

Отчёт по выполненной работе по базе данных

Представление (VIEW) в базе данных — "виртуальная таблица", которая не хранит данные, а показывает их из других таблиц в момент запроса.

1. Создание представлений в MySQL. База данных «Книжное дело»

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

```

25 -- дату поставки, наименование поставщика, стоимость поставки, объем поставки.
26 • create view first_view as
27 select p.Code_purchase, b.Title_book, p.Date_order, d.Name_delivery, (p.Cost*p.Amount) as itog_sum, p.Amount as volume from purchases p
28 join books b on b.Code_book = p.Code_book
29 join deliveries d on d.Code_delivery = p.Code_delivery
30 group by p.Code_purchase;
31
32 • select * from first_view;

```

Code_purchase	Title_book	Date_order	Name_delivery	itog_sum	volume
21	Война и мир	2003-12-30	Алексеев Д.В.	15030	45
22	Война и мир	2003-05-14	Смирнов А.Н.	49875	105
23	Вий	2003-01-29	Фомин В.С.	11400	25
24	Ночь перед Новым годом	2002-01-01	Фомин В.С.	936	4
26	Тенные аллеи	2006-04-20	Романова Е.И.	19725	25
27	Преступление и наказание	2005-01-01	Смирнов А.Н.	726	3

- б. Создайте представление, которое показывает все сведения об издательствах из города Москва.

```

34 -- все сведения об издательствах из Москвы
35 • create view Moscow_publish as
36 select * from publishing_house
37 where City = 'Москва';
38
39 • select * from Moscow_publish;

```

Code_publish	Publish	City
11	АСТ	Москва
12	Союз писателей	Москва

- ```

41 -- код книги, название, автор, кол-во на складе, макс стоимость книги
42 • select * from purchases;
43
44 • create view maxxxx as
45 select b.Code_book, b.Title_book, a.name_author, sum(p.Amount) as total_quant, max(p.cost * p.amount) as max_stoim from books b
46 join authors a on a.Code_author = b.Code_author
47 join purchases p on p.Code_book = b.Code_book
48 group by b.Code_book, b.Title_book
49 limit 10;
50
51 • select * from maxxxx;
52

```
- Result Grid

Filter Rows:

Export:

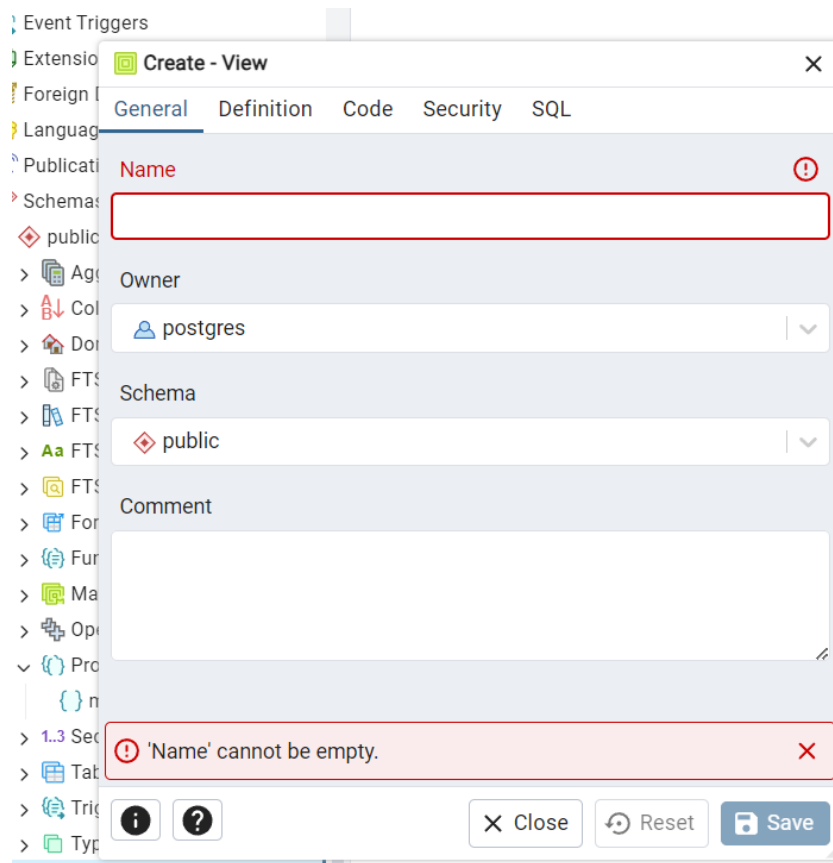
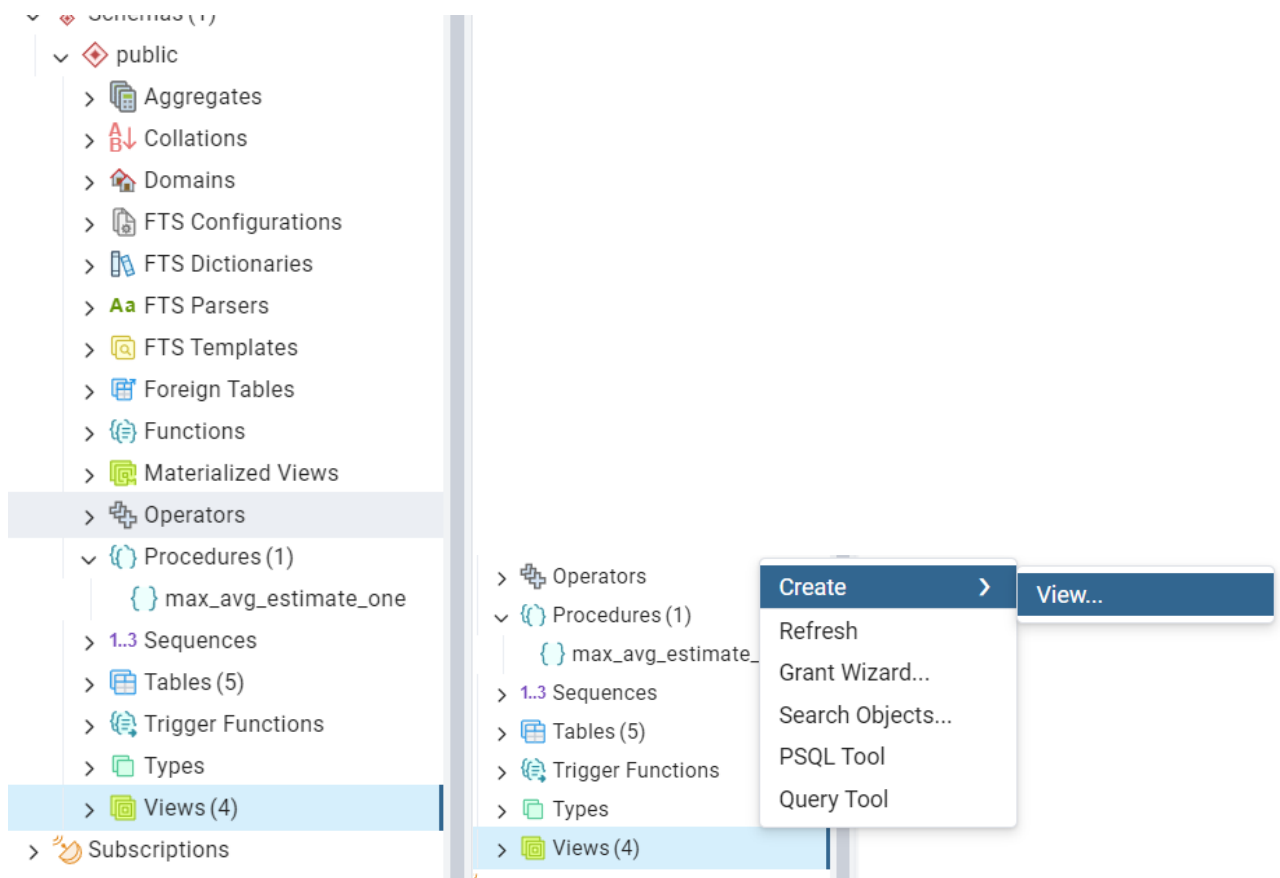
Wrap Cell Content: [IA](#)

|   | Code_book | Title_book               | name_author      | total_quant | max_stoim |
|---|-----------|--------------------------|------------------|-------------|-----------|
| ▶ | 102       | Война и мир              | Толстой Л.Н.     | 150         | 49875     |
|   | 104       | Вий                      | Гоголь Н.В.      | 25          | 11400     |
|   | 105       | Ночь перед Новым годом   | Гоголь Н.В.      | 4           | 936       |
|   | 107       | Тёмные аллеи             | Бунин И.А.       | 25          | 19725     |
|   | 101       | Преступление и наказание | Достоевский Ф.М. | 3           | 726       |

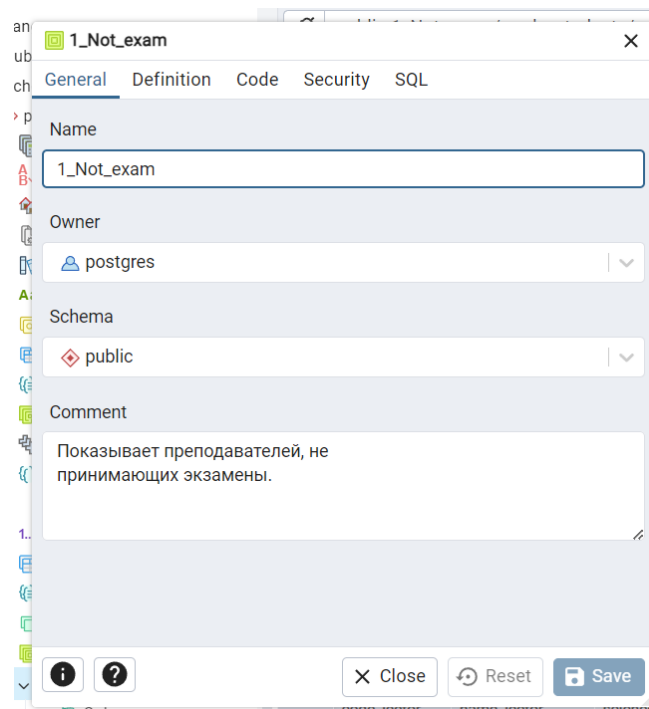
- ```
53 -- топ 5 книг с максимальным количеством на складе
54 • select b.Code_book, b.Title_book, a.name_author, sum(p.Amount) as total_quant from books b
55 join authors a on a.Code_author = b.Code_author
56 join purchases p on p.Code_book = b.Code_book
57 group by b.Code_book, b.Title_book, a.name_author
58 order by total_quant desc
59 limit 5;
60
61 • select * from purchases;
62
```

[illegible]

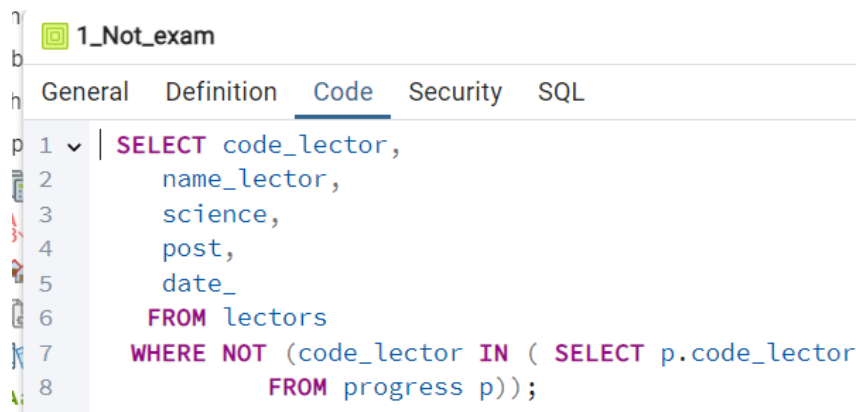
2. Создание представлений в PostgreSQL. База данных «Успеваемость»



- а. Создайте представление, которое показывает преподавателей, не принимающих экзамены.

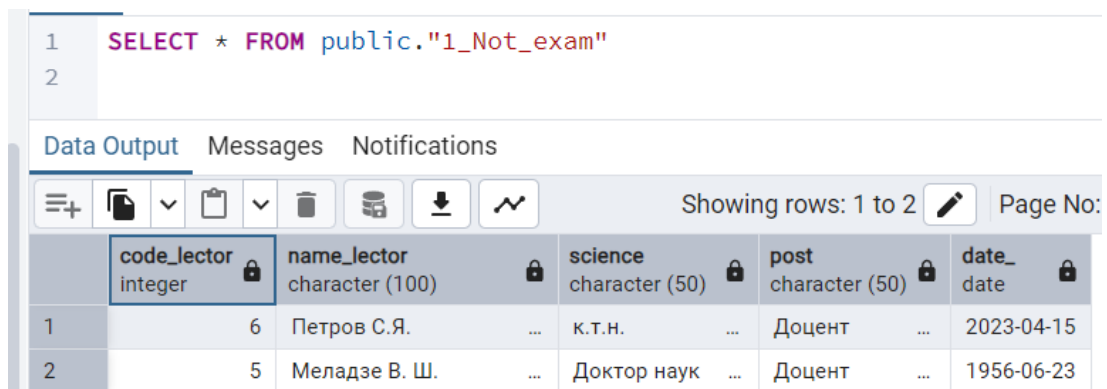


Код представления



```
SELECT code_lector,  
       name_lector,  
       science,  
       post,  
       date_  
FROM lectors  
WHERE NOT (code_lector IN ( SELECT p.code_lector  
                           FROM progress p));
```

Вывод



```
1 SELECT * FROM public."1_Not_exam"  
2
```

Showing rows: 1 to 2							
	code_lector integer	name_lector character (100)	science character (50)	post character (50)	date_ date		
1	6	Петров С.Я.	К.Т.Н.	Доцент	2023-04-15		
2	5	Меладзе В. Ш.	Доктор наук	Доцент	1956-06-23		

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

2_Name_speciality

General Definition Code Security SQL

Name: 2_Name_speciality

Owner: postgres

Schema: public

Comment: Представление списка групп определенной специальности

Close Reset Save

Код представления

2_Name_speciality

General Definition **Code** Security SQL

```

1 SELECT code_group,
2       name_group,
3       num_course,
4       name_speciality
5 FROM groups
6 WHERE name_speciality = 'Правоохранительная деятельность'

```

Вывод

Query Query History

```

1 SELECT * FROM public."2_Name_speciality"
2

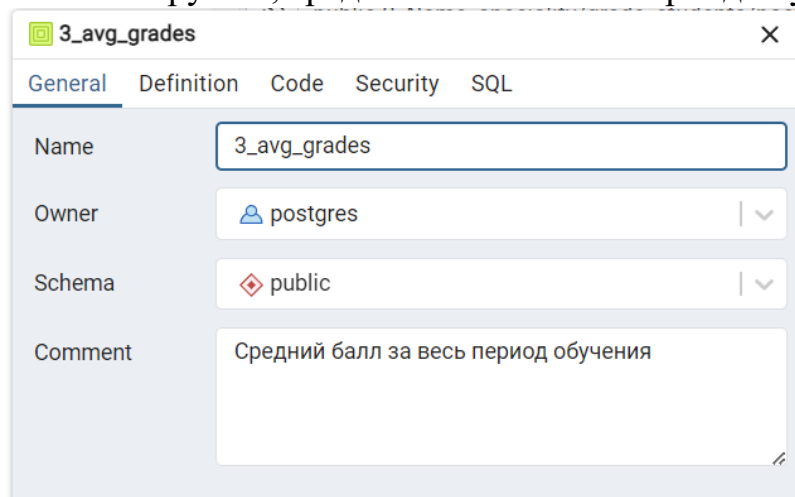
```

Data Output Messages Notifications

Showing rows: 1 to 2 Page

	code_group integer	name_group character (50)	num_course integer	name_speciality character (100)
1	3	21ПД-2	4	Правоохранительная деятельнос...
2	2	22ПД-1	3	Правоохранительная деятельнос...

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



3_avg_grades

General Definition Code Security SQL

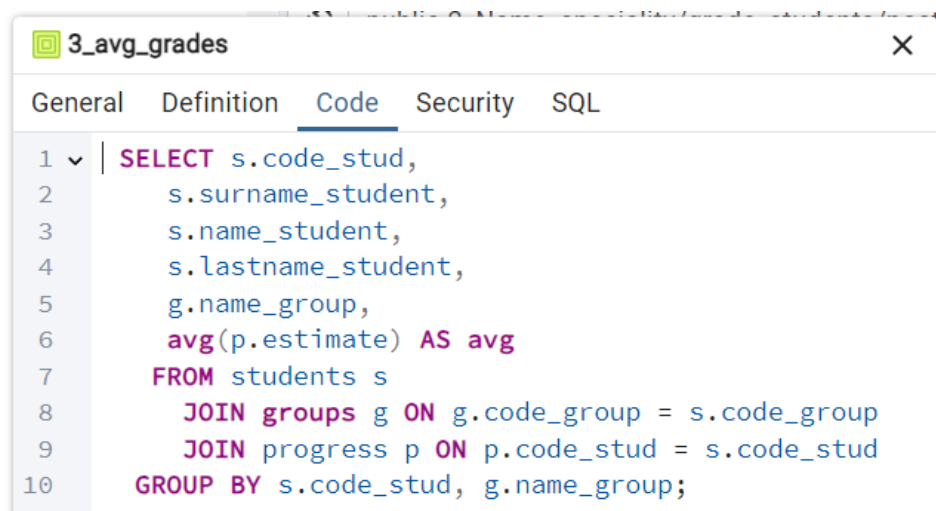
Name: 3_avg_grades

Owner: postgres

Schema: public

Comment: Средний балл за весь период обучения

Код представления



3_avg_grades

General Definition **Code** Security SQL

```
1 SELECT s.code_stud,  
2       s.surname_student,  
3       s.name_student,  
4       s.lastname_student,  
5       g.name_group,  
6       avg(p.estimate) AS avg  
7 FROM students s  
8     JOIN groups g ON g.code_group = s.code_group  
9     JOIN progress p ON p.code_stud = s.code_stud  
10    GROUP BY s.code_stud, g.name_group;
```

Вывод

```
1 SELECT * FROM public."3_avg_grades"  
2
```

Data Output Messages Notifications

	code_stud character (20)	surname_student character (50)	name_student character (50)	lastname_student character (50)	name_group character (50)	avg numeric
1	15	Киселева	Кристина	Рустамовна	21ПД-2	3.0000000000000000
2	14	Бастанова	Регина	Динисламовна	22ПД-1	3.0000000000000000
3	11	Сафина	Регина	Рифовна	22П-1	5.0000000000000000
4	12	Узбекович	Хасбик	нет сведений	22КСК-1	2.0000000000000000
5	13	Винковна	Лейла	Петровна	23ЗИО-1	4.2000000000000000

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

4_avg_grades

General Definition Code Security SQL

Name: 4_avg_grades

Owner: postgres

Schema: public

Comment: 3 студента с наибольшим средним баллов

Код представления

4_avg_grades

General Definition Code Security SQL

```
1 SELECT s.code_stud,
2     s.surname_student,
3     s.name_student,
4     s.lastname_student,
5     g.name_group,
6     avg(p.estimate) AS grades
7 FROM students s
8     JOIN groups g ON g.code_group = s.code_group
9     JOIN progress p ON p.code_stud = s.code_stud
10 GROUP BY s.code_stud, g.name_group
11 ORDER BY (avg(p.estimate)) DESC;
```

Вывод

1 SELECT * FROM public."4_avg_grades"

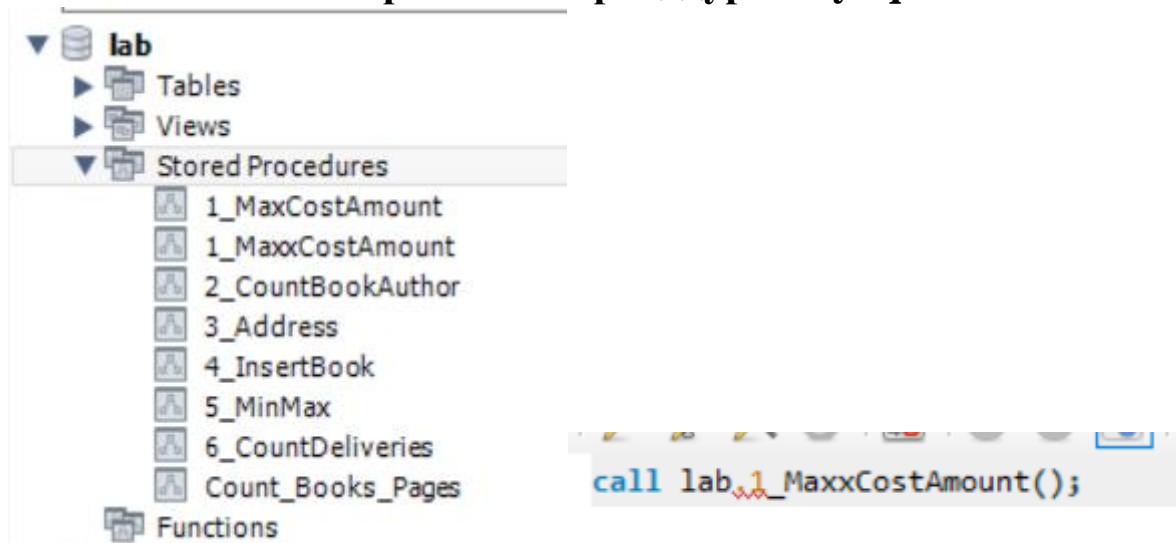
2

Data Output Messages Notifications

Showing rows: 1 to 5 Page No: 1 of 1

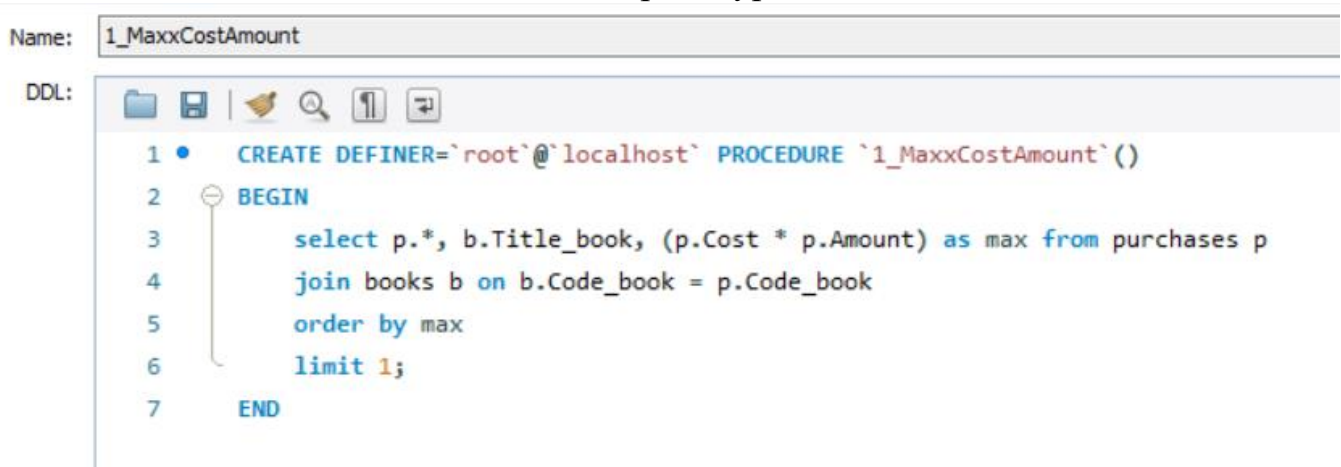
	code_stud character (20)	surname_student character (50)	name_student character (50)	lastname_student character (50)	name_group character (50)	grades numeric
1	11	Сафина	Регина	Рифовна	22П-1	5.0000000000000000
2	13	Винксовна	Лейла	Петровна	23ЗИО-1	4.2000000000000000
3	15	Киселева	Кристина	Рустамовна	21ПД-2	3.0000000000000000
4	14	Бастанова	Регина	Динисламовна	22ПД-1	3.0000000000000000
5	12	Узбекович	Хасбик	нет сведений	22КСК-1	2.0000000000000000

3. Хранимые процедуры MySQL



- а. 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).

Код процедуры



Вывод

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount	Title_book	max
27	101	2005-01-01	3004	0	242	3	Преступление и наказание	726

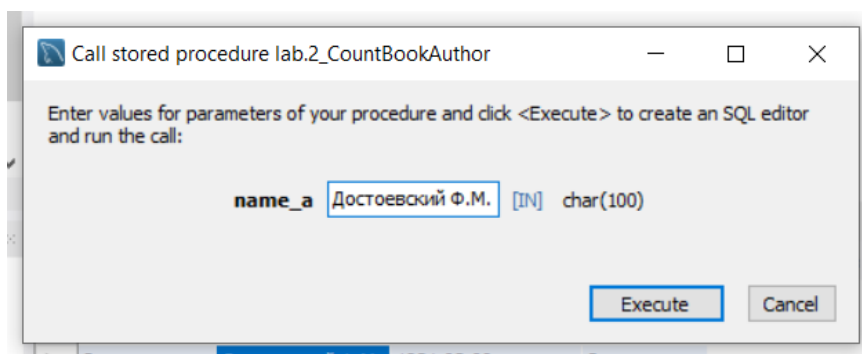
- б. Сосчитать количество книг определенного автора (ФИО автора является входным параметром).

Код процедуры

Name: 2_CountBookAuthor

DDL:

```
1 CREATE DEFINER=`root`@`localhost` PROCEDURE `2_CountBookAuthor`(name_a char(100))
2 BEGIN
3     select name_a, count(b.Code_book) from books b
4     join authors a on a.Code_author = b.Code_author
5     where a.name_author = name_a;
6 END
```



Вызов процедуры

```
call lab.2_CountBookAuthor('Достоевский Ф.М.');
```

Вывод процедуры

Result Grid

name_a	count(b.Code_book)
Достоевский Ф.М.	4

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

Изначальные данные

1 • `SELECT * FROM lab.deliveries;`

Code_delivery	Name_delivery	Name_company	Address	Phone	INN
3001	Фомин В.С.	ООО "КНИЖНЫЙ МИР"	г. Туймазы, пр-кт Ленина, д. 50	79903233323	4537658686
3002	Романова Е.И.	ООО "ЧИТАЙ-ГОРОД"	г. Москва, ш. Варшавское, д. 9, стр. 1	79115677766	2987770697
3003	Алексеев Д.В.	ОАО "БУКВА"	NULL	79991248989	1122324555
3004	Смирнов А.Н.	ОАО "КНИЖНАЯ ЛАВКА"	NULL	89992223344	NULL
3005	Иванова Л.В.	ООО "УКСИВТ"	г.Уфа, ул.Кирова 65, д.2	89613601151	NULL
3006	не известен	не известен	не известен	8900000	не извес...
3007	не известен	не известен	не известен	8900000	не извес...
3008	не известен	не известен	не известен	8900000	не извес...
3009	не известен	не известен	не известен	8900000	не извес...
3010	не известен	не известен	не известен	8900000	не извес...
NULL	NULL	NULL	NULL	NULL	NULL

Call stored procedure lab.3_Address

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

name_deliv [IN] char(30)

Execute Cancel

Вызов и вывод

```
1 • set @address_deliver = '0';
2 ✖ call lab.3_Address('Фомин В.С.', @address_deliver);
3 • select @address_deliver;
4
```

Result Grid

@address_deliver
г. Туймазы, пр-кт Ленина, д. 50

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

Изначальные данные

	Code_book	Title_book	Code_author	Pages	Code_publish
	38	Наука.Техника.Инновации	1	345	12
	101	Преступление и наказание	2	600	13
	102	Война и мир	1	1200	11
	103	Идиот	2	400	11
	104	Вий	3	399	12
	105	Ночь перед Новым годом	3	100	13
	106	Герой нашего времени	2	250	13
	107	Тёмные аллеи	5	150	14
	111	Мёртвые души	3	333	11
▶	113	Очень крутая книга	4	25	14
	114	Книга	1	333	14
★	NULL	NULL	NULL	NULL	NULL

Входные параметры

4_InsertBook

The name of the routine is parsed a statement. The DDL is parsed auto

1 • CREATE DEFINER='root'@'localhost' PROCEDURE `4_InsertBook` (title_b char(45), code_a int, pages_book int, code_pub int)

2 BEGIN

3 insert into books (Code_book, Title_book, Code_author, Pages, Code_publish)

4 value ((select * from(select max(Code_book) + 1 from books) as max), title_b, code_a, pages_book, code_pub);

5 END

e ((select * from(select max(Code_book) + 1 from books) as max), title_b, (

Call stored procedure lab.4_InsertBook

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

title_b

[IN] char(45)

code_a

[IN] int

pages_book

[IN] int

code_pub

[IN] int

Execute

Cancel

Вывод данных

	Code_book	Title_book	Code_author	Pages	Code_publish
	38	Наука.Техника.Инновации	1	345	12
	101	Преступление и наказание	2	600	13
	102	Война и мир	1	1200	11
	103	Идиот	2	400	11
	104	Вий	3	399	12
	105	Ночь перед Новым годом	3	100	13
	106	Герой нашего времени	2	250	13
	107	Тёмные аллеи	5	150	14
	111	Мёртвые души	3	333	11
▶	113	Очень крутая книга	4	25	14
	114	Книга	1	333	14
	115	Новая книга	5	532	11
★	NULL	NULL	NULL	NULL	NULL

е. 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».

Код процедуры

Name: 5_MinMax

DDL:

```

1 • CREATE DEFINER='root'@'localhost' PROCEDURE `5_MinMax`()
2 BEGIN
3     declare minCost float;
4     declare maxCost float;
5
6     select min(Cost * Amount), max(Cost * Amount) into minCost, maxCost from purchases;
7
8     select *,
9     case
10         when Cost * Amount = minCost then 'Минимальная стоимость'
11         when Cost * Amount = maxCost then 'Максимальная стоимость'
12         else 'Средняя стоимость'
13     end as CostCategory
14     from purchases;
15 END

```

Вывод данных

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount
21	102	2003-12-30	3003	1	334	45
22	102	2003-05-14	3004	1	475	105
23	104	2003-01-29	3001	1	456	25
24	105	2002-01-01	3001	0	234	4
26	107	2006-04-20	3002	1	789	25
27	101	2005-01-01	3004	0	242	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount	CostCategory
21	102	2003-12-30	3003	1	334	45	Средняя стоимость
22	102	2003-05-14	3004	1	475	105	Максимальная стоимость
23	104	2003-01-29	3001	1	456	25	Средняя стоимость
24	105	2002-01-01	3001	0	234	4	Средняя стоимость
26	107	2006-04-20	3002	1	789	25	Средняя стоимость
27	101	2005-01-01	3004	0	242	3	Минимальная стоимость

- f. 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'.

	Code_delivery	Name_delivery	Name_company	Address	Phone	INN
	3001	Фомин В.С.	ООО "КНИЖНЫЙ МИР"	г. Туймазы, пр-кт Ленина, д. 50	79903233323	4537658686
	3002	Романова Е.И.	ООО "ЧИТАЙ-ГОРОД"	г. Москва, ш. Варшавское, д. 9, стр. 1	79115677766	2987770697
	3003	Алексеев Д.В.	ОАО "БУКВА"	NULL	79991248989	1122324555
	3004	Смирнов А.Н.	ОАО "КНИЖНАЯ ЛАВКА"	NULL	89992223344	NULL
	3005	Иванова Л.В.	ООО "УКСИВТ"	г.Уфа, ул.Кирова 65, д.2	89613601151	NULL

Код процедуры

Name: 6_CountDeliveries

DDL:

```

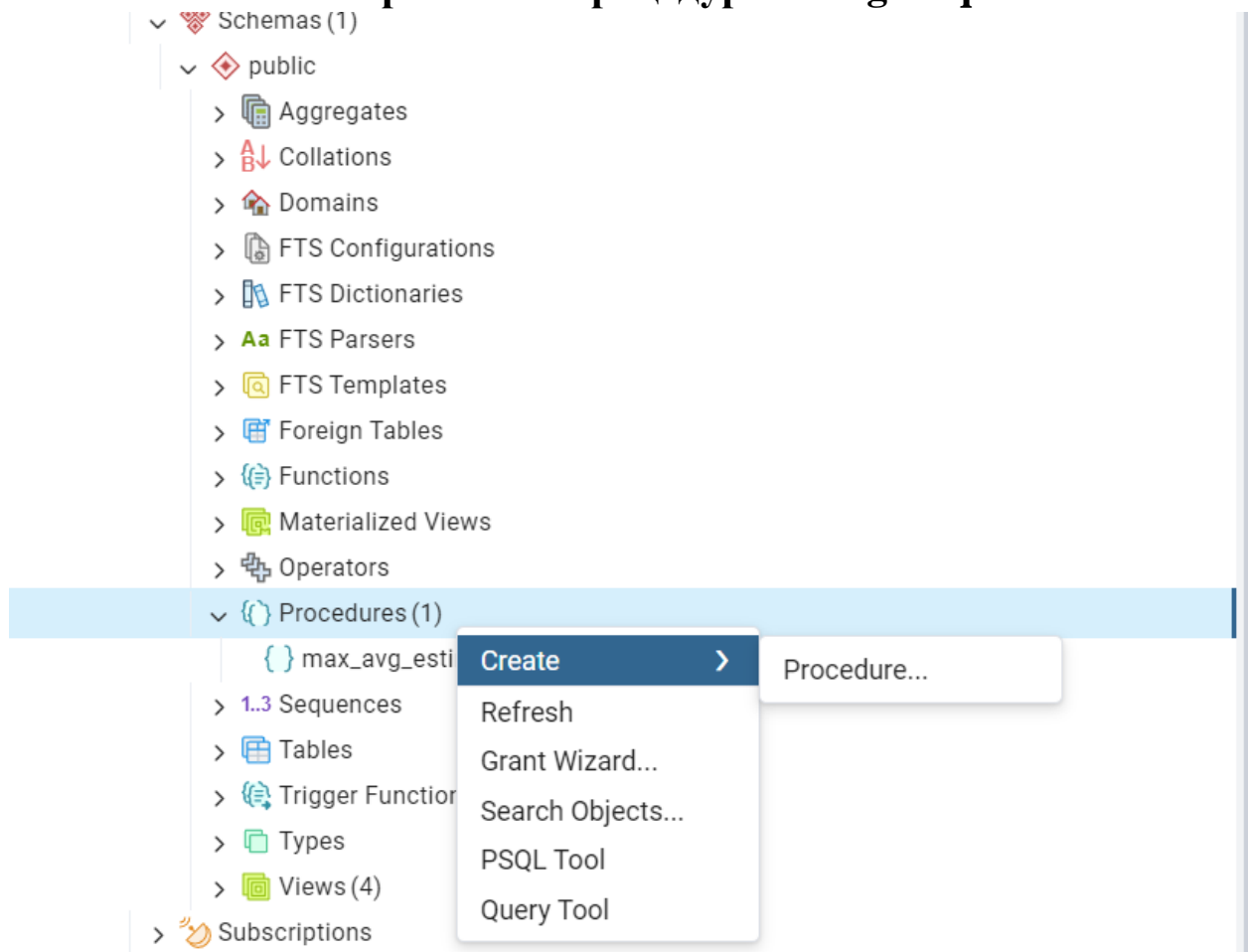
1 CREATE DEFINER='root'@'localhost' PROCEDURE `6_CountDeliveries`()
2 BEGIN
3     declare z int;
4     select count(*) into z from deliveries;
5
6     while z < 10 do
7         insert into deliveries (Code_delivery, Name_delivery, Name_company, Address, Phone, INN)
8             values ((select * from(select max(Code_delivery) + 1 from deliveries) as maxCode), 'не известен', 'не известен', 'не известен', 8900000, 'не известен');
9
10        select count(*) into z from deliveries;
11    end while;
12 END

```

Итоги

	Code_delivery	Name_delivery	Name_company	Address	Phone	INN
▶	3001	Фомин В.С.	ООО "КНИЖНЫЙ МИР"	г. Туймазы, пр-кт Ленина, д. 50	79903233323	4537658686
	3002	Романова Е.И.	ООО "ЧИТАЙ-ГОРОД"	г. Москва, ш. Варшавское, д. 9, стр. 1	79115677766	2987770697
	3003	Алексеев Д.В.	ОАО "БУКВА"	NULL	79991248989	1122324555
	3004	Смирнов А.Н.	ОАО "КНИЖНАЯ ЛАВКА"	NULL	89992223344	NULL
	3005	Иванова Л.В.	ООО "УКСИВТ"	г.Уфа, ул.Кирова 65, д.2	89613601151	NULL
	3006	не известен	не известен	не известен	89000000	не извес...
	3007	не известен	не известен	не известен	89000000	не извес...
	3008	не известен	не известен	не известен	89000000	не извес...
	3009	не известен	не известен	не известен	89000000	не извес...
	3010	не известен	не известен	не известен	89000000	не извес...
*	NULL	NULL	NULL	NULL	NULL	NULL

4. Хранимые процедуры PostgreSQL



- а. 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).

Создание процедуры

The screenshot shows the 'max_avg_estimate_one' procedure creation dialog box. The 'General' tab is active, displaying the following fields:

- Name:** max_avg_estimate_one
- Owner:** postgres
- Schema:** public
- Comment:** Вывести ФИО студента с максимальным средним баллом

Код процедуры

```
{ } max_avg_estimate_one
```

General Definition **Code** Options Parameters Security SQL

```
1 |
2 v select s.surname_student, s.name_student, s.code_stud from students s
3 join progress p on p.code_stud = s.code_stud
4 group by surname_student, name_student, s.code_stud
5 having avg(p.estimate) = (select max(avg_es) from
6                          (select avg(estimate) as avg_es from progress
7                           group by code_stud));
8
```

Вызов процедуры

Query Query History

```
1 call max_avg_estimate_one();
```

Data Output **Messages** Notifications

CALL

Query returned successfully in 104 msec.

Вывод данных

Query Query History

```
1 v select s.surname_student, s.name_student, s.code_stud from students s
2 join progress p on p.code_stud = s.code_stud
3 group by surname_student, name_student, s.code_stud
4 having avg(p.estimate) = (select max(avg_es) from
5                          (select avg(estimate) as avg_es from progress
6                           group by code_stud));
6
```

Data Output Messages Notifications

Showing rows: 1 to 1 Page No: 1

	surname_student character (50)	name_student character (50)	code_stud [PK] character (20)
1	Сафина	Регина	11

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

Код и вывод данных

```
1 create or replace procedure get_avg_estimate(  
2     in student_name VARCHAR,  
3     in student_surname VARCHAR)  
4  
5     language plpgsql  
6     as $$  
7     declare avg_estimate FLOAT;  
8     begin  
9         select avg(p.estimate) into avg_estimate  
10        from students s  
11        join progress p on s.code_stud = p.code_stud  
12        where s.name_student = student_name and s.surname_student = student_surname;  
13  
14        raise notice 'Средняя оценка: %', avg_estimate;  
15    end;  
16  
17    CALL get_avg_estimate('Регина', 'Сафина');
```

Data Output Messages Notifications

ЗАМЕЧАНИЕ: Средняя оценка: 5
CALL

Query returned successfully in 72 msec.

> Operators

✓ Procedures (2)

{ } get_avg_estimate(IN student_name character varying, IN student_surname

{ } max_avg_estimate_one

1 3 Sequences

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

Код и вывод данных

```

1 create or replace procedure get_specialt_course(
2     in student_name VARCHAR,
3     in student_surname VARCHAR,
4     out name_speciality VARCHAR,
5     out num_course INTEGER)
6
7 language plpgsql
8 as $$
9 begin
10     select g.name_speciality, g.num_course into name_speciality, num_course from groups g
11     join students s on s.code_group = g.code_group
12     where s.name_student = student_name and s.surname_student = student_surname;
13 end;
14 $$;
15
16 CALL get_specialt_course('Регина', 'Сафина', null, null);

```

Data Output Messages Notifications

Showing rows: 1 to 1

	name_speciality character varying	num_course integer
1	Информационные системы и программирование	3

> Operators
 > Procedures (3)
 {} get_avg_estimate(IN student_name character varying, IN student_surname character varying)
 {} get_specialt_course(IN student_name character varying, IN student_surname character varying, OUT name_speciality character varying, OUT num_course integer)
 {} max_avg_estimate_one
 > 1.3 Sequences
 > Tables (5)

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

	code_stud [PK] character (20)	surname_student character (50)	name_student character (50)	lastname_student character (50)	code_group integer	birthday date	phone numeric (15)
1	11	Сафина	Регина	Рифовна	1	2006-02-27	89613601151
2	12	Узбекович	Хасбик	нет сведений	6	2006-02-26	89874839939
3	13	Винксовна	Лейла	Петровна	4	2007-07-03	89639573384
4	14	Бастанова	Регина	Динисламовна	2	2006-03-13	89053502859
5	15	Киселева	Кристина	Рустамовна	3	2005-01-01	89063743585

```

1 create procedure insertt_user(
2     in name_s VARCHAR, in surname VARCHAR,
3     in lastname VARCHAR, in cod_group INTEGER,
4     in birth DATE, in phone_num NUMERIC)
5
6 language plpgsql
7 as $$
8 begin
9     insert into students (surname_student, name_student, lastname_student, code_group, birthday, phone)
10    values (surname, name_s, lastname, cod_group, birth, phone_num);
11 end;
12 $$;
13
14 CALL insertt_user('Марат', 'Сафиуллин', 'Иннокентьевич', 3, '2001-01-25', 89638594453);

```

Data Output Messages Notifications

CALL

Query returned successfully in 122 msec.

	code_stud [PK] character (20)	surname_student character (50)	name_student character (50)	lastname_student character (50)	code_group integer	birthday date	phone numeric (15)
1	11	Сафина	Регина	Рифовна	1	2006-02-27	89613601151
2	12	Узбекович	Хасбик	нет сведений	6	2006-02-26	89874839939
3	13	Винковна	Лейла	Петровна	4	2007-07-03	89639573384
4	14	Бастанова	Регина	Динисламовна	2	2006-03-13	89053502859
5	15	Киселева	Кристина	Рустамовна	3	2005-01-01	89063743585
6	16	Сафиуллин	Марат	Иннокентьевич	3	2001-01-25	89638594453

- е. 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».

```

1 create function check_student_ages()
2 RETURNS TABLE (
3     surname_student VARCHAR(50),
4     name_student VARCHAR(50),
5     age NUMERIC,
6     age_message TEXT)
7 language plpgsql
8 as $$
9 declare
10     avg_age FLOAT;
11 begin
12     select avg(extract(year from AGE(birthday))) into avg_age from students;
13
14     return query
15     select s.surname_student, s.name_student, extract(year from age(s.birthday)) as Age,
16     case
17         when extract(year from age(s.birthday)) > avg_age then 'Ваш возраст больше среднего возраста всех студентов'
18         when extract(year from age(s.birthday)) < avg_age then 'Ваш возраст меньше среднего возраста всех студентов'
19         else 'Ваш возраст равен среднему возрасту всех студентов'
20     end as age_message
21 from students s;
22 end;
23 $$;
24
25 SELECT * FROM check_student_ages();

```

Вывод данные

	surname_student character varying	name_student character varying	age numeric	age_message text
1	Бастанова	Регина	19	Ваш возраст меньше среднего возраста всех студент...
2	Винксовна	Лейла	17	Ваш возраст меньше среднего возраста всех студент...
3	Сафина	Регина	19	Ваш возраст меньше среднего возраста всех студент...
4	Киселева	Кристина	20	Ваш возраст больше среднего возраста всех студентов
5	Узбекович	Хасбик	19	Ваш возраст меньше среднего возраста всех студент...
6	Иванов	Иван	21	Ваш возраст больше среднего возраста всех студентов
7	Сафиуллин	Марат	24	Ваш возраст больше среднего возраста всех студентов

Изначальные данные

	code_stud [PK] character (20)	surname_student character (50)	name_student character (50)	lastname_student character (50)	code_group integer	birthday date	phone numeric (15)
1	11	Сафина	Регина	Рифовна	1	2006-02-27	89613601151
2	12	Узбекович	Хасбик	нет сведений	6	2006-02-26	89874839939
3	13	Винксовна	Лейла	Петровна	4	2007-07-03	89639573384
4	14	Бастанова	Регина	Динисламовна	2	2006-03-13	89053502859
5	15	Киселева	Кристина	Рустамовна	3	2005-01-01	89063743585
6	16	Сафиуллин	Марат	Иннокентьевич	3	2001-01-25	89638594453
7	5	Иванов	Иван	Иванович	3	2004-01-31	89010000000

- f. 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'.

Изначальные данные

	code_subject [PK] integer	name_subject character (100)	count_hours integer
1	1	ОБЖ	122
2	2	Математика	152
3	3	Философия	58
4	4	Право	202

Код процедуры

query query history

```
1  ✓ create function fill_subject()
2  returns INTEGER
3  language plpgsql
4  as $$
5  declare
6      rec_count INTEGER;
7      next_id INTEGER;
8  ✓ begin
9      select count(*) into rec_count from subjects;
10     |
11     select max(code_subject), 0) + 1 into next_id from subjects;
12
13  ✓ while rec_count < 10 loop
14      insert into subjects (code_subject, name_subject, count_hours)
15      values (next_id, 'не известно', 0);
16
17      rec_count := rec_count + 1;
18      next_id := next_id + 1;
19  end loop;
20
21  return rec_count;
22 end;
23 $$;
24
25 select fill_subject();
```

Вывод данных

	code_subject [PK] integer	name_subject character (100)	count_hours integer
1	1	ОБЖ	122
2	2	Математика	152
3	3	Философия	58
4	4	Право	202
5	5	не известно	0
6	6	не известно	0
7	7	не известно	0
8	8	не известно	0
9	9	не известно	0
10	10	не известно	0

5. Триггеры MySQL

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

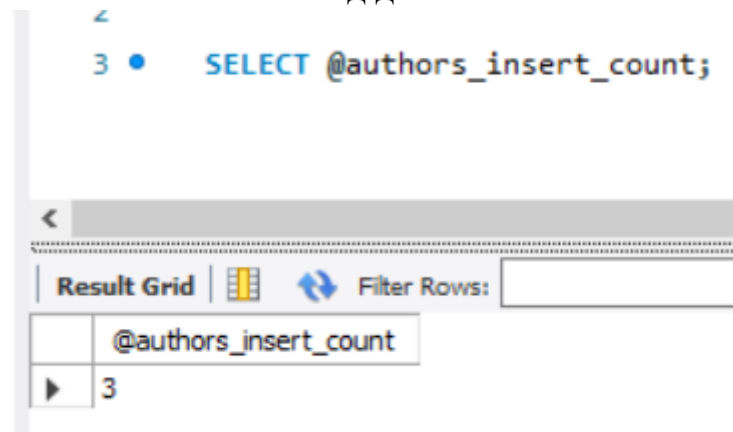
Код триггеров

```
1 CREATE DEFINER='root'@'localhost' TRIGGER `authors_AFTER_INSERT` AFTER INSERT ON `authors` FOR EACH ROW BEGIN
2 SET @authors_insert_count = IFNULL(@authors_insert_count, 0) + 1;
3 END
```

Вставка новых строк



Вывод данных



- б. 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.

Изначальные данные

	Code_author	name_author	Birthday	book_count
	1	Толстой Л.Н.	1828-09-09	0
	2	Достоевский Ф.М.	1821-02-09	0
	3	Гоголь Н.В.	1809-03-20	0
	4	Лермонтов М.Ю.	1814-10-15	0
	5	Бунин И.А.	1870-10-10	0
	6	Сафина Р.Р.	2006-02-27	0
	7	Чехов А.П.	1860-01-29	0
▶	8	Есенин С.А.	1895-10-03	0
*	NULL	NULL	NULL	NULL

Код триггера

Name: UpdateBookCount

DDL:

```

1 • CREATE DEFINER='root'@'localhost' PROCEDURE `UpdateBookCount`()
2 BEGIN
3     UPDATE authors a
4     SET a.book_count = (
5         SELECT COUNT(*)
6         FROM books b
7         WHERE b.author_id = a.author_id
8     );
9 END

```

BEFORE INSERT
 ▼ AFTER INSERT
 books_AFTER_INSERT
 BEFORE UPDATE
 AFTER UPDATE
 BEFORE DELETE
 AFTER DELETE

```

1 • CREATE DEFINER='root'@'localhost' TRIGGER `books_AFTER_INSERT` AFTER INSERT ON `books` FOR EACH ROW BEGIN
2     UPDATE authors
3     SET book_count = book_count + 1
4     WHERE code_author = NEW.code_author;
5 END

```

Проверка данных

Review the SQL Script to be Applied on the Database

```

1 INSERT INTO `lab`.`books` (`Code_book`, `Title_book`, `Code_author`, `Pages`, `Code_publish`) VALUES ('116', 'Об колледже УКСИВТ', '6', '555', '12');
2 INSERT INTO `lab`.`books` (`Code_book`, `Title_book`, `Code_author`, `Pages`, `Code_publish`) VALUES ('117', 'Новая книга 2 часть', '6', '56', '12');
3

```

Вывод данных

	Code_author	name_author	Birthday	book_count
▶	1	Толстой Л.Н.	1828-09-09	3
	2	Достоевский Ф.М.	1821-02-09	3
	3	Гоголь Н.В.	1809-03-20	3
	4	Лермонтов М.Ю.	1814-10-15	1
	5	Бунин И.А.	1870-10-10	2
	6	Сафина Р.Р.	2006-02-27	2
	7	Чехов А.П.	1860-01-29	0
	8	Есенин С.А.	1895-10-03	0
•	NULL	NULL	NULL	NULL

с. 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 %.

Вывод данных

1 • `SELECT * FROM lab.purchases;`

	Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount
▶	21	102	2003-12-30	3003	1	334	45
	22	102	2003-05-14	3004	1	475	105
	23	104	2003-01-29	3001	1	456	25
	24	105	2002-01-01	3001	0	234	4
	26	107	2006-04-20	3002	1	789	25
	27	101	2005-01-01	3004	0	242	3
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Код триггера

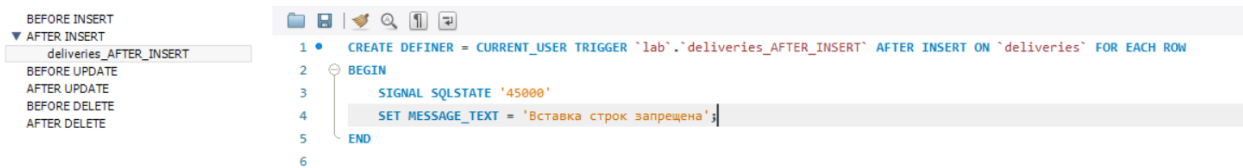
BEFORE INSERT	
▼ AFTER INSERT	
purchases_AFTER_INSERT	
BEFORE UPDATE	
AFTER UPDATE	
BEFORE DELETE	
AFTER DELETE	

```

1 • CREATE DEFINER='root'@'localhost' TRIGGER `purchases_AFTER_INSERT` AFTER INSERT ON `purchases` FOR EACH ROW BEGIN
2     DECLARE book_quantity INT;
3     DECLARE current_price DECIMAL(10, 2);
4
5     -- Получаем текущее количество и цену книги
6     SELECT Amount, Cost INTO book_quantity, current_price FROM books
7     WHERE Code_book = NEW.Code_book;
8     IF (book_quantity + NEW.Amount) < 10 THEN
9         UPDATE books
10        SET Amount = Amount * 1.20
11        WHERE Code_book = NEW.Code_book;
12    END IF;
13 END

```


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



The screenshot shows the SQL Developer interface. On the left, a tree view shows the database structure with a trigger named 'deliveries_AFTER_INSERT' under the 'AFTER INSERT' event. The main window displays the SQL script for creating this trigger:

```
1 CREATE DEFINER = CURRENT_USER TRIGGER `lab`.`deliveries_AFTER_INSERT` AFTER INSERT ON `deliveries` FOR EACH ROW
2 BEGIN
3     SIGNAL SQLSTATE '45000'
4     SET MESSAGE_TEXT = 'Вставка строк запрещена';
5 END
6
```

Review the SQL Script to be Applied on the Database

```
1 INSERT INTO `lab`.`deliveries` (`Code_delivery`, `Name_delivery`, `Name_comp
2 INSERT INTO `lab`.`deliveries` (`Code_delivery`, `Name_delivery`, `Name_comp
3
```

❌ Execute SQL Statements

Error: There was an error while applying the SQL script to the database.

Message Log

Executing:
INSERT INTO `lab`.`deliveries` (`Code_delivery`, `Name_delivery`, `Name_company`, `Address`,
`Phone`, `INN`) VALUES ('3011', 'Файзуллин Э. Э.', 'ОАО \УКСИВТСКИЙ\',"г.Уфа, ул.
Ветошникова, д.95', '89874940637', '1122324555');
INSERT INTO `lab`.`deliveries` (`Code_delivery`, `Name_delivery`, `Name_company`, `Address`,
`Phone`, `INN`) VALUES ('3012', 'Быков А.В.', 'ООО \Море шоколада\',"г.Волгоград,
ул.Гагарина,д.109', '89873487233', '3453453423');

Operation failed: There was an error while applying the SQL script to the database.
ERROR 1644: 1644: Вставка строк запрещена
SQL Statement:
INSERT INTO `lab`.`deliveries` (`Code_delivery`, `Name_delivery`, `Name_company`, `Address`,
`Phone`, `INN`) VALUES ('3011', 'Файзуллин Э. Э.', 'ОАО \УКСИВТСКИЙ\',"г.Уфа, ул.
Ветошникова, д.95', '89874940637', '1122324555')

ERROR 1644: 1644: Вставка строк запрещена
SQL Statement:
INSERT INTO `lab`.`deliveries` (`Code_delivery`, `Name_delivery`, `Name_company`, `Address`,
`Phone`, `INN`) VALUES ('3012', 'Быков А.В.', 'ООО \Море шоколада\',"г.Волгоград,
ул.Гагарина,д.109', '89873487233', '3453453423')

6. Триггеры PostgreSQL

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

max_code_lector()

General

Definition

Code

Options

Parameters

Security

SQL

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

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

- b. 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 - Trigger function

General

Definition

Code

Options

Parameters

Security

SQL

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

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

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

```
insert_exception()
General Definition Code Options Parameters Security SQL
1 BEGIN
2 RAISE EXCEPTION 'Вставка строк запрещена';
3 END;
```

- e. Проверьте выполнение команд транзакции при добавлении новой информации о преподавателях.

7. Транзакции

- a. Проверьте выполнение команд транзакции при добавлении новой информации об издательствах.

	Code_publish	Publish	City
▶	11	АСТ	Москва
	12	Союз писателей	Москва
	13	Питер	Санкт-Петербург
	14	Мир	Казань
	15	Китан	Уфа
*	NULL	NULL	NULL

```
1 • SELECT * FROM lab.publishing_house;
2
3 • SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
4 • start transaction;
5
6 • insert into publishing_house(Code_Publish, Publish, City)
7   values(16, 'Издательство 1', 'Краснодар');
8
9 • savepoint after_first_insert;
10
11 • insert into publishing_house(Code_Publish, Publish, City)
12   values(17, 'Издательство 2', 'Сызрань');
```

	Code_publish	Publish	City
▶	11	АСТ	Москва
	12	Союз писателей	Москва
	13	Питер	Санкт-Петербург
	14	Мир	Казань
	15	Китап	Уфа
	16	Издательство 1	Краснодар
	17	Издательство 2	Сызрань
•	NULL	NULL	NULL

14 •	<code>SELECT LAST_INSERT_ID();</code>
<	
Result Grid	Filter Rows: <input type="text"/>
	LAST_INSERT_ID()
▶	110

```

14 • SELECT LAST_INSERT_ID();
15 • insert into publishing_house(Code_Publish, Publish, City)
16 values(18, 'Издательство 3', 'Иркутск');
17

```

	Code_publish	Publish	City
▶	11	АСТ	Москва
	12	Союз писателей	Москва
	13	Питер	Санкт-Петербург
	14	Мир	Казань
	15	Китап	Уфа
	16	Издательство 1	Краснодар
	17	Издательство 2	Сызрань
	18	Издательство 3	Иркутск
▲	NULL	NULL	NULL

```

18 • rollback to savepoint after_first_insert;
19 • select * from publishing_house
20   where Publish in ('Издательство 1', 'Издательство 2', 'Издательство 3');

```

Result Grid			
Filter Rows: <input type="text"/>			
Edit:			
Export/Import:			
Wrap Cell Cont			
	Code_publish	Publish	City
▶	16	Издательство 1	Краснодар
•	NULL	NULL	NULL

```

22 • commit;
23 • SELECT * FROM publishing_house;

```

Result Grid			
Filter Rows: <input type="text"/>			
Edit:			
	Code_publish	Publish	City
▶	11	АСТ	Москва
	12	Союз писателей	Москва
	13	Питер	Санкт-Петербург
	14	Мир	Казань
	15	Китап	Уфа
	16	Издательство 1	Краснодар
•	NULL	NULL	NULL

8. Работа с пользователями

а. Администратор – обладает всеми правами

```

3 • CREATE USER 'lab_admin'@'localhost' IDENTIFIED BY 'admin_pass';
4
5 • GRANT ALL PRIVILEGES ON lab.* TO 'lab_admin'@'localhost';
6
7 • FLUSH PRIVILEGES;
8 • SHOW GRANTS FOR 'lab_admin'@'localhost';

```

Result Grid	
Filter Rows: <input type="text"/>	
Export:	
Wrap Cell Content:	
	Grants for lab_admin@localhost
▶	GRANT USAGE ON *.* TO 'lab_admin'@'localhost'
	GRANT ALL PRIVILEGES ON 'lab'.* TO 'lab_admin'@'localhost'

```
-- Удаление пользователя
DROP USER 'lab_admin'@'localhost';
```

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

```
2
3 • CREATE USER 'dispatcher'@'localhost' IDENTIFIED BY 'dispatcher_pass';
4
5 • GRANT SELECT, INSERT, UPDATE ON lab.books TO 'dispatcher'@'localhost';
6 • GRANT SELECT, INSERT, UPDATE ON lab.authors TO 'dispatcher'@'localhost';
7 • GRANT SELECT, INSERT, UPDATE ON lab.publishing_house TO 'dispatcher'@'localhost';
8 • GRANT SELECT, INSERT, UPDATE ON lab.deliveries TO 'dispatcher'@'localhost';
9
10 • FLUSH PRIVILEGES;
11 • SHOW GRANTS FOR 'dispatcher'@'localhost';
```

<

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Grants for dispatcher@localhost	
▶	GRANT USAGE ON *.* TO 'dispatcher'@'local...
	GRANT SELECT, INSERT, UPDATE ON 'lab'.'a...
	GRANT SELECT, INSERT, UPDATE ON 'lab'.'b...
	GRANT SELECT, INSERT, UPDATE ON 'lab'.'d...
	GRANT SELECT, INSERT, UPDATE ON 'lab'.'p...

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

```
3 • CREATE USER 'manager'@'localhost' IDENTIFIED BY 'manager_pass';
4 • GRANT SELECT, INSERT ON lab.deliveries TO 'manager'@'localhost';
5 • GRANT SELECT, INSERT ON lab.publishing_house TO 'manager'@'localhost';
6 • GRANT SELECT, INSERT ON lab.purchases TO 'manager'@'localhost';
7
8 • FLUSH PRIVILEGES;
9 • SHOW GRANTS FOR 'manager'@'localhost';
```

<

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Grants for manager@localhost	
▶	GRANT USAGE ON *.* TO 'manager'@'localhost'
	GRANT SELECT, INSERT ON 'lab'.'deliveries' TO 'manager'@'localhost'
	GRANT SELECT, INSERT ON 'lab'.'publishing_house' TO 'manager'@'localhost'
	GRANT SELECT, INSERT ON 'lab'.'purchases' TO 'manager'@'localhost'

- d. Поставщики – просматривают только свои поставки

```

1 • SELECT * FROM lab.publishing_house;
2
3 • CREATE USER 'sup1'@'localhost' IDENTIFIED BY '1_pass';
4 • CREATE USER 'sup2'@'localhost' IDENTIFIED BY '2_pass';
5
6 • CREATE VIEW s1_deliv AS
7   SELECT * FROM deliveries WHERE Name_company = 'ИП "МирСмешариков"';
8
9 • CREATE VIEW s2_deliv AS
10  SELECT * FROM deliveries WHERE Name_company = 'ООО "ЗемляКниг"';
11
12 • GRANT SELECT ON lab.deliveries TO 'sup1'@'localhost';
13 • GRANT SELECT ON lab.deliveries TO 'sup2'@'localhost';
14
15 • FLUSH PRIVILEGES;
16 • SHOW GRANTS FOR 'sup1'@'localhost';
17 • SHOW GRANTS FOR 'sup2'@'localhost';

```

<

Result Grid



Filter Rows:

Export:



Wrap Cell Content:

	Grants for sup1@localhost
▶	GRANT USAGE ON *.* TO `sup1`@`localhost`
	GRANT SELECT ON `lab`.`deliveries` TO `sup1`...