

Представления

MySQL:

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

```
create or replace view supply_info as
select
    p.code_purchase,
    b.title_book,
    p.date_order,
    d.name_delivery,
    p.cost,
    p.amount
from
    purchases p
join
    books b on p.code_book = b.code_book
join
    deliveries d on p.code_delivery = d.code_delivery;
Вызываем представление
```

Результат:

code_purchase	title_book	date_order	name_delivery	cost	amount
1	Основы программирования на Python	2023-01-15 10:00:00	Иванов И.И.	500.00	2
2	Введение в базы данных	2023-02-20 14:30:00	Петров П.П.	450.00	5
3	Алгоритмы и структуры данных	2023-03-10 09:00:00	Сидоров С.С.	600.00	1
4	Современные веб-технологии	2023-04-05 16:00:00	Смирнов А.С.	550.00	3
5	Машинное обучение для начинающих	2023-05-12 11:00:00	Кузнецова А.В.	700.00	2
6	Разработка мобильных приложений	2023-06-25 13:00:00	Попов Д.А.	650.00	4
7	Тестирование программного обеспечения	2023-07-08 10:30:00	Васильев С.И.	800.00	1

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

```
create or replace view moscow_publishers as
select
    code_publish',
    publish,
    city'
from
    publishing_house
where
    city = 'москва';
```

Вызываем представление

Результат:

code_publish	publish	city
2	Эксмо	Москва
3	АСТ	Москва
4	Манн, Иванов и Фербер	Москва
5	Альпина Паблишер	Москва
7	ДМК Пресс	Москва
9	Вильямс	Москва

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

```
create or replace view books_inventory as
select
    b.code_book as 'код книги',
    b.title_book as 'наименование книги',
    a.name_author as 'автор',
    sum(p.amount) as 'количество на складе',
    max(p.cost) as 'максимальная стоимость'
from
    books b
join
    authors a on b.code_author = a.code_author
join
    purchases p on b.code_book = p.code_book
group by
    b.code_book, b.title_book, a.name_author;
```

Вызываем представление

Результат:

code_book	title_book	name_author	sum(p.amount)	max(p.cost)
1	Основы программирования на Python	Иванов Иван Иванович	2	500.00
2	Введение в базы данных	Петров Петр Петрович	5	450.00
3	Алгоритмы и структуры данных	Сидоров Сидор Сидорович	1	600.00
4	Современные веб-технологии	Смирнов Алексей Сергеевич	3	550.00
5	Машинное обучение для начинающих	Кузнецова Анна Владимировна	2	700.00
6	Разработка мобильных приложений	Попов Дмитрий Александрович	4	650.00
7	Тестирование программного обеспечения	Васильев Сергей Иванович	1	800.00

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

```
create or replace view top_5_books_by_quantity as
```

```

select
    *
from
    books_inventory
order by
    'количество на складе' desc
limit 5;

```

Вызываем представление

Результат:

code_book	title_book	name_author	sum(p.amount)	max(p.cost)
1	Основы программирования на Python	Иванов Иван Иванович	2	500.00
2	Введение в базы данных	Петров Петр Петрович	5	450.00
3	Алгоритмы и структуры данных	Сидоров Сидор Сидорович	1	600.00
4	Современные веб-технологии	Смирнов Алексей Сергеевич	3	550.00
5	Машинное обучение для начинающих	Кузнецова Анна Владимировна	2	700.00

Хранимые процедуры

MySQL:

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).

DELIMITER //

```

create procedure books.MaxAmount()
begin
    select p.*, b.title_book
from purchases p
join books b on p.code_book = b.code_book
order by p.cost * p.amount desc
limit 1;
end //

```

DELIMITER ;

Объединяем таблицы «books» и «purchases» по полю «code_book», определяем стоимость каждой покупки и вычисляем максимальную общую сумму.

Вызываем хранимую процедуру

call books.MaxAmount();

Результат:

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount	title_book
10	10	2023-10-10 12:00:00	10	Опт	850.00	5	Операционные системы

Сосчитать количество книг определенного автора (ФИО автора является

входным параметром).

delimiter //

```
create procedure books.CountBooksAuthors(in name_author
VARCHAR(255))
```

```
begin
```

```
select count(b.code_author) from books b
```

```
join authors a on a.code_author = b.code_author
```

```
where a.name_author = name_author;
```

```
end //
```

```
delimiter ;
```

Вводим имя автора в качестве входного параметра, затем объединяем таблицы «authors» и «books» и определяем количество произведений конкретного автора в базе данных.

Вызываем хранимую процедуру

```
call books.CountBooksAuthors('Иванов Иван Иванович');
```

Результат:

	count(b.code_author)
►	2

Определить адрес определенного поставщика

(Наименование

поставщика является входным параметром, адрес поставщика – выходным параметром).

delimiter //

```
create procedure books.Address(in name_company VARCHAR(255), out
address VARCHAR(255))
```

```
begin
```

```
select d.address into address from deliveries d
```

```
where d.name_company = name_company;
```

```
end //
```

```
delimiter ;
```

Вводим входные данные: наименование поставщика, адрес поставщика. На основе этих данных определяем конкретного поставщика из базы данных

Вызываем хранимую процедуру

```
set @sddress = '';
```

```
call books.Address('ООО Книжный мир', @address);
select @address;
```

Результат:

@address
г. Москва, ул. Ленина, д. 1

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

delimiter //

```
create procedure insertbook(
    in p_title varchar(255),
    in p_author_code int,
    in p_pages int,
    in p_publish_code int
)
begin
    declare new_code int;
    select ifnull(max(code_book), 0) + 1 into new_code from books;

    insert into books(code_book, title_book, code_author, pages,
code_publish)
        values (new_code, p_title, p_author_code, p_pages,
p_publish_code);
end //
delimiter ;
```

Создаём хранимую процедуру insertbook для добавления новой книги в таблицу books. Вводим входные параметры: название книги, код автора, количество книг, код издательства.

Вызываем хранимую процедуру

```
call insertbook('Новая книга', 1, 300, 1);
```

Результат:

Code_book	Title_book	Code_author	Pages	Code_publish
1	Основы программирования на Python	1	320	2
2	Введение в базы данных	2	450	1
3	Алгоритмы и структуры данных	3	500	3
4	Современные веб-технологии	4	280	4
5	Машинное обучение для начинающих	5	350	5
6	Разработка мобильных приложений	6	400	6
7	Тестирование программного обеспечения	7	300	7
8	Сетевые технологии	8	420	8
9	Криптография и защита информации	9	380	9
10	Операционные системы	10	480	10
11	Новая книга	1	300	1

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».

delimiter //

create procedure minmax()

begin

declare min_cost decimal(10,2);

declare max_cost decimal(10,2);

select min(cost * amount), max(cost * amount)

into min_cost, max_cost

from purchases;

select

p.*,

b.title_book,

case

when (p.cost * p.amount) = min_cost then 'минимальная стоимость'

when (p.cost * p.amount) = max_cost then
'максимальная стоимость'

else 'средняя стоимость'

end as cost_status

from purchases p

join books b on p.code_book = b.code_book;

end //

delimiter ;

Процедура анализирует закупки книг, определяет минимальную и максимальную общую стоимость и помечает каждую запись в результате

Вызываем хранимую процедуру

call minmax();

Результат:

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount	costCategory
1	1	2023-01-15 10:00:00	1	Розница	500.00	2	средняя стоимость
2	2	2023-02-20 14:30:00	2	Опт	450.00	5	средняя стоимость
3	3	2023-03-10 09:00:00	3	Розница	600.00	1	минимальная стоимость
4	4	2023-04-05 16:00:00	4	Опт	550.00	3	средняя стоимость
5	5	2023-05-12 11:00:00	5	Розница	700.00	2	средняя стоимость
6	6	2023-06-25 13:00:00	6	Опт	650.00	4	средняя стоимость
7	7	2023-07-08 10:30:00	7	Розница	800.00	1	средняя стоимость
8	8	2023-08-18 15:00:00	8	Опт	750.00	3	средняя стоимость
9	9	2023-09-01 08:00:00	9	Розница	900.00	2	средняя стоимость
10	10	2023-10-10 12:00:00	10	Опт	850.00	5	максимальная стоимость

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'.

delimiter //

create procedure filldeliveries()

begin

declare delivery_count int;

declare new_code int;

declare initial_count int;

select count(*) into initial_count from deliveries;

set delivery_count = initial_count;

while delivery_count < 10 do

select ifnull(max(code_delivery), 0) + 1 into new_code from deliveries;

insert into deliveries(

code_delivery, name_delivery, name_company,

address, phone, inn

) values (

new_code, 'не известен', null, null, null, null

);

set delivery_count = delivery_count + 1;

end while;

select concat('Добавлено ', delivery_count - initial_count, ' записей') as result;

end //

delimiter;

Определяем максимальный код поставщика, выполняем итерацию, пока код не достигнет значения 10.

Затем вставляем данные и увеличиваем максимальный код и количество.

Вызываем хранимую процедуру

call filldeliveries();

Результат:

5	не известен	NULL	NULL	NULL	NULL
6	не известен	NULL	NULL	NULL	NULL
7	не известен	NULL	NULL	NULL	NULL
8	не известен	NULL	NULL	NULL	NULL
9	не известен	NULL	NULL	NULL	NULL
10	не известен	NULL	NULL	NULL	NULL
*	NULL	NULL	NULL	NULL	NULL

PostgreSQL:

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 or replace function topstudents()
returns table(surname character varying, name character varying)
language plpgsql
as $$
begin
    return query
    select s.surname, s.name_
    from students s
    where s.code_stud in ( select code_stud from progress
                          group by code_stud
                          order by avg(estimate) desc
                        );
end;
$$;
```

Для анализа и оценки результатов мы используем таблицу с данными о прогрессе. Для каждого студента мы вычисляем среднее значение оценки (AVG(p.Estimate)). Затем мы находим максимальное значение из этих средних (MAX(AvgEstimate)). После этого мы выбираем записи, в которых среднее значение оценки студента равно максимальному.

Вызываем хранимую процедуру:

```
select * from topstudents();
```

Результат:

	surname character varying 	name character varying 
1	Иванов	Иван
2	Петров	Петр
3	Сидоров	Сидор
4	Кузнецов	Кузьма
5	Смирнов	Смир
6	Васильев	Василий
7	Новиков	Новик
8	Федоров	Федор
9	Морозов	Петр
10	Иванов	Петр

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

```
create or replace function studentsestimate(student_name character  
varying, student_lastname character varying)
```

```
returns numeric
```

```
as $$
```

```
begin
```

```
return(
```

```
    select p.estimate
```

```
        from progress p
```

```
        where p.code_stud in (
```

```
            select code_stud
```

```
            from students
```

```
            where name_ = student_name and lastname =
```

```
student_lastname));
```

```
end;
```

```
$$ language plpgsql;
```

Определяем средний балл для каждого учащегося и выводим среднее значение оценки для определённого студента из базы данных.

Вызываем хранимую процедуру

```
select * from studentsestimate('Иван', 'Иванович');
```

Результат:

	studentsestimate numeric
1	5

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

```
create or replace function Speciality(in surname character varying, in  
name_stud character varying, in lastname character varying, out  
speciality character varying, out course integer)
```

```
returns record as $$
```

```
begin
```

```
    select g.name_speciality, g.num_course  
    into speciality, course  
    from groups_ g  
    where g.code_group in (  
        select s.code_group  
        from students s  
        where s.surname = Speciality.surname and s.name_ =  
Speciality.name_stud and s.lastname = Speciality.lastname);  
end;
```

```
$$ language plpgsql;
```

Вызываем хранимую процедуру

```
select * from Speciality('Иванов','Иван', 'Иванович');
```

Результат:

	speciality character varying	course integer
1	Программирование в компьютерных системах	2

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

```
create or replace function InsertStudents()
```

```
returns void as $$
```

```
declare
```

```
next_code_stud int;
```

```

begin
    select coalesce(max(code_stud::int),0) + 1 into next_code_stud
from students;
    insert into students (code_stud) values (next_code_stud);
    perform count(*) from students;
end;
$$ language plpgsql;

```

Вводим исходные данные, определяем максимальное значение кода студента, увеличиваем его на единицу

Вызываем хранимую процедуру

```

select * from InsertStudents('Рогожкина', 'Алеся', 'Станиславовна', 1,
'2005-02-16', '79638995709');

```

Результат:

code_stud [PK] character (10)	surname character varying (50)	name_ character varying (50)	lastname character varying (50)	code_group integer	birthday date	phone numeric (15)
1	Иванов	Иван	Иванович	1	1995-05-15	220001
2	Петров	Петр	Петрович	2	1996-06-16	220002
3	Сидоров	Сидор	Сидорович	3	1997-07-17	220003
4	Кузнецов	Кузьма	Кузьмич	4	1998-08-18	220004
5	Смирнов	Смир	Смирович	5	1999-09-19	220005
6	Васильев	Василий	Васильевич	6	2000-10-20	220006
7	Новиков	Новик	Новикович	7	2001-11-21	220007
8	Федоров	Федор	Федорович	8	2002-12-22	220008
9	Морозов	Петр	Петрович	8	2003-01-23	220009
10	Иванов	Петр	Петрович	10	2004-02-24	220010
11	[null]	[null]	[null]	[null]	[null]	[null]
12	Рогожкина	Алеся	Станиславовна	1	2005-02-16	79638995709

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 AverageAge()
returns table(student_surname character varying, student_name_stud
character varying, student_lastname character varying, age_stud text)
as $$
declare
    average_age int;
begin

```

```

select avg(extract(year from age(current_date, birthday))) into
average_age
from students;
return query
select name_, surname, lastname,
case
when extract(year from age(current_date, birthday)) >
average_age then 'Вы старше среднего возраста всех студентов'
when extract(year from age(current_date, birthday)) <
average_age then 'Ваш возраст меньше среднего возраста всех
студентов'
else 'Ваш возраст равен среднему возрасту всех студентов'
end as age_stud
from students;
end;
$$ language plpgsql;

```

Устанавливаем возраст на основе даты рождения, определяем средний возраст всех учащихся и пишем, кто старше или младше среднего возраста, а также чей возраст соответствует среднему.

Вызываем хранимую процедуру

```
select * from AverageAge();
```

Результат:

	student_surname character varying	student_name_stud character varying	student_lastname character varying	age_stud text
1	Иван	Иванов	Иванович	Вы старше среднего возраста всех студентов
2	Петр	Петров	Петрович	Вы старше среднего возраста всех студентов
3	Сидор	Сидоров	Сидорович	Вы старше среднего возраста всех студентов
4	Кузьма	Кузнецов	Кузьмич	Вы старше среднего возраста всех студентов
5	Смир	Смирнов	Смирович	Вы старше среднего возраста всех студентов
6	Василий	Васильев	Васильевич	Ваш возраст равен среднему возрасту всех студентов
7	Новик	Новиков	Новикович	Ваш возраст меньше среднего возраста всех студентов
8	Федор	Федоров	Федорович	Ваш возраст меньше среднего возраста всех студентов
9	Петр	Морозов	Петрович	Ваш возраст меньше среднего возраста всех студентов
10	Петр	Иванов	Петрович	Ваш возраст меньше среднего возраста всех студентов
11	[null]	[null]	[null]	Ваш возраст равен среднему возрасту всех студентов
12	Алеся	Рогожкина	Станиславовна	Ваш возраст меньше среднего возраста всех студентов

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 $$
declare subjects_count int;
next_code_subject int;
begin
    select count(*) into subjects_count from subjects;
    select coalesce(max(code_subject), 0) + 1 into next_code_subject
from subjects;
    while subjects_count < 10 loop
        insert into subjects (code_subject, name_subject) values
(next_code_subject, 'не известен');
        select count(*) into subjects_count from subjects;
    end loop;
end;
$$ language plpgsql;

```

Определяем максимальный код предмета и проходим цикл до тех пор, пока не достигнем количества равного 10. Затем добавляем новые записи в таблицу дисциплин ставя значение 'не известно'

Вызываем хранимую процедуру

```
select * from FillSubjects();
```

Результат:

	code_subject [PK] integer	name_subject character varying (100)	count_hours integer
1	1	Математический анализ	120
2	2	Физика	100
3	3	Программирование	150
4	4	Биология	90
5	5	Химия	80
6	6	Экономика	110
7	7	История	70
8	8	Литература	60
9	9	География	50
10	10	не известен	[null]

Триггеры

MySQL:

Создайте триггер, запускаемый при занесении новой строки в таблицу

Авторы. Триггер должен увеличивать счетчик числа добавленных строк.

delimiter //

create trigger after_author_insert

after insert on authors

for each row

begin

insert into row_counters (table_name, counter)

values ('authors', 1)

on duplicate key update counter = counter + 1;

end //

delimiter ;

Вызываем триггер

insert into authors (code_author, name_author) values ('новый автор', '1900-01-01');

Результат:

Code_author	Name_author	Birthday
1	Иванов Иван Иванович	1980-05-15 00:00:00
2	Петров Петр Петрович	1975-10-20 00:00:00
3	Сидоров Сидор Сидорович	1985-03-01 00:00:00
4	Смирнов Алексей Сергеевич	1990-12-25 00:00:00
5	Кузнецова Анна Владимировна	1982-07-08 00:00:00
6	Попов Дмитрий Александрович	1978-09-12 00:00:00
7	Васильев Сергей Иванович	1988-02-18 00:00:00
8	Федоров Андрей Петрович	1972-04-05 00:00:00
9	Соколова Елена Михайловна	1983-11-30 00:00:00
10	Михайлов Михаил Юрьевич	1987-06-22 00:00:00
11	Иванов Петр Иванович	1981-05-15 00:00:00
12	Иванов Иван Иванович	1980-05-15 00:00:00
13	Иванов Иван Иванович	1980-05-15 00:00:00
14	Иванов Иван Иванович	1980-05-15 00:00:00
15	новый автор	1900-01-01 00:00:00

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 procedure counter_book()

begin

 update authors a

 join (

 select code_author, count(title_book) as book_count

 from books

 group by code_author

) b on a.code_author = b.code_author

 set a.count_books = b.book_count;

end //

delimiter ;

Вызываем триггер

insert into books values(13, 'Новая книга', 1, 200, 2);

Результат:

Code_author	Name_author	Birthday	Count_books
2	Петров Петр Петрович	1975-10-20 00:00:00	0
3	Сидоров Сидор Сидорович	1985-03-01 00:00:00	1
4	Смирнов Алексей Сергеевич	1990-12-25 00:00:00	1
5	Кузнецова Анна Владимировна	1982-07-08 00:00:00	2
6	Попов Дмитрий Александрович	1978-09-12 00:00:00	1
7	Васильев Сергей Иванович	1988-02-18 00:00:00	1
8	Федоров Андрей Петрович	1972-04-05 00:00:00	1
9	Соколова Елена Михайловна	1983-11-30 00:00:00	1
10	Михайлов Михаил Юрьевич	1987-06-22 00:00:00	1
11	Иванов Петр Иванович	1981-05-15 00:00:00	1
12	Иванов Иван Иванович	1980-05-15 00:00:00	1
13	Новый автор	1901-01-01 00:00:00	1

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='root'@'localhost' trigger `purchases_before_insert`
before insert on `purchases` for each row
begin
```

```
    if new.amount < 10 then
```

```
        set new.cost = new.cost * 1.2;
```

```
    end if;
```

```
end //
```

delimiter ;

Если количество записей меньше 10, то стоимость книги увеличивается на 20%

Вызываем триггер

```
insert into purchases values (2, '2023-08-20', 1, 'Опт', 500.00, 5);
```

Результат:

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount
1	1	2023-01-15 10:00:00	1	Розница	500.00	2
2	2	2023-02-20 14:30:00	2	Опт	450.00	5
3	3	2023-03-10 09:00:00	3	Розница	600.00	1
4	4	2023-04-05 16:00:00	4	Опт	550.00	3
5	5	2023-05-12 11:00:00	5	Розница	700.00	2
6	6	2023-06-25 13:00:00	6	Опт	650.00	4
7	7	2023-07-08 10:30:00	7	Розница	800.00	1
8	2	2023-08-20	1	Опт	600	5

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

delimiter //


```

create definer='root'@'localhost' trigger `deliveries_before_insert`
before insert on `deliveries` for each row
begin
    signal sqlstate '45000'
    set message_text = 'вставка строк запрещена';
end //
delimiter ;

```

Добавляем новую запись, но выходит ошибка о запрете вставки новых строк

Вызываем триггер

```

(10, 'Экспресс-доставка', 'ООО Логистик Групп', 'пр. Ленина 42',
79161234567, 98765432109);

```

Результат:

108 22:26:29 insert into deliveries values(11,'Быстрый', 'ЗАО Экспресс', 'ул. Достоевского 15', 79870440560,... Error Code: 1644. вставка строк запрещена 0.046 sec

Проверьте выполнение команд транзакции при добавлении новой информации об издательствах

```

start transaction;
select * from publishing_house;
savepoint before_test_insert;
insert into publishing_house (publish, city)
values ('новое издательство', 'москва');
rollback to savepoint before_test_insert;
select * from publishing_house;

```

Откатываем изменения до точки сохранения

Code_publish	Publish	City
4	Манн, Иванов и Фербер	Москва
5	Альпина Пабlishер	Москва
6	БХВ-Петербург	Санкт-Петербург
7	ДМК Пресс	Москва
8	Наука и Техника	Санкт-Петербург
9	Вильямс	Москва
10	Символ-Плюс	Санкт-Петербург
11	новое издательство	москва

Code_publish	Publish	City
1	Питер	Санкт-Петербург
2	Эксмо	Москва
3	АСТ	Москва
4	Манн, Иванов и Фербер	Москва
5	Альпина Пабlishер	Москва
6	БХВ-Петербург	Санкт-Петербург
7	ДМК Пресс	Москва
8	Наука и Техника	Санкт-Петербург
9	Вильямс	Москва
10	Символ-Плюс	Санкт-Петербург

PostgreSql:

Создайте триггер, запускаемый при занесении новой строки в таблицу

Преподаватели. Триггер должен увеличивать счетчик числа добавленных строк.

begin

```
new.code_lector = (  
    select (max(code_lector), 0) + 1  
    from lectors  
);  
return new;
```

end;

insert into lectors(Name_lector, Science, Post, Date_)

values('Иванов Иван Иванович', 'д.э.н', 'профессор', '2005-11-11');

<i>code_lector</i> [PK] integer	<i>name_lector</i> character varying (100)	<i>science</i> character varying (50)	<i>post</i> character varying (50)	<i>date_</i> date
6	Федоров Федор Федорович	д.э.н.	Профессор	2007-10-19
8	Лебедев Лебедь Лебедевич	д.т.н.	Профессор	2009-12-21
10	Сидоров Сидор Сидорович	к.э.н.	Преподаватель	2002-05-14
3	Кузнецов Кузьма Кузьмич	д.э.н.	Профессор	2003-06-15
7	Морозов Мороз Морозович	к.т.н.	Доцент	2008-11-20
9	Петров Савелий Яковлевич	к.т.н.	Доцент	2003-05-05
12	Иванов Иван Иванович	д.э.н	профессор	2005-11-11
11	Сергеев Сергей Сергеевич	к.т.н.	Доцент	2023-01-01

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)

egin

call insert_st(new.Code_stud);

return new;

end;

Вызываем хранимую процедуру в триггере

begin

if not exists(select 1 from students where Code_stud = code_stud1)

```

then
insert into students
values(Code_stud1, 'Иванов', 'Павел', 'Сергеевич', 1, '2006-09-
13', 79111234566,0);
end if;end;
insert into progress
values('20', 7, 4, '2023-06-20', 5, 19);

```

Добавили нового студента

20	7		4		2023-06-20		5		19	
20	Иванов	...	Павел	...	Сергеевич	...	1	2006-09-13	79111234566	5.0000000000000000

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

```

egin
raise exception 'Вставка строк запрещена' using errcode =
'45000';
end;
insert into groupss
values(11,'24Э-2',3,'Экономист');

```

Вставляем новую строку, но выводится сообщение

ERROR: Вставка строк запрещена

CONTEXT: PL/pgSQL function exception_f() line 2 at RAISE

Проверьте выполнение команд транзакции при добавлении новой информации о преподавателях.

```

start transaction;
select * from lectors;
savepoint lectors_;
insert into lectors
values((select coalesce(max(Code_lector), 0) + 1 from lectors),
'Сергеев Сергей Сергеевич', 'к.т.н', 'Доцент', '2023- 01-01');
rollback to savepoint lectors_;

```

Выполняем выборку всех преподавателей, сохраняем изменения и вставляем новую запись

code_lector [PK] integer	name_lector character varying (100)	science character varying (50)	post character varying (50)	date_ date
1	Иванов Иван Иванович	к.т.н.	Доцент	2000-03-12
2	Петров Петр Петрович	д.т.н.	Профессор	2001-04-13
4	Смирнов Смир Смирович	к.т.н.	Доцент	2004-07-16
5	Васильев Василий Васильевич	д.т.н.	Профессор	2005-08-17
6	Федоров Федор Федорович	д.э.н.	Профессор	2007-10-19
8	Лебедев Лебедь Лебедевич	д.т.н.	Профессор	2009-12-21
10	Сидоров Сидор Сидорович	к.э.н.	Преподаватель	2002-05-14
3	Кузнецов Кузьма Кузьмич	д.э.н.	Профессор	2003-06-15
7	Морозов Мороз Морозович	к.т.н.	Доцент	2008-11-20
9	Петров Савелий Яковлевич	к.т.н.	Доцент	2003-05-05
11	Сергеев Сергей Сергеевич	к.т.н.	Доцент	2023-01-01
12	Иванов Иван Иванович	д.э.н.	профессор	2005-11-11

И откатываем до точки сохранения

code_lector [PK] integer	name_lector character varying (100)	science character varying (50)	post character varying (50)	date_ date
1	Иванов Иван Иванович	к.т.н.	Доцент	2000-03-12
2	Петров Петр Петрович	д.т.н.	Профессор	2001-04-13
4	Смирнов Смир Смирович	к.т.н.	Доцент	2004-07-16
5	Васильев Василий Васильевич	д.т.н.	Профессор	2005-08-17
6	Федоров Федор Федорович	д.э.н.	Профессор	2007-10-19
8	Лебедев Лебедь Лебедевич	д.т.н.	Профессор	2009-12-21
10	Сидоров Сидор Сидорович	к.э.н.	Преподаватель	2002-05-14
3	Кузнецов Кузьма Кузьмич	д.э.н.	Профессор	2003-06-15
7	Морозов Мороз Морозович	к.т.н.	Доцент	2008-11-20
9	Петров Савелий Яковлевич	к.т.н.	Доцент	2003-05-05
11	Сергеев Сергей Сергеевич	к.т.н.	Доцент	2023-01-01

Работа с пользователями

Роли:

Администратор имеет полный доступ

Диспетчер имеет право просматривать, заполнять и редактировать справочники

Менеджер имеет право оформлять поставки, а также добавлять и просматривать новую информацию во всех справочниках

```
CREATE USER 'dbadmin'@'localhost' IDENTIFIED BY '12345';
GRANT ALL PRIVILEGES ON books.* TO 'dbadmin'@'localhost';
SHOW GRANTS FOR 'dbadmin'@'localhost';
```

▶	GRANT USAGE ON *.* TO 'dbadmin'@'localhost'
	GRANT ALL PRIVILEGES ON 'books'.* TO 'dbadmin'@'localhost'

```
CREATE USER 'dbdispatcher'@'localhost' IDENTIFIED BY 'pass';
```

```
GRANT INSERT, SELECT, UPDATE ON books.authors TO
'dbdispetcher'@'localhost';
GRANT INSERT, SELECT, UPDATE ON books.books TO
'dbdispetcher'@'localhost';
GRANT INSERT, SELECT, UPDATE ON books.publishing_house TO
'dbdispetcher'@'localhost';
GRANT INSERT, SELECT, UPDATE ON books.deliveries TO
'dbdispetcher'@'localhost';
SHOW GRANTS FOR 'dbdispetcher'@'localhost';
```

GRANT USAGE ON *.* TO 'dbdispetcher'@'localhost'
GRANT SELECT, INSERT, UPDATE ON 'books`.`authors` TO 'dbdispetcher'@'localhost'
GRANT SELECT, INSERT, UPDATE ON 'books`.`books` TO 'dbdispetcher'@'localhost'
GRANT SELECT, INSERT, UPDATE ON 'books`.`deliveries` TO 'dbdispetcher'@'localhost'
GRANT SELECT, INSERT, UPDATE ON 'books`.`publishing_house` TO 'dbdispetcher'@'localhost'

```
CREATE USER 'dbmanager'@'localhost' IDENTIFIED BY '0987';
GRANT INSERT, SELECT ON books.authors TO 'dbmanager'@'localhost';
GRANT INSERT, SELECT ON books.books TO 'dbmanager'@'localhost';
GRANT INSERT, SELECT ON books.publishing_house TO
'dbmanager'@'localhost';
GRANT INSERT, SELECT ON books.deliveries TO
'dbmanager'@'localhost';GRANT INSERT, SELECT ON books.purchases TO
'dbmanager'@'localhost';
SHOW GRANTS FOR 'dbmanager'@'localhost';
```

GRANT USAGE ON *.* TO 'dbmanager'@'localhost'
GRANT SELECT, INSERT ON 'books`.`authors` TO 'dbmanager'@'localhost'
GRANT SELECT, INSERT ON 'books`.`books` TO 'dbmanager'@'localhost'
GRANT SELECT, INSERT ON 'books`.`deliveries` TO 'dbmanager'@'localhost'
GRANT SELECT, INSERT ON 'books`.`publishing_house` TO 'dbmanager'@'localhost'
GRANT SELECT, INSERT ON 'books`.`purchases` TO 'dbmanager'@'localhost'

Поставщики имеют право просматривать только свои поставки

```
CREATE VIEW delivery_1 AS
SELECT p.*, d.Name_delivery
FROM Purchases p
JOIN Deliveries d ON p.Code_delivery = d.Code_delivery
WHERE p.Code_delivery = 1;
CREATE VIEW delivery_2 AS
SELECT p.*, d.Name_delivery
FROM Purchases p
JOIN Deliveries d ON p.Code_delivery = d.Code_delivery
WHERE p.Code_delivery = 2;
CREATE VIEW delivery_3 AS
SELECT p.*, d.Name_delivery
```

```

FROM Purchases p
JOIN Deliveries d ON p.Code_delivery = d.Code_delivery
WHERE p.Code_delivery = 3;
CREATE VIEW delivery_4 AS
SELECT p.*, d.Name_delivery
FROM Purchases p JOIN Deliveries d ON p.Code_delivery =
d.Code_delivery
WHERE p.Code_delivery = 4;
CREATE USER 'delivery_1'@'localhost' IDENTIFIED BY 'qwe';
CREATE USER 'delivery_2'@'localhost' IDENTIFIED BY 'asd';
CREATE USER 'delivery_3'@'localhost' IDENTIFIED BY 'zxc';
CREATE USER 'delivery_4'@'localhost' IDENTIFIED BY 'ewq';
GRANT SELECT ON books.delivery_1 TO 'delivery_1'@'localhost';
GRANT SELECT ON books.delivery_2 TO 'delivery_2'@'localhost';
GRANT SELECT ON books.delivery_3 TO 'delivery_3'@'localhost';
GRANT SELECT ON books.delivery_4 TO 'delivery_4'@'localhost';
SHOW GRANTS FOR 'delivery_1'@'localhost';

```

Code_purchase	Date_order	code_delivery	Type_purchase	Cost	Amount	Code_book	Name_delivery
1	2023-01-1...	1	0	350	100	1	Иванов И.И.
4	2023-04-0...	1	1	600	50	2	Иванов И.И.
7	2023-07-0...	1	1	700	15	1	Иванов И.И.