

ХРАНИМЫЕ ПРОЦЕДУРЫ MySQL

```
use db_books;  
delimiter //
```

```
create procedure infobook()  
begin  
select b.title_book, p.*, (p.Cost * p.Amount) as total_cost from  
Purchases p  
join books b on b.code_book = p.Code_book  
where p.Cost * p.Amount = (select max(p.Cost * p.Amount) from  
Purchases p);  
end//
```

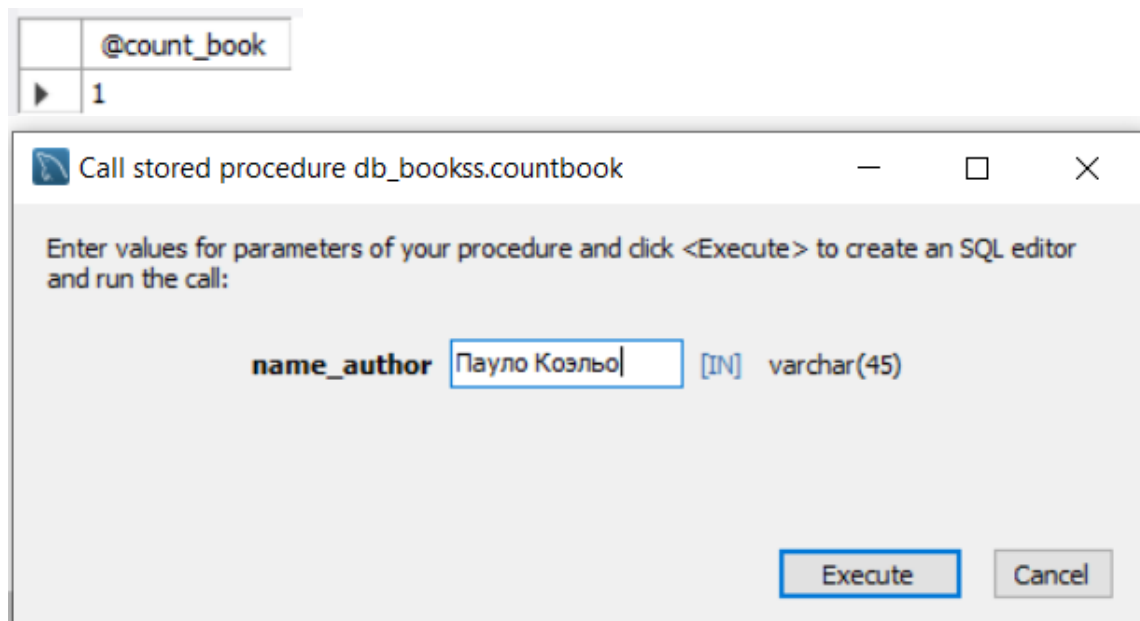
Находим самую дорогую покупку книг, а именно соединяем таблицы Purchases и Books, вычисляем стоимость каждой покупки (цена на количество), находит максимальную общую стоимость, показывает название книги и данные о покупке, общую стоимость

	title_book	Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount	total_cost
►	Ловец огней на звездном поле	6	6	2023-12-19	3	опт	420	20	8400

```
create procedure countbook(  
in name_author varchar(45),  
out count_book int  
)  
begin  
select count(b.code_author) into count_book  
from Books b  
join Authors a on b.Code_author = a.Code_author  
where a.name_author = name_author;  
end//
```

Подсчитывает количество книг конкретного автора в базе данных.

Передаем имя автора, в таблице Books ищет автора с тем же номером, для этого соединяет таблицы Books и Authors, считает его кол-во книг.



```
create procedure adressdelivery(  
    in Name_company varchar(20),  
    out Adress varchar(50)  
)  
begin  
    select d.adress into Adress  
    from Deliveries d  
    where d.Name_company = Name_company;  
end//
```

Задаем входной параметр Название поставщика, выходной Адрес поставщика, находим конкретного поставщика из бд.

Call stored procedure db_bookss.adressdelivery

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

Name_company [IN] varchar(20)

Execute **Cancel**

Result Grid

	@Adress
▶	НЕТ СВЕДЕНИЙ

```

create procedure insertbook(
  in Title_book varchar(45),
  in Code_author int,
  in Pages int,
  in Code_publish int
)
begin
insert into Books(Title_book, Code_author, Pages, Code_publish)
values(Title_book, Code_author, Pages, Code_publish);
select * from books;
end//

```

Входные параметры: название, код автора, кол-во страниц, код поставщика, добавляет новую книгу с этими данными в бд

Call stored procedure db_bookss.insertbook

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

Title_book	<input type="text" value="Звезда"/>	[IN]	varchar(45)
Code_author	<input type="text" value="1"/>	[IN]	int
Pages	<input type="text" value="123"/>	[IN]	int
Code_publish	<input type="text" value="2"/>	[IN]	int

57	не известен gnyg)			
58	Звезда	1	123	2

```

create procedure Supplies()
begin
select p.*,
case
  when (p.Cost * p.Amount) = (select min( p.Cost * p.Amount) from
Purchases p) then
    'Минимальная стоимость'
  when p.Cost * p.Amount = (select max(p.Cost * p.Amount) from
Purchases p) then
    'Максимальная стоимость'
  when p.Cost * p.Amount is null then ''
  else 'Средняя стоимость'
end as total_cost
from Purchases p
order by p.Cost* p.Amount desc;
end//

```

Выводим все данные о покупке, для каждой покупки считает общую стоимость. Если стоимость равняется минимальной, то добавляется “Минимальная стоимость”, если максимальная - “Максимальная стоимость”, если Null- пустая строка, иначе - “Средняя стоимость”. Сортирует результат по убыванию.

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount	total_cost
6	6	2023-12-19	3	опт	420	20	Максимальная стоимость
10	20	2025-12-23	1	опт	150	25	Средняя стоимость
11	20	2025-12-23	1	опт	150	25	Средняя стоимость
12	20	2025-12-23	1	опт	150	25	Средняя стоимость
13	20	2025-12-23	1	опт	150	25	Средняя стоимость
1	1	2023-03-08	1	опт	310	10	Средняя стоимость
4	4	2023-03-15	2	розница	415	1	Средняя стоимость
3	3	2023-03-12	3	розница	412	1	Средняя стоимость
5	5	2023-03-17	1	розница	408	1	Средняя стоимость
14	20	2025-12-23	1	опт	180	2	Средняя стоимость
2	2	2023-03-10	1	розница	5	1	Минимальная стоимость
7	NULL	NULL	NULL	опт	NULL	NULL	

```

create procedure AddAuthors()
begin
declare count_author int;
select count(*) into count_author from Authors;
while count_author < 10 do
    insert into Authors(Name_author) values('не известен');
    set count_author = count_author + 1;
end while;
end//

```

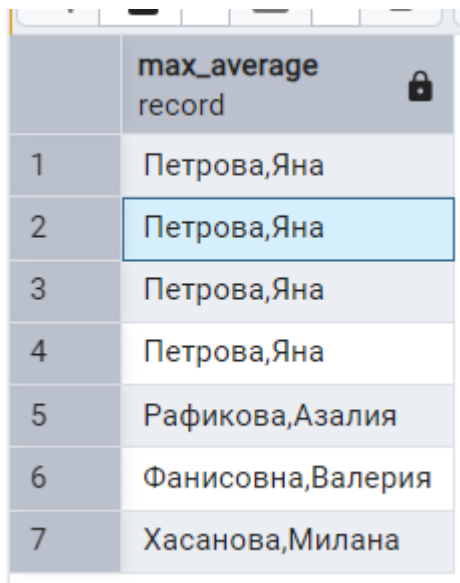
Добавляет в бд автора с именем “не известен”, пока общее кол-во авторов не достигнет 10

Code_author	Name_author	Birthday	Count_books
1	Чарльз Мартин Лефлер	1861-01-30	4
2	Агата Мэри Кларисса Миллер	1890-09-15	2
3	Джордж Самюэль Клейсон	1874-11-07	1
4	Пауло Козьмо	1947-08-24	1
5	Франк Тилье	1973-10-15	1
6	Антуан де Сент-Экзюпери	1990-06-29	0
7	не известен	NULL	0
8	не известен	NULL	0
9	не известен	NULL	0
10	не известен	NULL	0
11	qwe	2025-12-12	0
12	qwe	2025-12-12	0
13	qwe	2025-12-12	0
14	qwe	2025-12-12	0
15	Петрова Яна Андреевна	2006-09-13	0
NULL	NULL	NULL	NULL

PostgreSql

```
CREATE OR REPLACE FUNCTION max_average()  
RETURNS TABLE(Surname TEXT, Namee TEXT) AS $$  
BEGIN  
    RETURN QUERY  
    SELECT s.Surname::TEXT, s.Namee::TEXT  
    FROM Students s  
    JOIN Progress p ON p.Code_stud = s.Code_stud  
    WHERE p.Estimate = (  
        SELECT MAX(Estimate) FROM Progress  
    );  
END;  
$$ LANGUAGE plpgsql;
```

Возвращает фамилию и имя студента с самым максимальным баллом по оценкам, соединяя таблицы Students и Progress.



	max_average record
1	Петрова, Яна
2	Петрова, Яна
3	Петрова, Яна
4	Петрова, Яна
5	Рафикова, Азалия
6	Фанисовна, Валерия
7	Хасанова, Милана

```
CREATE OR REPLACE FUNCTION  
avg_estimate_of_students(SurnameInput TEXT, NameeInput TEXT,  
LastnameInput TEXT)  
RETURNS NUMERIC AS $$  
BEGIN  
    RETURN(  
        SELECT AVG(Estimate) FROM Progress  
        WHERE Code_stud = (SELECT Code_stud FROM Students  
        WHERE Surname = SurnameInput AND Namee = NameeInput AND  
        Lastname = LastnameInput);
```

```

SELECT p.Estimate FROM Progress p
JOIN Students s on s.Code_stud = p.Code_stud
WHERE s.Surname = SurnameInput and s.Nameee = NameeInput
and s.Lastname = LastnameInput
);
END;
$$ LANGUAGE plpgsql;

```

Возвращает среднюю оценку конкретного студента по ФИО.

1 `select * from public."avg_estimate_of_students"('Иванов', 'Павел', 'Сергеевич');`

Data Output Сообщения Notifications

Showing rows: 1 to 1

	avg_estimate_of_students	
	numeric	
1	3	

```

CREATE OR REPLACE FUNCTION Group_of_students(SurnameInput
TEXT, NameeInput TEXT, LastnameInput TEXT)
RETURNS table(Num_course TEXT, Name_speciality TEXT) AS $$
BEGIN
RETURN QUERY
SELECT g.Num_course::TEXT, g.Name_speciality::TEXT FROM
Groupss g
JOIN Students s on s.Code_group = g.Code_group
WHERE s.Surname = SurnameInput and s.Nameee = NameeInput
and s.Lastname = LastnameInput;
END;

```

Показывает курс и специальность студента по ФИО

1 `select * from public."group_of_students"('Иванов', 'Павел', 'Сергеевич')`

Data Output Сообщения Notifications

Showing rows

	num_course	name_speciality
	text	text
1	3	Информационные системы и программирование

```

CREATE OR REPLACE FUNCTION Insert_Student(
SurnameInput VARCHAR(25),

```

```

NameeInput VARCHAR(20),
LastnameInput VARCHAR(30),
BirthdayInput DATE,
PhoneInput VARCHAR(11),
Code_groupInput INT
)
RETURNS VOID AS
$$
BEGIN
    INSERT INTO Students (surname, namee, lastname, birthday, phone,
code_group)
    VALUES (SurnameInput, NameeInput, LastnameInput, BirthdayInput,
PhoneInput, Code_groupInput);
END;
$$ LANGUAGE plpgsql;

```

```

select * from public."insert_student"(
'Иванов',
'Павел',
'Сергеевич',
'"2006-09-13"',
'89965820492',
'1'
);

```

42	Рафикова	Кристина	Ягофаровна	2005-09-13	89965423684	2	[default]
43	Рафикова	Кристина	Ягофаровна	2005-09-13	89965423684	2	[default]
45	Иванов	Павел	Сергеевич	2006-09-13	89965820492	1	0

Добавляет нового студента в базу данных, по данным которые мы ввели

```

CREATE OR REPLACE FUNCTION Age()
RETURNS TABLE(Namee TEXT, Age TEXT, Messagee TEXT) AS $$
BEGIN
    RETURN QUERY

```



```

SELECT s.Namee::TEXT, extract(year from age(current_date,
birthday))::TEXT as Age,
CASE
    WHEN extract(year from age(current_date, birthday)) < (select
avg(extract(year from age(current_date, birthday))) as Age from
Students s)
    THEN 'Ваш возраст меньше среднего возраста всех студентов'
    WHEN extract(year from age(current_date, birthday)) > (select
avg(extract(year from age(current_date, birthday))) as Age from
Students s)
    THEN 'Ваш возраст больше среднего возраста всех студентов'
    ELSE 'Ваш возраст равен среднему возрасту всех студентов'
END AS Messagee
FROM Students s;
END;
$$ LANGUAGE plpgsql;

```

Выводит имя студента, его возраст и если он меньше среднего, выводится соответствующее сообщение, если выше - Ваш возраст больше среднего возраста всех студентов, иначе - возраст равен среднему.




SELECT public.age()	
Output	Сообщения Notifications
<div> </div>	
age	
record	
Яна,18,Ваш возраст меньше среднего возраста всех студентов	
Азалия,19,Ваш возраст равен среднему возрасту всех студентов	
Алесья,18,Ваш возраст меньше среднего возраста всех студентов	
Валерия,19,Ваш возраст равен среднему возрасту всех студентов	
Милана,18,Ваш возраст меньше среднего возраста всех студентов	
Павел,18,Ваш возраст меньше среднего возраста всех студентов	
Регина,19,Ваш возраст равен среднему возрасту всех студентов	
Алина,19,Ваш возраст равен среднему возрасту всех студентов	
Кристина,19,Ваш возраст равен среднему возрасту всех студентов	
Кристина,19,Ваш возраст равен среднему возрасту всех студентов	
Кристина,19,Ваш возраст равен среднему возрасту всех студентов	
Кристина,19,Ваш возраст равен среднему возрасту всех студентов	
Ильдар,24,Ваш возраст больше среднего возраста всех студентов	
Павел,18,Ваш возраст меньше среднего возраста всех студентов	

```

CREATE OR REPLACE FUNCTION Insert_subjects()
RETURNS VOID AS $$
DECLARE count_rec INTEGER;
BEGIN
    SELECT COUNT(*) INTO count_rec FROM Subjects;
    WHILE count_rec < 10 LOOP
        INSERT INTO Subjects(name_subject, count_hours)
        VALUES ('не известно', 0);
        count_rec := count_rec + 1;
    END LOOP;
END;
$$ LANGUAGE plpgsql;

```

Добавляет предметы “не известно” с нулевым количеством часов, если их меньше 10

	code_subject [PK] integer 	name_subject character varying (85) 	count_hours integer 
1	1	Английский язык	48
2	2	Землеведение	256
3	3	Экономика и бухгалтерский учёт	196
4	4	Разработка мобильных приложений	182
5	8	не известно	0
6	9	не известно	0
7	10	не известно	0
8	11	не известно	0
9	12	не известно	0
10	13	не известно	0

ТРИГГЕРЫ и ТРАНЗАКЦИИ

MySql

```
use db_bookss;  
delimiter //
```

```
create trigger RecUpdate  
before insert on Authors  
for each row  
begin  
    declare max_code int;  
  
    select COALESCE(max(Code_author), 0) into max_code from Authors;  
    set new.Code_author = max_code + 1;  
end//
```

Триггер, срабатывающий при занесении новой строки в таблицу Авторы. Автоматически увеличивает счётчик числа добавленных строк

14	qwe	2025-12-12	0
15	Петрова Яна Андреевна	2006-09-13	0
16	Петрова Яна Андреевна	2006-09-13	0
17	Петрова Яна Андреевна	2006-09-13	0

```
create procedure CountBookss()  
begin  
    update Authors a  
    set Count_books = (select count(*) from books b where  
b.Code_author = a.Code_author)  
    where a.Code_author is not null;  
end//
```

Создали процедуру и добавили в таблицу авторов поле Количество книг(по умолчанию 0). Подсчитывает количество книг по каждому автору и записывает в новое поле.

```

create trigger CountBooksAfterInsertBook
after insert on Books
for each row
begin
    update Authors a
    set Count_books = Count_books + 1
    where
CountBooksAfterInsertBookCountBooksAfterInsertBookCode_author =
new.Code_author;
end//

```

Создали триггер, срабатывающий при занесении новой информации о книге.

Code_author	Name_author	Birthday	Count_books
1	Чарльз Мартин Лефлер	1861-01-30	4
2	Агата Мэри Кларисса Миллер	1890-09-15	2
3	Джордж Самюэль Клейсон	1874-11-07	1
4	Пауло Козльо	1947-08-24	1
5	Франк Тиле	1973-10-15	1

```

create trigger AfterSupplyInsert
before insert on purchases
for each row
begin
    if NEW.Amount < 10 then
        set NEW.Cost = NEW.Cost * 1.20;
    end if;
end//

```

Триггер для повышения цены книг на 20% при поставках, если остаток меньше 10 экземпляров

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount
1	1	2023-03-08	1	опт	310	10
2	2	2023-03-10	1	розница	5	1
3	3	2023-03-12	3	розница	412	1
4	4	2023-03-15	2	розница	415	1
5	5	2023-03-17	1	розница	408	1
6	6	2023-12-19	3	опт	420	20

```

create trigger BanInsertDeliveries
before insert on Deliveries
for each row
begin
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = "Вставка строк запрещена";
end//

```

```

START TRANSACTION;
    INSERT INTO Publishing_house(Publish, City)
VALUES('ОАО Радуга','Москва');

    INSERT INTO Publishing_house(Publish, City)
VALUES('ОАО Тюльпан','Санкт-Петербург');
COMMIT;

```

Запретили вставку строк в таблицу поставщики, выводя при этом сообщение.

```

insert into Purchases(Code_book, Date_order, Code_delivery,
Type_purchase, Cost, Amount)
values (20,'2025-12-23',1,'опт',150,25);

```

	Message
.UES('ОАО Радуга','...)	0 row(s) affected
.UES('ОАО Радуга','...)	Error Code: 1644 Вставка строк запрещена

PostgreSql

```
CREATE OR REPLACE FUNCTION update_author()  
RETURNS TRIGGER AS $$  
DECLARE  
    max_code INTEGER;  
BEGIN  
    SELECT COALESCE(MAX(Code_author), 0) INTO max_code FROM  
Authors;  
    NEW.Code_author := max_code + 1;  
    RETURN NEW;  
end;  
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER update_author_trigger  
BEFORE INSERT ON Lectors  
FOR EACH ROW  
EXECUTE FUNCTION update_author();
```

Триггер, срабатывающий при занесении новой строки в таблицу преподаватели. Увеличивает счётчик числа добавленных строк

code_lector [PK] integer	name_lector character varying (85)	science character varying (25)	post character varying (25)	datee date
1	Ахметова Айгуль Мутагаровна	Кандидат наук	Преподаватель	2000-01-05
2	Шарипова Диана Айдаровна	Магистр	Преподаватель	2006-05-08
3	Хасанова Анастасия Ильдаровна	Магистр	Преподаватель	2019-02-15
4	Набиева Лиана Рамилевна	Бакалавр	Преподаватель	2003-08-06
5	Петров Савелий Яковлевич	Магистр	[null]	[null]

```
CREATE OR REPLACE FUNCTION avg_estimate_students()  
RETURNS TRIGGER AS $$  
BEGIN  
    UPDATE students s  
    SET "Avg_estimate" = (  
        SELECT AVG(p.estimate) FROM progress p  
        WHERE p.Code_stud = s.Code_stud  
    )  
    WHERE s.Code_stud is not null;
```

```

RETURN NEW;
end;
$$ LANGUAGE plpgsql;

```

```

create trigger update_avg_estimate
after insert on progress
for each row
execute function avg_estimate_students();

```

В таблицу Студенты добавили поле Средний балл(по умолчанию 0), создали хранимую процедуру, которая подсчитывает средний балл для каждого студента и заносит его в новое поле. Создали триггер, который запускается после новых оценок студентов и автоматически обновляет средний бал студента.

1	Петрова	Яна	Андреевна	2006-09-13	89965820492	1	4.8
2	Рафикова	Азалия	Ильшатовна	2005-10-01	89871088391	2	5
3	Хабирова	Алесия	Игоревна	2007-05-06	89966842536	3	3.6666666666666665
4	Фанисовна	Валерия	Владимировна	2005-10-12	89965236142	1	4
5	Хасанова	Милана	Радионова	2007-01-02	8996523968	2	5
6	Иванов	Павел	Сергеевич	2006-10-12	89965683295	4	3

```

CREATE OR REPLACE PROCEDURE add_student_if_not(
    surname_stud CHARACTER,
    name_stud CHARACTER,
    lastname_stud CHARACTER,
    birt_stud DATE,
    code_group INTEGER
)
LANGUAGE plpgsql AS $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Students WHERE Code_stud =
code_studd) THEN
        INSERT INTO Students (surname, namee, lastname,
birthday, phone, code_group)
VALUES (
    surname_stud,
    name_stud,
    lastname_stud,
    '2000-12-12'::birt_stud,
    NULL,

```



```

        code_group
    );
END IF;
END;
$$;

CREATE OR REPLACE FUNCTION add_student_if_not_trigger()
RETURNS TRIGGER
LANGUAGE plpgsql AS $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM Students WHERE Code_stud =
NEW.Code_stud) THEN
        CALL add_student_if_not(
            'Шашкин',
            'Ильдар',
            'Сергеевич',
            1
        );
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER progress_check_student2
BEFORE INSERT ON Progress
FOR EACH ROW
EXECUTE FUNCTION add_student_if_not_trigger();

```

Создали триггер, который запускается при новых оценках. Есть проверка на наличие студента, если студента нет, то запускается процедура на вставку записи в таблицу Студенты(параметры заданы произвольно).

9	Шашкин	Ильдар	Сергеевич	2000-12-12	[null]	1	3.5
10	Шашкин	Ильдар	Сергеевич	2000-12-12	[null]	1	3
12	Шашкин	Ильдар	Сергеевич	2000-12-12	[null]	1	4

```

CREATE OR REPLACE FUNCTION BanInsert()

```

```

RETURNS TRIGGER
LANGUAGE plpgsql AS $$
BEGIN
    RAISE EXCEPTION 'Вставка строк запрещена';
    RETURN NULL;
END;
$$;

```

```

CREATE TRIGGER ban_insert_trigger
BEFORE INSERT ON Groupss
FOR EACH ROW
EXECUTE FUNCTION BanInsert();

```

Запретили вставлять новые строки в таблицу группы, выводя при этом сообщение

❗ Вставка строк запрещена CONTEXT: функция PL/pgSQL baninsert(), строка 3, оператор RAISE >

```

BEGIN;
SELECT COUNT(*) FROM Lectors;

```

```

INSERT INTO Lectors (name_lector, science, post, datee)
VALUES ('Иванов Иван Иванович', 'Кандидат наук', 'Доцент', '2023-01-15');

```

```

SELECT COUNT(*) FROM Lectors;
COMMIT;

```

Проверили выполняются ли команды транзакции при добавлении преподавателей

code_lector [PK] integer	name_lector character varying (85)	science character varying (25)	post character varying (25)	datee date
1	Ахметова Айгуль Мутагаровна	Кандидат наук	Преподаватель	2000-01-05
2	Шарипова Диана Айдаровна	Магистр	Преподаватель	2006-05-08
3	Хасанова Анастасия Ильдаровна	Магистр	Преподаватель	2019-02-15
4	Набиева Лиана Рамилевна	Бакалавр	Преподаватель	2003-08-06
5	Петров Савелий Яковлевич	Магистр	[null]	[null]
6	Иванов Иван Иванович	Кандидат наук	Доцент	2023-01-15

РАБОТА С ПОЛЬЗОВАТЕЛЕМ

```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'Admin123!';  
GRANT ALL PRIVILEGES ON db_books.* TO 'Admin'@'localhost' WITH  
GRANT OPTION;
```

Создали нового пользователя - админа, который может подключаться только с локального хоста, присвоили ему пароль и дали ему все права на бд db_books и таблицы внутри нее, с возможностью передачи своих прав другому

```
CREATE USER 'dispatcher'@'localhost' IDENTIFIED BY 'dispatcher1234!';  
GRANT SELECT, INSERT, UPDATE ON db_bookss.books TO  
'dispatcher'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON db_bookss.authors TO  
'dispatcher'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON db_bookss.publishing_house TO  
'dispatcher'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON db_bookss.deliveries TO  
'dispatcher'@'localhost';
```

Создали нового пользователя - диспетчер, который может подключаться только с локального хоста, присвоили ему пароль и дали ему права на чтение, добавление и изменение авторов, книг, издательств и поставщиков

```
CREATE USER 'supply'@'localhost' IDENTIFIED BY 'supply12345!';  
GRANT SELECT, UPDATE ON db_bookss.authors TO 'supply'@'localhost';  
GRANT SELECT, UPDATE ON db_bookss.books TO 'supply'@'localhost';  
GRANT SELECT, UPDATE ON db_bookss.deliveries TO  
'supply'@'localhost';  
GRANT SELECT, UPDATE ON db_bookss.publishing_house TO  
'supply'@'localhost';  
GRANT SELECT, INSERT, UPDATE ON db_bookss.purchases TO  
'supply'@'localhost';
```

Создали нового пользователя, присвоили ему пароль и дали права на просмотр и добавление авторов, книг, поставщиков, издательств, поставок

```
CREATE VIEW supplier1_deliveries AS  
SELECT * FROM db_bookss.deliveries WHERE Code_delivery = 1;
```

```
CREATE USER 'supplier1'@'localhost' IDENTIFIED BY 'Supplier1Pass!';  
GRANT SELECT ON db_bookss.supplier1_deliveries TO  
'supplier1'@'localhost';
```

Создали представление для поставщика. Создали пользователя с доступом только к этому представлению и присвоили ему пароль

```
SHOW GRANTS FOR 'admin'@'localhost';  
SHOW GRANTS FOR 'dispatcher'@'localhost';  
SHOW GRANTS FOR 'supply'@'localhost';  
SHOW GRANTS FOR 'supplier1'@'localhost';
```

Grants for admin@localhost
GRANT USAGE ON *.* TO `admin`@`localhost`
Grants for dispatcher@localhost
GRANT USAGE ON *.* TO `dispatcher`@`localhost`
GRANT SELECT, INSERT, UPDATE ON `db_bookss`.`books` TO `dispatcher`@`localhost`
GRANT SELECT, INSERT, UPDATE ON `db_bookss`.`deliveries` TO `dispatcher`@`localhost`
GRANT SELECT, INSERT, UPDATE ON `db_bookss`.`publishing_house` TO `dispatcher`@`localhost`
Grants for supply@localhost
GRANT USAGE ON *.* TO `supply`@`localhost`
GRANT SELECT, UPDATE ON `db_bookss`.`authors` TO `supply`@`localhost`
GRANT SELECT, UPDATE ON `db_bookss`.`books` TO `supply`@`localhost`
GRANT SELECT, UPDATE ON `db_bookss`.`deliveries` TO `supply`@`localhost`
GRANT SELECT, UPDATE ON `db_bookss`.`publishing_house` TO `supply`@`localhost`
GRANT SELECT, INSERT, UPDATE ON `db_bookss`.`purchases` TO `supply`@`localhost`
Grants for supplier1@localhost
GRANT USAGE ON *.* TO `supplier1`@`localhost`
GRANT SELECT ON `db_bookss`.`supplier1_deliveries` TO `supplier1`@`localhost`