Морозова Елена Сергеевна	43500.0000
4	Андреева Мария Владимировна	69600.0000
5	Григорьев Алексей Павлович	65250.0000
6	Федоров Артем Дмитриевич	73950.0000
7	Егорова Наталья Петровна	41760.0000
8	Васильев Дмитрий Олегович	47850.0000
9	Иванов Иван Иванович	60900.0000
10	Петров Петр Петрович	60900.0000

2. Представление "Трудовая книжка": ФИО сотрудника – должность – дата вступления в должность – дата перехода на другую должность (или "по наст. время").

```
CREATE VIEW employment_book AS
```

```
SELECT
```

```
    e.full_name AS employee_name,
```

```
    er.position AS position,
```

```
    er.joining_date AS start_date,
```

```
    COALESCE
```

```
        (
```

```
            TO_CHAR
```

```
            (
```

```
                LEAD(er.joining_date) OVER (PARTITION BY er.employee ORDER BY
```

```
er.joining_date),
```

```
                'YYYY-MM-DD'
```

```
            )
```

```
        , 'по наст. время'
```

```
    )
```

```
    AS end_date
```

```
FROM employment_record er
```

```
JOIN employees e ON er.employee = e.employee_id;
```

Вытаскиваем фиио, должность и дату заступления на должность с помощью Lead возвращает дату о вступлении в должность. Partition разбивает данные по каждому сотруднику, с помощью To char преобразуем дату в строку а если lead вернул null то пишем «по настоящее время»

	employee_name character varying (40) 🔒	position character varying (30) 🔒	start_date date 🔒	end_date text 🔒
1	Иванов Иван Иванович	Менеджер	2020-03-10	2021-03-12
2	Иванов Иван Иванович	Бухгалтер	2021-03-12	2022-04-15
3	Иванов Иван Иванович	Программист	2022-04-15	2023-05-18
4	Иванов Иван Иванович	Маркетолог	2023-05-18	по наст. время
5	Петров Петр Петрович	Бухгалтер	2019-05-22	по наст. время
6	Сидорова Анна Викторовна	Программист	2021-07-01	2022-07-12
7	Сидорова Анна Викторовна	Бухгалтер	2022-07-12	по наст. время
8	Кузнецов Сергей Алексеевич	Маркетолог	2018-04-15	по наст. время
9	Морозова Елена Сергеевна	Продавец	2022-01-10	по наст. время
10	Васильев Дмитрий Олегович	Логист	2020-09-05	по наст. время
11	Андреева Мария Владимировна	Инженер	2017-06-20	по наст. время
12	Григорьев Алексей Павлович	Аналитик	2019-11-01	по наст. время
13	Федоров Артем Дмитриевич	Юрист	2016-08-10	по наст. время
14	Егорова Наталья Петровна	Специалист поддержки	2023-02-15	по наст. время

3. Представление "Штатное расписание на 1 января текущего года": выдать для каждого сотрудника его должность на эту дату.

```
CREATE VIEW Staffing_Schedule AS
```

```
SELECT Full_Name, current_position
```

```
FROM employees
```

```
WHERE date_of_joining <= DATE '2025-01-01'
```

Выбираем сотрудника и его должность, смотрим на дату заступления и если она меньше или равна выше указанной дате выводим

	full_name character varying (40)	current_position character varying (30)
1	Сидорова Анна Викторовна	Программист
2	Кузнецов Сергей Алексеевич	Маркетолог
3	Морозова Елена Сергеевна	Продавец
4	Андреева Мария Владимировна	Инженер
5	Григорьев Алексей Павлович	Аналитик
6	Федоров Артем Дмитриевич	Юрист
7	Егорова Наталья Петровна	Специалист поддержки
8	Васильев Дмитрий Олегович	Логист
9	Петров Петр Петрович	Менеджер

## Хранимые процедуры MySQL

1. Вывести все сведения о поставке (все поля таблицы Purchases), а также название книги (поле Title\_book) с максимальной общей стоимостью (использовать поля Cost и Amount).

```
CREATE DEFINER='root'@'localhost' PROCEDURE `delivery_info`()
```

```
BEGIN
```

```
SELECT
```

```
    books.code_book,
```

```
    books.title_book,
```

```
    SUM(purchases.amount) AS totalAmount,
```

```
    MAX(purchases.cost * purchases.amount) AS maxTotalCost
```

```
FROM purchases
```

```
JOIN books ON books.code_book = purchases.code_book
```

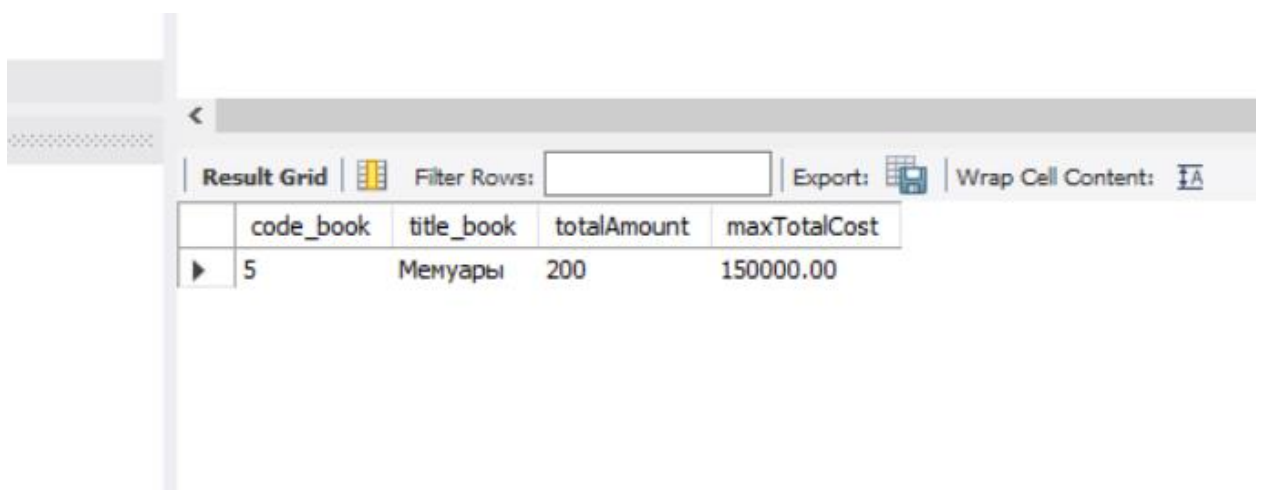
```
GROUP BY books.code_book, books.title_book
```

```
ORDER BY maxTotalCost DESC
```

```
LIMIT 1;
```

```
END
```

Выбираем нужные нам поля с помощью sum считаем все количество книги и с помощью max находим наибольшее число после умножения totalamount и cost, группируем по названию и коду и сортируем по убыванию с лимитом строк 1



	code_book	title_book	totalAmount	maxTotalCost
►	5	Мемуары	200	150000.00

2. Сосчитать количество книг определенного автора (ФИО автора является входным параметром).

```
CREATE DEFINER='root'@'localhost' PROCEDURE `autor_amount_of_books`(name  
varchar(45))
```

```
BEGIN
```

```
SELECT COUNT(*) AS book_count
```

```
FROM books
```

```
JOIN authors ON books.code_author = authors.code_author
```

```
WHERE authors.name_author = name;
```

```
END
```

Соединяем книги с авторами и ставим условие что имя автора = тому что передали в аргументы процедуры

```
1 • call db_books.autor_amount_of_books('Александр Пушкин');  
2
```

<
Result Grid   Filter Rows:   Export:   Wrap Cell Content:
book_count
▶ 8



3. Определить адрес определенного поставщика (Наименование поставщика является входным параметром, адрес поставщика – выходным параметром).

```
CREATE DEFINER='root'@'localhost' PROCEDURE `find_deliverer`(name_deliverer  
varchar(45), out name varchar(45))
```

```
BEGIN
```

```
select adress into name from deliveries
```

```
where name_company = name_deliverer;
```

```
END
```

Выбираем адрес и ставим условие чтобы название компании было равно параметру в аргументах процедуры



```
1 • set @name = '0';  
2 • call db_books.find_deliverer('ОАО Книготорг', @name);  
3 • select @name;  
4
```



@name
Москва

4. Выполните операцию вставки в таблицу Books. Код книги должен увеличиваться автоматически на единицу.

```
CREATE DEFINER='root'@'localhost' PROCEDURE `insert_in_books`(title varchar(45),
pages int, code_author int, code_publish int, cost DECIMAL(10,2))
```

```
BEGIN
```

```
DECLARE max_code INT;
```

```
SELECT MAX(Code_book) + 1 INTO max_code FROM books;
```

```
insert into books (Code_book, title_book, pages, code_author, code_publish, cost)
```

```
values(max_code, title, pages, code_author, code_publish, cost);
```

```
END
```

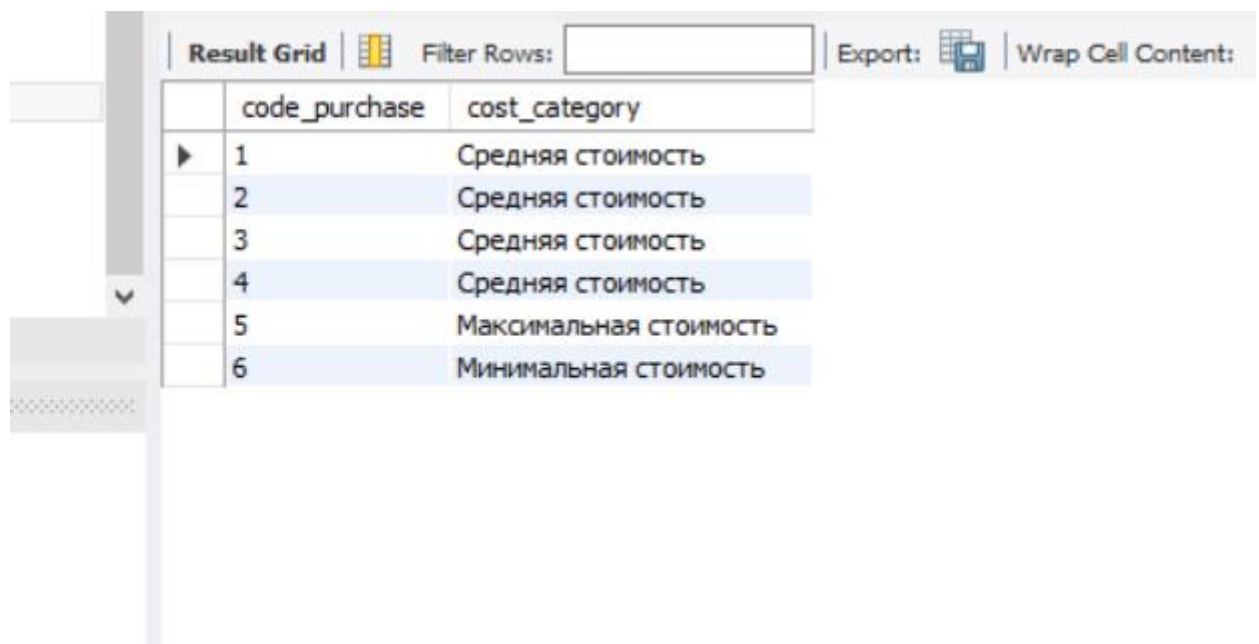
При выполнении процедуры с помощью max берем последний номер книги и увеличиваем на 1 после чего записываем в таблицу

1	Евгений Онегин	300	1	1	500.00
2	Война и мир	1225	2	2	1500.00
3	Преступление и наказание	670	3	3	900.00
4	Детство	250	2	1	400.00
5	Мемуары	300	2	2	800.00
6	Мальчик со шпагой	1000	3	4	100.00
7	Тень и кость	700	1	5	700.00
8	Паучья Банда	500	2	6	800.00
9	Труды	700	2	8	900.00
10	Наука. Техника. Иновации	300	1	2	302.00
11	Стихи	300	1	1	500.00
12	Стихи	300	1	1	500.00
13	Стихи	300	1	1	500.00
14	Стихи	300	1	1	500.00
15	Вселенная	300	1	1	1000.00
16	Красная таблетка	347	9	5	1616.40
17	Красная таблетка 2	347	9	5	1347.00
*	NULL	NULL	NULL	NULL	NULL

5. Определить поставки с минимальной и максимальной стоимостью книг. Отобразить список всех поставок. Если стоимость поставки – максимальная, то вывести сообщение «Максимальная стоимость», если стоимость – минимальная, то вывести сообщение «Минимальная стоимость», иначе – «Средняя стоимость».

```
CREATE DEFINER='root'@'localhost' PROCEDURE `sort_pur`()
BEGIN
select code_purchase,
CASE
    WHEN (amount*cost) = (select max((amount*cost) ) from purchases)
        THEN "Максимальная стоимость"
    WHEN (amount*cost) = (select min((amount*cost) ) from purchases)
        THEN "Минимальная стоимость"
    ELSE "Средняя стоимость"
END AS cost_category
from purchases;
END
```

Считаем стоимость всей поставки если она равно максимуму то приписываем ей максимальная стоимость, а если минимальная то минимальная все остальные соответственно средняя стоимость



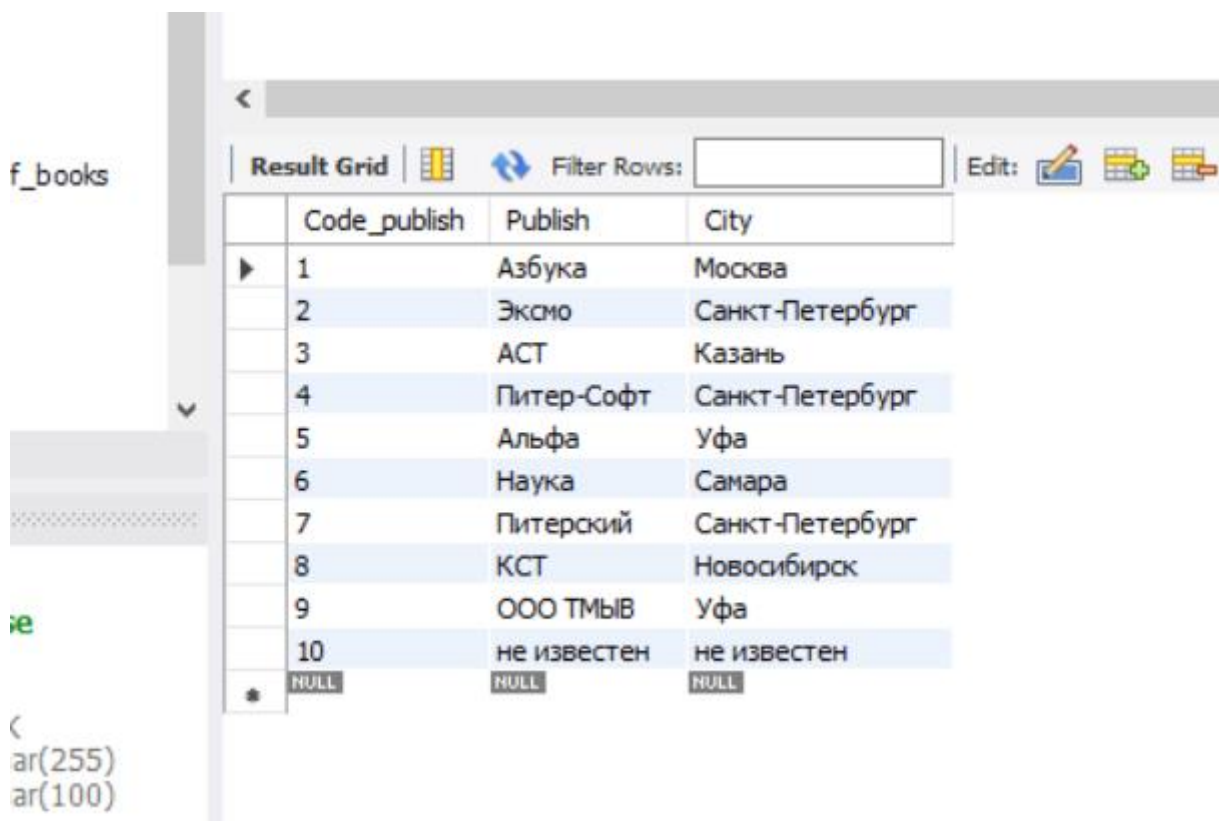
The screenshot shows a 'Result Grid' window with a table containing 6 rows. The columns are 'code\_purchase' and 'cost\_category'. The data is as follows:

code_purchase	cost_category
1	Средняя стоимость
2	Средняя стоимость
3	Средняя стоимость
4	Средняя стоимость
5	Максимальная стоимость
6	Минимальная стоимость

6. Определить количество записей в таблице поставщиков. Пока записей меньше 10, делать в цикле добавление записи в таблицу с автоматическим наращиванием значения ключевого поля, а вместо названия поставщика ставить значение 'не известен'.

```
CREATE DEFINER='root'@'localhost' PROCEDURE `new_data`()
BEGIN
DECLARE max_code INT;
while ((select max(Code_publish) from publishing_house) < 10) do
    SELECT MAX(Code_publish) + 1 INTO max_code FROM publishing_house;
    insert into publishing_house (Code_publish,Publish,City )
        values(max_code, "не известен", "не известен");
END while;
END
```

С помощью цикла добавляем новые записи и в условии проверяем номер записи и пока он не равняется 10 цикл работает



	Code_publish	Publish	City
▶	1	Азбука	Москва
	2	Эксмо	Санкт-Петербург
	3	АСТ	Казань
	4	Питер-Софт	Санкт-Петербург
	5	Альфа	Уфа
	6	Наука	Самара
	7	Питерский	Санкт-Петербург
	8	КСТ	Новосибирск
	9	ООО ТМЫВ	Уфа
	10	не известен	не известен
★	NULL	NULL	NULL

## Хранимые процедуры PostgreSQL

1. Вывести фамилии и имена студентов (поля Surname, Name из таблицы Students) с максимальным средним баллом за весь период обучения (условие по полю Estimate из таблицы Progress).

```
CREATE FUNCTION GetStudentsWithTheHighestAverageGrade()  
RETURNS TABLE(surname VARCHAR(255), name VARCHAR(255)) AS $$  
DECLARE HighestAverageGrade DECIMAL(10, 2);  
BEGIN  
    SELECT AVG(progress.estimate) INTO HighestAverageGrade FROM progress  
    GROUP BY progress.code_stud  
    ORDER BY AVG(progress.estimate) DESC  
    LIMIT 1;  
  
    RETURN QUERY SELECT students.surname, students.name FROM students  
    JOIN progress ON progress.code_stud = students.code_stud  
    GROUP BY students.code_stud  
    HAVING AVG(progress.estimate) = HighestAverageGrade;  
END;  
$$ LANGUAGE plpgsql;
```

Выбираем среднее от успеваемости группируем по коду студента.

Data Output			Сообщения	Notifications
	surname character varying	name character varying		
1	Сергеев	Антон		
2	Иванов	Петр		

2. Определить средний балл определенного студента (ФИО студента является входным параметром).

```
CREATE FUNCTION GetStudentAverageEstimate(IN student_name VARCHAR(255))
RETURNS DECIMAL(10, 2) AS $$
BEGIN
RETURN (
SELECT AVG(progress.estimate) FROM students
JOIN progress ON progress.code_stud = students.code_stud
WHERE surname ' ' name ' ' lastname = student_name
);
END;
$$ LANGUAGE plpgsql;
```

Выбираем среднее от успеваемости и ставим условие чтобы фио равнялось тому что указано в аргументах.

Data Output		Сообщения	Notifications
	getstudentaverageestimate numeric		
1	5.0000000000000000		

3. Определить специальность и номер курса определенного студента (ФИО студента является входным параметром, Название специальности и Номер курса – выходными параметрами).

```
CREATE OR REPLACE FUNCTION GetStudentsGropus(IN student_name VARCHAR(255))
RETURNS TABLE(speciality VARCHAR(255), num_course INT) AS $$
```

BEGIN

RETURN QUERY (

```
SELECT groups.name_group, groups.num_course FROM groups
```

JOIN students ON students.code\_group = groups.code\_group

```
WHERE surname ' ' name ' ' lastname = student_name
```

$$);$$

END;

```
$$ LANGUAGE plpgsql;
```

Находим группу студента по его фио которое передаем в аргументы функции.

Data Output

Сообщения

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

SQL

	speciality character varying	num_cource integer
1	AC-101	1

4. Выполните операцию вставки в таблицу Students. Код студента должен автоматически увеличиваться на единицу.

```
CREATE OR REPLACE FUNCTION AddStudent(IN surname VARCHAR(255), IN name  
VARCHAR(255), IN lastname VARCHAR(255), IN birthday DATE)  
RETURNS VOID AS $$  
BEGIN  
    INSERT INTO students(code_stud, surname, name, lastname, birthday)  
    VALUES((SELECT MAX(st.code_stud) + 1 FROM students AS st), surname, name, lastname,  
birthday);  
END;  
$$ LANGUAGE plpgsql;
```

Из таблицы с учениками берем максимальный код записи и при добавлении добавляем + 1



5. Определить средний возраст всех студентов. Вывести список всех студентов. Если возраст студента больше среднего возраста, то вывести сообщение «Вы старше среднего возраста всех студентов», если возраст – меньше, то вывести сообщение «Ваш возраст меньше среднего возраста всех студентов», а иначе – «Ваш возраст равен среднему возрасту всех студентов».

```
CREATE OR REPLACE FUNCTION GetStudentsAge()
RETURNS TABLE(surname VARCHAR(255), name VARCHAR(255), age NUMERIC, message TEXT)
AS $$
DECLARE
    average_age DECIMAL(10, 2);
BEGIN
    SELECT AVG(EXTRACT(YEAR FROM AGE(birthday))) INTO average_age FROM students;

    RETURN QUERY
    SELECT
        students.surname,
        students.name,
        EXTRACT(YEAR FROM AGE(birthday)) as "Age",
        CASE
            WHEN EXTRACT(YEAR FROM AGE(birthday)) < average_age THEN 'Ваш возраст меньше
среднего'
            WHEN EXTRACT(YEAR FROM AGE(birthday)) = average_age THEN 'Ваш возраст равен
среднему'
            WHEN EXTRACT(YEAR FROM AGE(birthday)) > average_age THEN 'Вы старше среднего
возраста'
        END CASE
    FROM students
;
END;
$$ LANGUAGE plpgsql;
```

Выбираем возраст вычисляем среднее и через множественный выбор предписываем значения.

Data Output   Сообщения   Notifications				
<div> <div>☰+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> </div>				
	surname character varying 	name character varying 	age numeric 	message text 
1	Смирнова	Ольга	24	Ваш возраст меньше среднего
2	Новиков	Алексей	22	Ваш возраст меньше среднего
3	Воронов	Евгений	48	Вы старше среднего возраста
4	Лебедев	Николай	22	Ваш возраст меньше среднего
5	Кириллова	Мария	21	Ваш возраст меньше среднего
6	Иванов	Петр	25	Вы старше среднего возраста
7	Михайлова	Елена	23	Ваш возраст меньше среднего
8	Сидорова	Анна	23	Ваш возраст меньше среднего
9	Кузнецов	Дмитрий	25	Вы старше среднего возраста
10	Сергеев	Антон	23	Ваш возраст меньше среднего
11	Петров	Иван	23	Ваш возраст меньше среднего
12	-	-	[null]	[null]
13	Иванов	Павел	20	Ваш возраст меньше среднего
14	Иванов	Петр	25	Вы старше среднего возраста

6. Определить количество записей в таблице дисциплин. Пока записей меньше 10, делать в цикле добавление записи в таблицу с автоматическим наращиванием значения ключевого поля, а вместо названия дисциплины ставить значение 'не известно'.

```
CREATE OR REPLACE FUNCTION FillSubjects()
RETURNS VOID AS $$
DECLARE
    record_count INT;
BEGIN
    SELECT COUNT(*) INTO record_count FROM subjects;

    WHILE record_count < 10 LOOP
        INSERT INTO subjects(code_subject, name_subject, count_hours)
        VALUES(
            (SELECT MAX(s.code_subject) + 1 FROM subjects AS s),
            'не известно', 0);

        SELECT COUNT(*) INTO record_count FROM subjects;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

Используем цикл с условием и пока количество записей меньше 10 добавляем новые строки

Запрос История запросов

Data Output Сообщения Notifications

	<b>code_subject</b> [PK] integer	<b>name_subject</b> character varying (255)	<b>count_hours</b> integer
1	1	Математический анализ	120
2	2	Математический анализ	100
3	3	Физика	90
4	4	Программирование	110
5	5	Введение в информатику	80
6	6	История математики	60
7	7	Механика	70
8	8	Алгебра	90
9	9	не известно	0
10	10	не известно	0

## ТРИГГЕРЫ My SQL

1. Создайте триггер, запускаемый при занесении новой строки в таблицу Авторы. Триггер должен увеличивать счетчик числа добавленных строк.

```
delimiter $$
```

```
create definer = current_user trigger `db_books`.`count_new_authors` after insert on authors for  
each row
```

```
begin
```

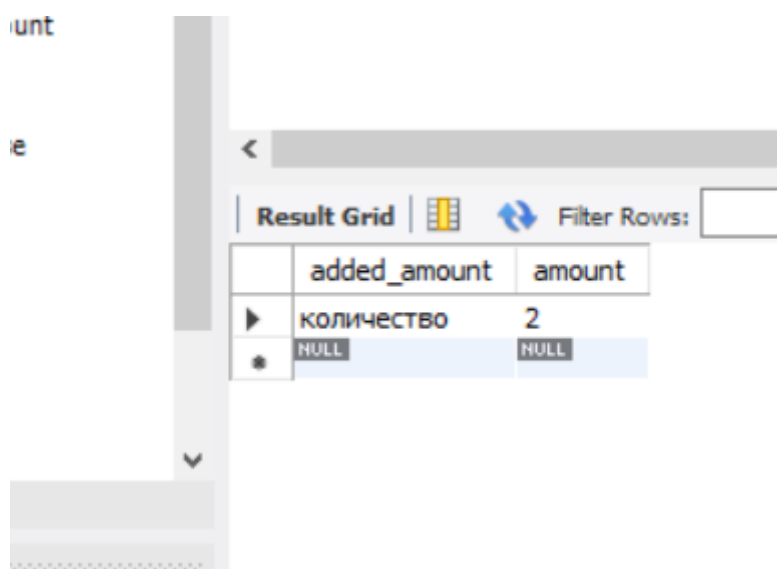
```
update new_authors_count set amount = amount + 1;
```

```
end;
```

```
$$
```

```
delimiter $$
```

Триггер запускается после вставки в таблицу авторов и увеличивает счетчик на 1



	added_amount	amount
▶	количество	2
★	NULL	NULL

2. Добавьте в таблицу Авторы поле Количество книг (Count\_books) целого типа со значением по умолчанию 0. Создайте хранимую процедуру, которая подсчитывает количество книг по каждому автору и заносит в поле Count\_books эту информацию. Создайте триггер, запускаемый после внесения новой информации о книге.

```
delimiter $$
```

```
create definer = current_user trigger `db_books`.`new_book_trigger` after insert on books for each row
```

```
begin
```

```
update authors a
```

```
set a.book_amount = (
```

```
select COUNT(*)
```

```
from books b
```

```
where b.code_author = a.code_author
```

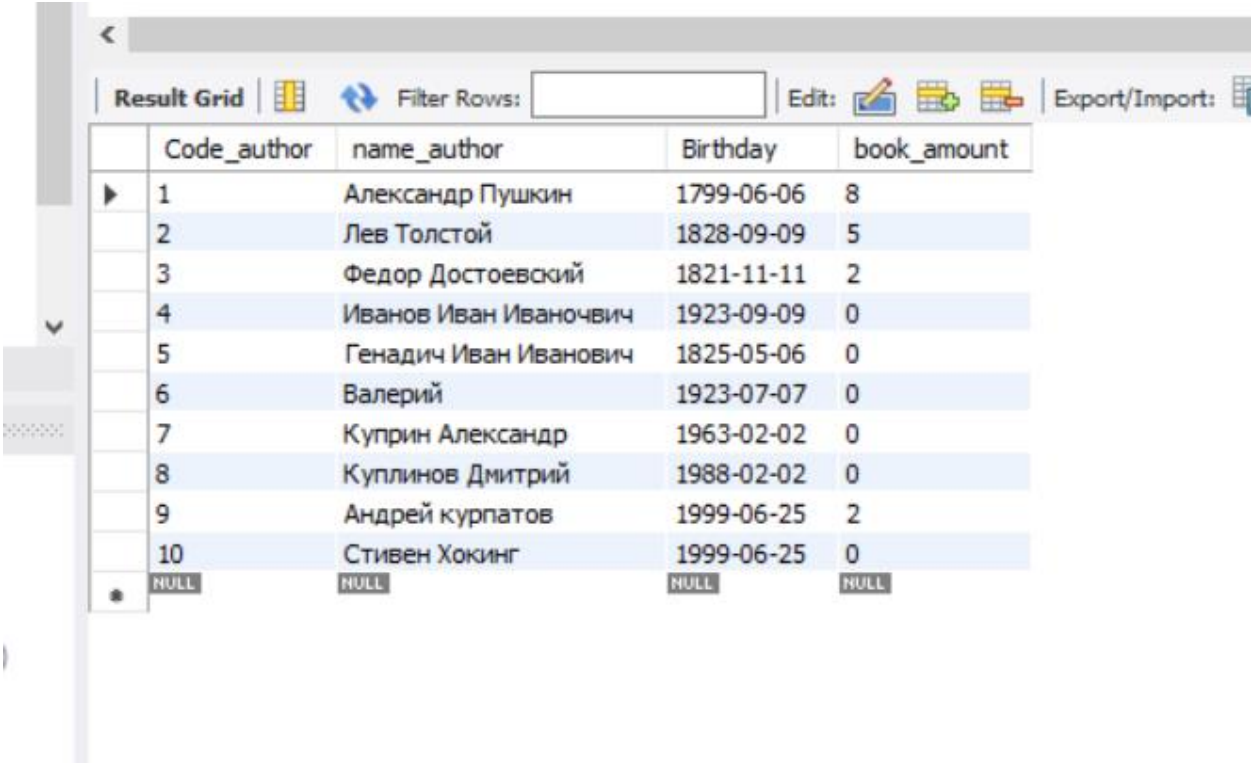
```
);
```

```
end;
```

```
$$
```

```
delimiter $$
```

при добавлении новой книги через код автора мы обновляем количество книг



The screenshot shows a database application interface with a table titled 'Result Grid'. The table has four columns: 'Code\_author', 'name\_author', 'Birthday', and 'book\_amount'. It contains 10 rows of data, each representing an author and their book count. The authors listed are Александр Пушкин, Лев Толстой, Федор Достоевский, Иванов Иван Иванович, Генадич Иван Иванович, Валерий, Куприн Александр, Куплинов Дмитрий, Андрей курпатов, and Стивен Хокинг. The 'book\_amount' values are 8, 5, 2, 0, 0, 0, 0, 0, 2, and 0 respectively. A final row shows NULL values for all columns.

Code_author	name_author	Birthday	book_amount
1	Александр Пушкин	1799-06-06	8
2	Лев Толстой	1828-09-09	5
3	Федор Достоевский	1821-11-11	2
4	Иванов Иван Иванович	1923-09-09	0
5	Генадич Иван Иванович	1825-05-06	0
6	Валерий	1923-07-07	0
7	Куприн Александр	1963-02-02	0
8	Куплинов Дмитрий	1988-02-02	0
9	Андрей курпатов	1999-06-25	2
10	Стивен Хокинг	1999-06-25	0
NULL	NULL	NULL	NULL

3. Создайте триггер, запускаемый при внесении информации о новых поставках. Выполните проверку о количестве добавляемой книги в таблице Книги. Если количество экземпляров книг в таблице меньше 10, то необходимо увеличить стоимость книг на 20 %.

```
delimiter $$
```

```
create definer = current_user trigger `db_books`.`new_delivery` after insert on purchases for each row  
begin
```

```
if (new.Amount ) < 10
```

```
then update books
```

```
set Cost = Cost * 1.2
```

```
where Code_book = new.Code_book;
```

```
end if;
```

```
end;
```

```
$$
```

```
delimiter $$
```

при добавлении мы смотрим на количество и если их меньше 10 мы добавляем 20% к стоимости

15	Вселенная	300	1	1	1000.00
16	Красная таблетка	347	9	5	1616.40
17	Красная таблетка 2	347	9	5	1347.00
•	NULL	NULL	NULL	NULL	NULL

4. Запретить вставлять новые строки в таблицу Поставщики, выводя при этом сообщение «Вставка строк запрещена».

```
delimiter $$
```

```
create definer = current_user trigger `db_books`.`no_update` before insert on deliveries for each  
row
```

```
begin
```

```
    signal sqlstate '45000'
```

```
    set message_text = 'Вставка строк запрещена';
```

```
end;
```

```
$$
```

```
delimiter $$
```

при попытке добавления строки в таблицу с помощью вызова ошибки мы выводим сообщение о запрете вставки

38	17:31:09	SELECT * FROM db_books.books LIMIT 0, 500	17 row(s) returned	0.000 sec / 0.000 sec
39	17:32:26	insert into deliveries (Code_delivery, Name_company, INN, Phone, adress) select max(Code_delivery+1), "q...	Error Code: 1644. Вставка строк запрещена	0.000 sec



5. Проверьте выполнение команд транзакции при добавлении новой информации об издательствах.

delimiter \$\$

```
create definer = current_user procedure `db_books`.`new_pub_house` (name varchar(45))
begin
    declare exit handler for sqlexception
    begin
        rollback;
    end;

    if name = "" or name is null then
        signal sqlstate "45000"
        set message_text = "Пустое название";
    end if;

    start transaction;

    insert into publishing_house (Code_publish, Publish, City)
    select max(Code_publish) + 1 , name, "Рим" from publishing_house;

    commit;
end$$
delimiter ;
```

при вызове происходит проверка и если название пустое то транзакция откатывается

## ТРИГГЕРЫ POSTGRE

1. Создайте триггер, запускаемый при занесении новой строки в таблицу Преподаватели. Триггер должен увеличивать счетчик числа добавленных строк.

```
CREATE OR REPLACE FUNCTION public.count_new_teachers()
```

```
    RETURNS trigger
```

```
    LANGUAGE 'plpgsql'
```

```
    COST 100
```

```
    VOLATILE NOT LEAKPROOF
```

```
AS $BODY$
```

```
begin
```

```
    update new_teachers
```

```
    set total = (select max(total) +1 from new_teachers);
```

```
    RETURN NEW;
```

```
end;
```

```
$BODY$;
```

```
ALTER FUNCTION public.count_new_teachers()
```

```
    OWNER TO postgres;
```

При добавлении новой записи в таблицу с учителями берем счётчик кол-ва учителей и увеличиваем на 1

Каталоги

Обёртки сторонних данных

Приведения

Расширения

Событийные триггеры

Схемы (1)

public

Aggregates

Operators

Анализаторы FTS

Домены

Конфигурации FTS

Материализованные I

1..3 Последовательности

Правила сортировки

Представления

Процедуры (4)

Словари FTS

Сторонние таблицы

Таблицы (6)

groups

lectors

new\_teachers

progress

students

subjects

Типы

Триггерные функции (

Запрос

История запросов

1

SELECT \* FROM public.new\_teachers

2

ORDER BY amount ASC

Data Output

Сообщения

Notifications

amount

[PK] "char"

total

integer

1

кол-во

3

2. Добавьте в таблицу Студенты поле Средний балл (Avg\_Estimate) вещественного типа со значением по умолчанию 0. Создайте хранимую процедуру, которая подсчитывает средний балл для каждого студента и заносит в поле Avg\_Estimate эту информацию. Создайте триггер, запускаемый после внесения новой информации об оценках студента и автоматически обновляет информацию о среднем балле студента.

```
CREATE OR REPLACE FUNCTION public.avg_score()
```

```
RETURNS trigger
```

```
LANGUAGE 'plpgsql'
```

```
COST 100
```

```
VOLATILE NOT LEAKPROOF
```

```
AS $BODY$
```

```
begin
```

```
    update students
```

```
    set avg_estimate = sub.avg_grade
```

```
    from
```

```
    (select code_stud, avg(estimate) as avg_grade
```

```
    from progress
```

```
    group by code_stud) as sub
```

```
    where students.codestud = sub.code_stud;
```


```
end;
```

```
$BODY$;
```

```
ALTER FUNCTION public.avg_score()
```

```
OWNER TO postgres;
```

Page No. 1

avg_estimate double precision 
4
5
5
0
0
0
0

3. Создайте триггер, запускаемый при внесении информации о новых оценках. Выполните проверку наличия информации о добавляемом студенте в таблице Студенты. Если данная информация в таблице отсутствует, то необходимо запустить хранимую процедуру на вставку записи в таблицу Студенты (параметры можно задать произвольно).

```
CREATE OR REPLACE FUNCTION public.new_marks()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
begin
    if not exists(
        select 1 from students where students.codestud = new.codestud
    ) then
        call new_stud();
    end if;

    return new;
end;
$BODY$;
```

```
ALTER FUNCTION public.new_marks()
```

```
OWNER TO postgres;
```

при добавлении нового студента идет проверка на его существование и если он не существует то вызывается процедура добавления нового студента

4. Запретить вставлять новые строки в таблицу Группы, выводя при этом сообщение «Вставка строк запрещена».

```
CREATE OR REPLACE FUNCTION public.no_add_in_group()
```

```
RETURNS trigger
```

```
LANGUAGE 'plpgsql'
```

```
COST 100
```

```
VOLATILE NOT LEAKPROOF
```

```
AS $BODY$
```

```
begin
```

```
    raise exception 'вставка запрещена';
```

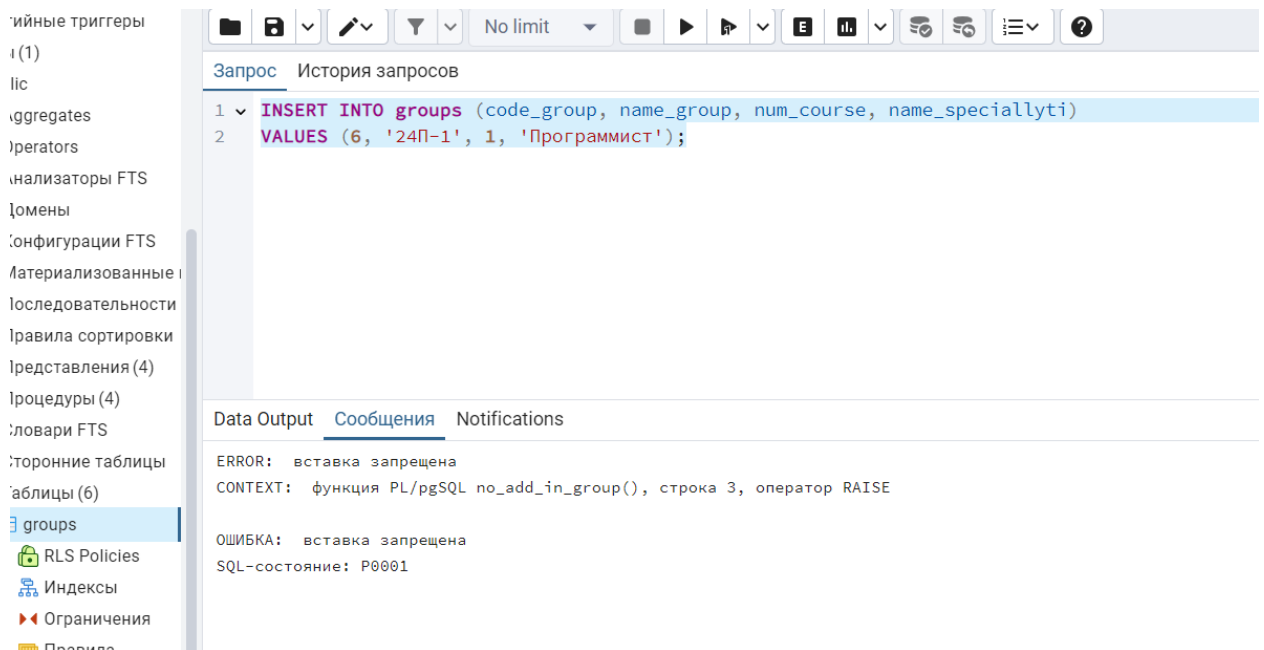
```
end;
```

```
$BODY$;
```

```
ALTER FUNCTION public.no_add_in_group()
```

```
OWNER TO postgres;
```

При вставке в таблицу группы выкидываем ошибку с текстом о запрете вставки



## Пользователи

1. Администратор – обладает всеми правами

```
REATE USER 'admin_user'@'localhost' IDENTIFIED BY 'admin_pass';  
GRANT ALL PRIVILEGES ON `db_books`. * TO 'admin_user'@'localhost';
```

Передаем все привелегии администратору

2. Диспетчер – просматривает, заполняет и изменяет справочники: книги, авторы, издательства, поставщики.

```
CREATE USER 'dispatcher_user'@'localhost' IDENTIFIED BY  
'dispatcher_pass';
```

```
GRANT SELECT, INSERT, UPDATE ON `db_books`.books TO  
'dispatcher_user'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON `db_books`.authors TO  
'dispatcher_user'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON `db_books`.publishing_house TO  
'dispatcher_user'@'localhost';
```

```
GRANT SELECT, INSERT, UPDATE ON `db_books`.suppliers TO  
'dispatcher_user'@'localhost';
```

Создаем диспетчера и даем возможность смотреть(select), заполнять(insert) и изменять (update).

3. Менеджер по работе с поставщиками – просматривает и добавляет новую информацию в справочники, оформляет поставки.

```
CREATE USER 'supplier_manager'@'localhost' IDENTIFIED BY  
'manager_pass';
```

```
GRANT SELECT, INSERT ON `db_books`.books TO  
'supplier_manager'@'localhost';
```

```
GRANT SELECT, INSERT ON `db_books`.authors TO  
'supplier_manager'@'localhost';
```

```
GRANT SELECT, INSERT ON `db_books`.publishing_house TO  
'supplier_manager'@'localhost';
```

```
GRANT SELECT, INSERT ON `db_books`.suppliers TO  
'supplier_manager'@'localhost';
```



```
GRANT SELECT, INSERT ON `db_books`.deliveries TO  
'supplier_manager'@'localhost';
```

Создаем менеджера и даем возможность смотреть и добавлять данные в бд

4. Поставщики – просматривают только свои поставки

```
CREATE VIEW my_deliveries AS  
  
SELECT * FROM deliveries WHERE Code_delivery = (SELECT  
Code_delivery FROM deliveries WHERE Code_delivery =  
CURRENT_USER());
```

```
GRANT SELECT ON my_deliveries TO  
'supplier_user'@'localhost';
```

Создаем представление для получения поставок конкретного поставщика и даем право поставщику просматривать из этого представления информацию