ВИДЕОКУРС Основы Баз данных от Geek Brains (20 уроков) https://www.mysql.com/

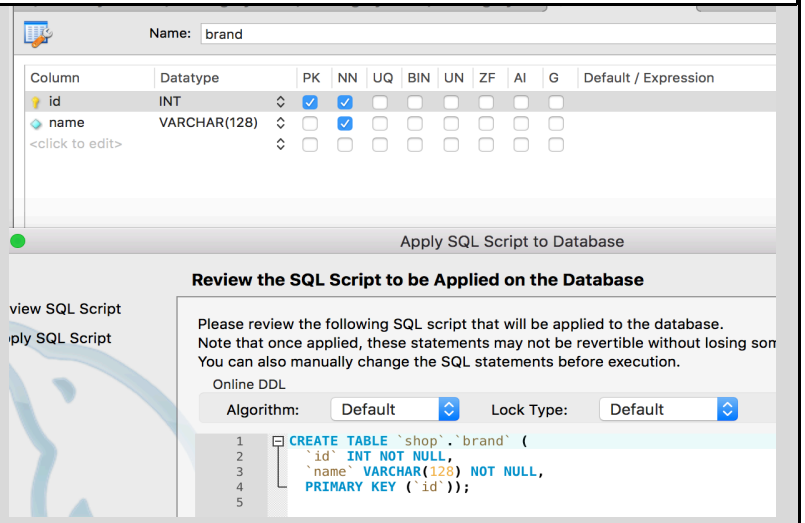
Эта шпаргалка будет бесполезна без просмотра видеокурса

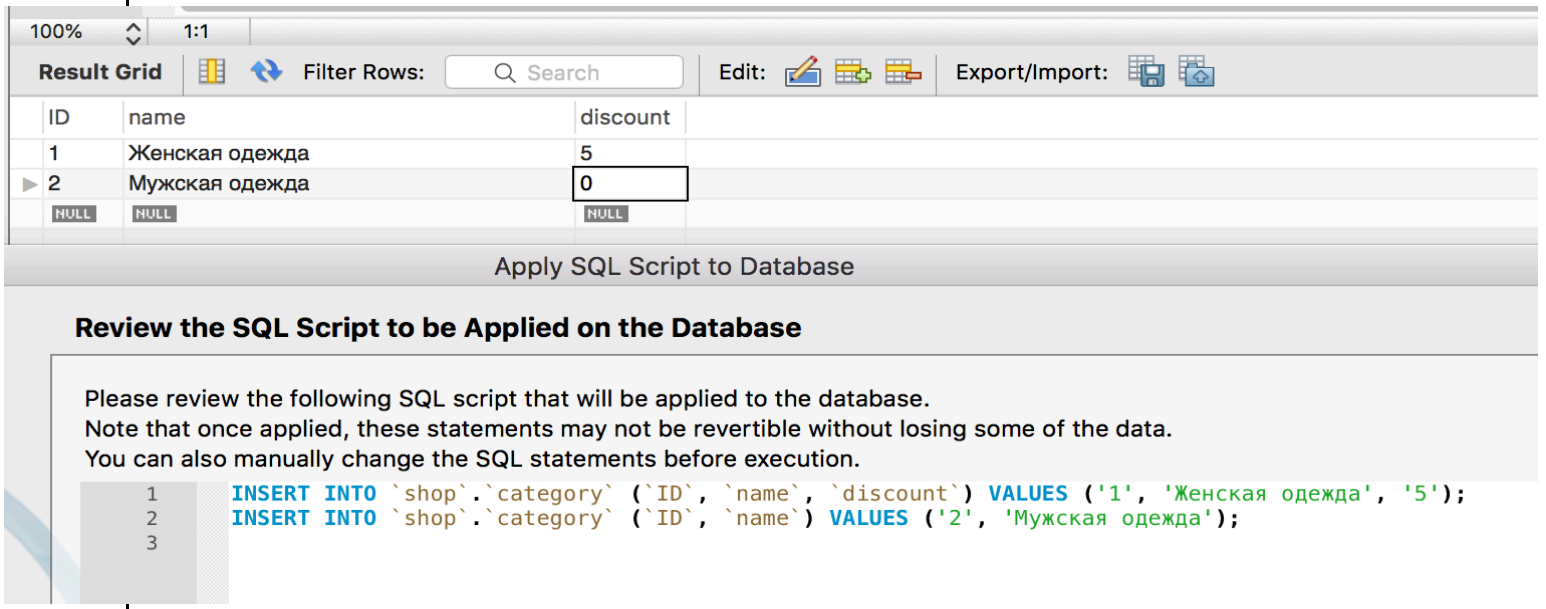
https://geekbrains.ru/courses/86

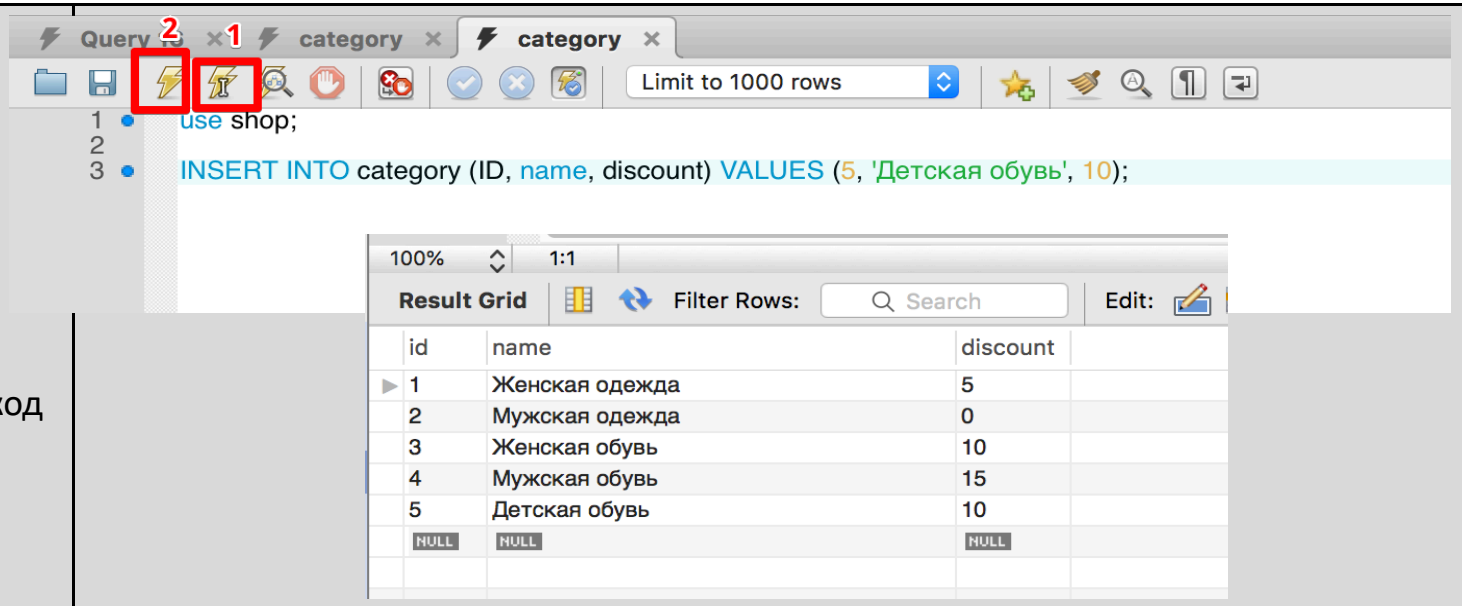
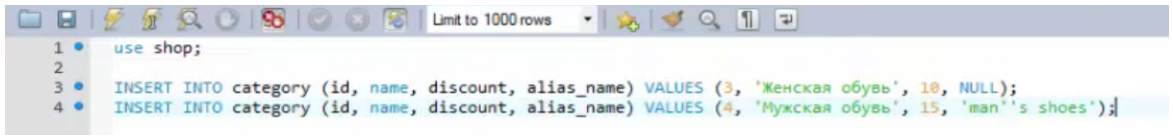
СОДЕРЖАНИЕ (файл состоит из нескольких листов)

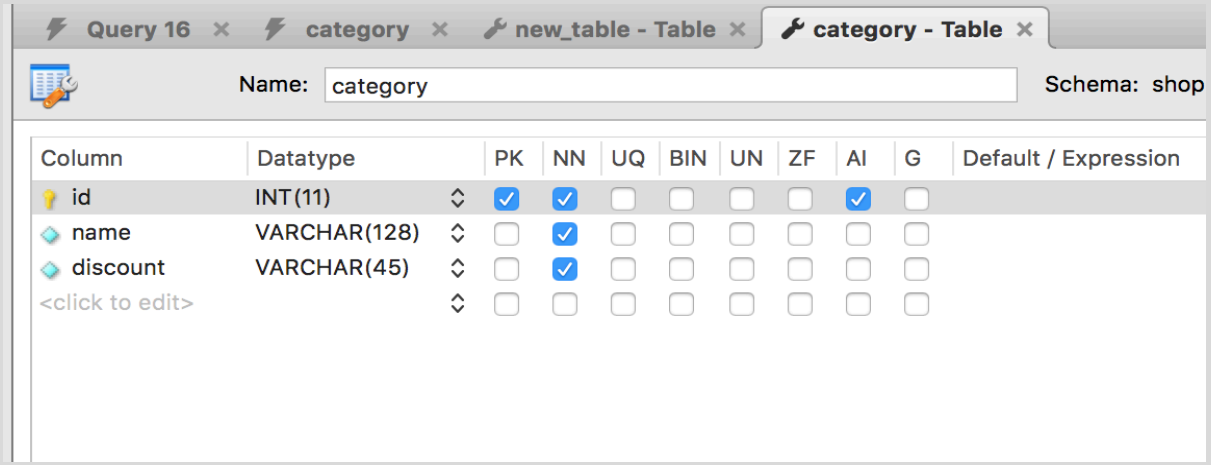
Присоединяйтесь ко мне в Instagram, делюсь наглядными инструкциями и шпаргалками по курсам, которые я прошла на Geek Brains

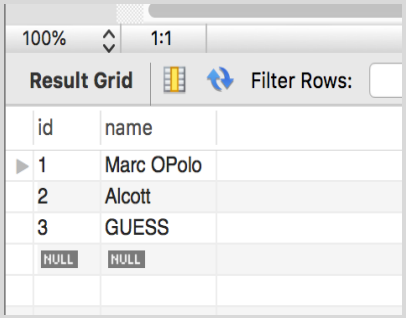
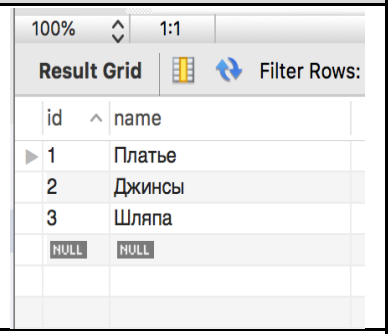
https://www.instagram.com/natalya_dol_frontender/

Lesson 5	Команды на языке SQL, ЗАПОЛНЯЕМ ТАБЛИЦЫ	
№	Действие	Команда на языке SQL и скрин результата
1	CREATE the Tables. Table - Create table	CREATE TABLE `shop`.`brand` (`id` INT NOT NULL, `name` VARCHAR(128) NOT NULL, PRIMARY KEY (`id`)); 

2	Заполнение таблицы	 **INSERT INTO `shop`.`category` (`ID`, `name`, `discount`) VALUES ('1', 'Женская одежда', '5');** **INSERT INTO `shop`.`category` (`ID`, `name`) VALUES ('2', 'Мужская одежда');**

3	Заполнение таблицы через код SQL	 **use shop;** **INSERT INTO category (ID, name, discount) VALUES (3, 'Женская обувь', 10);** **INSERT INTO category(ID, name, discount) VALUES (4, 'Мужская обувь', 15);**
4	Если текст с ковычками смотри 'man' 's shoes'	

5 (6:19)	Чтобы не прописывать каждый раз ID	**при создании таблицы нужно поставить галочку в AI в строке ID. Кликаем Alter Table и правим**
		 **ALTER TABLE `shop`.`category` CHANGE COLUMN `id` `id` INT(11) NOT NULL AUTO_INCREMENT ;**
6	Вставить строку в таблицу теперь можно так, выполняем команду, нажимаем молнию	**use shop;** **INSERT INTO category (name, discount) VALUES ('Шляпы', 0);**

7	Заполняем таблицу brand	**use shop;** **INSERT INTO brand (name) VALUES ('Marc O' 'Polo');** **INSERT INTO brand (name) VALUES ('Alcott');** **INSERT INTO brand (name) VALUES ('GUESS');** 
8	Заполняем таблицу product type	**use shop;** **INSERT INTO product_type (name) VALUES ('Платье');** **INSERT INTO product_type (name) VALUES ('Джинсы');** **INSERT INTO product_type (name) VALUES ('Шляпа');** 

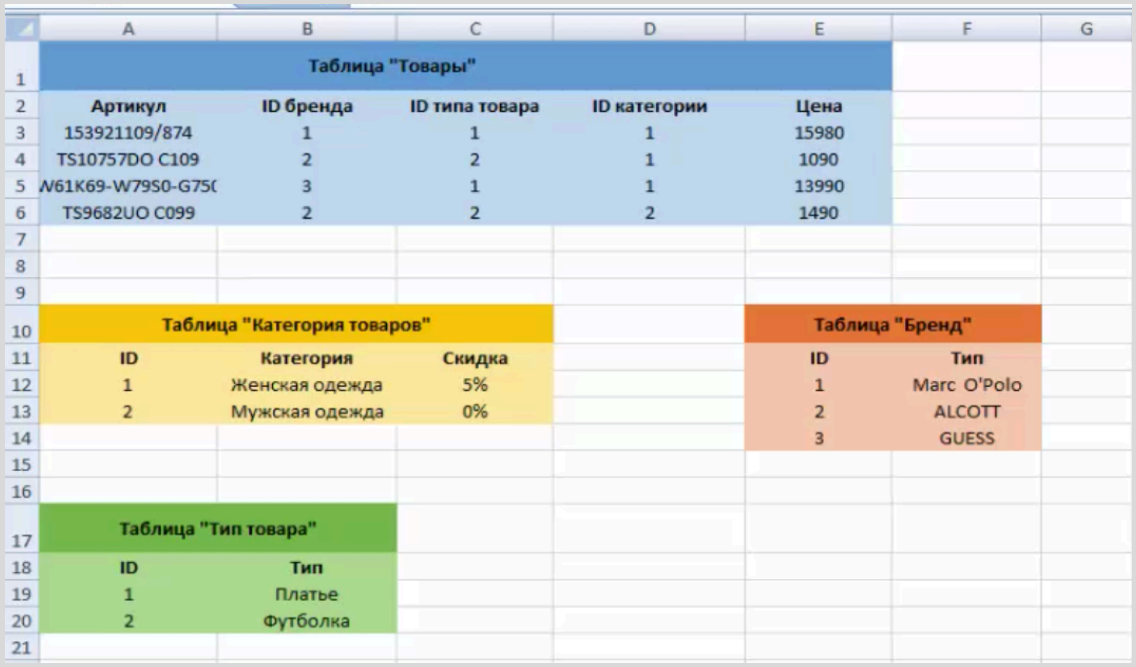
	Образец базы	

Таблица "Товары"

Артикул	ID бренда	ID типа товара	ID категории	Цена
153921109/874	1	1	1	15980
TS10757DO C109	2	2	1	1090
W61K69-W79S0-G75(	3	1	1	13990
TS9682UO C099	2	2	2	1490

Таблица "Категория товаров"

ID	Категория	Скидка
1	Женская одежда	5%
2	Мужская одежда	0%

Таблица "Бренд"

ID	Тип
1	Marc O'Polo
2	ALCOTT
3	GUESS

Таблица "Тип товара"

ID	Тип
1	Платье
2	Футболка

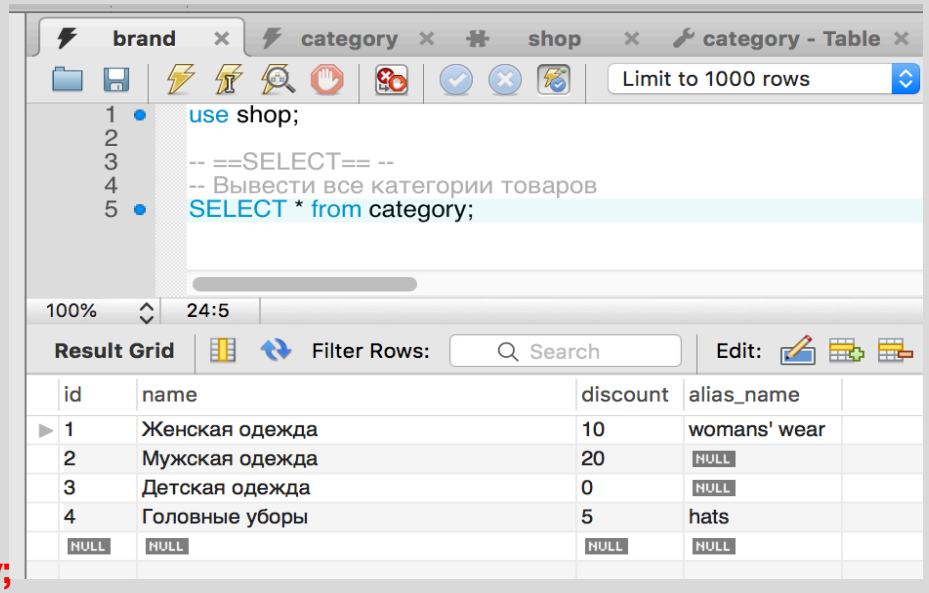
!!!!!!!

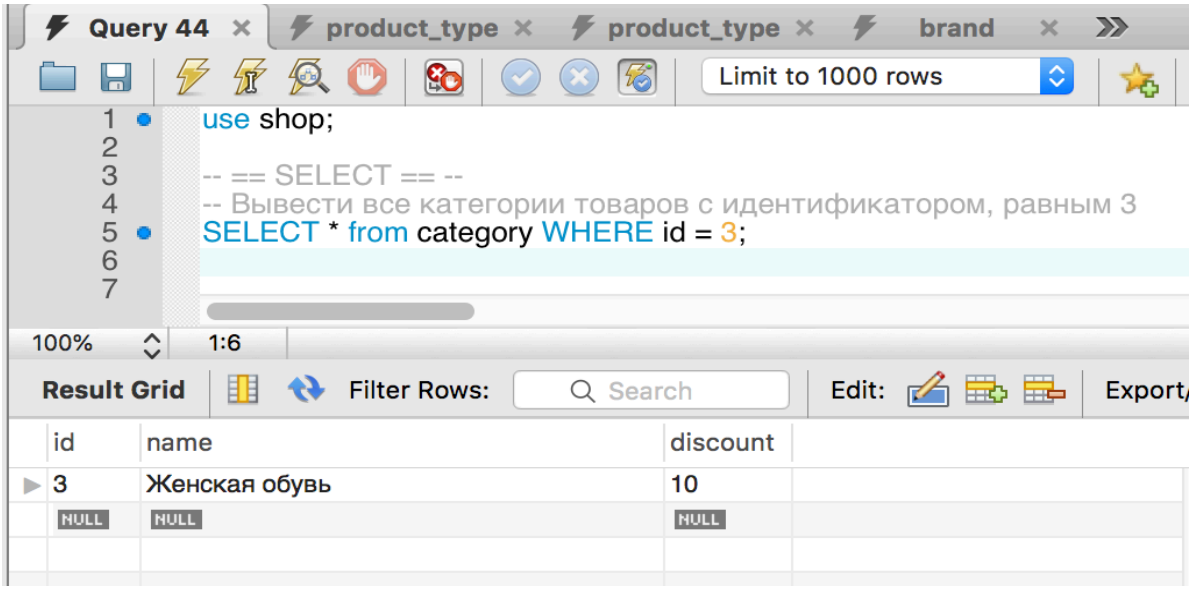
Комментарии обрамляем в -- текст --

```
-- == SELECT <столбец> == --
-- вывести названия всех категорий товаров
SELECT name FROM category;

-- вывести названия и скидки товаров
SELECT discount, name  FROM category;

-- вывести все скидки
SELECT discount FROM category;
```

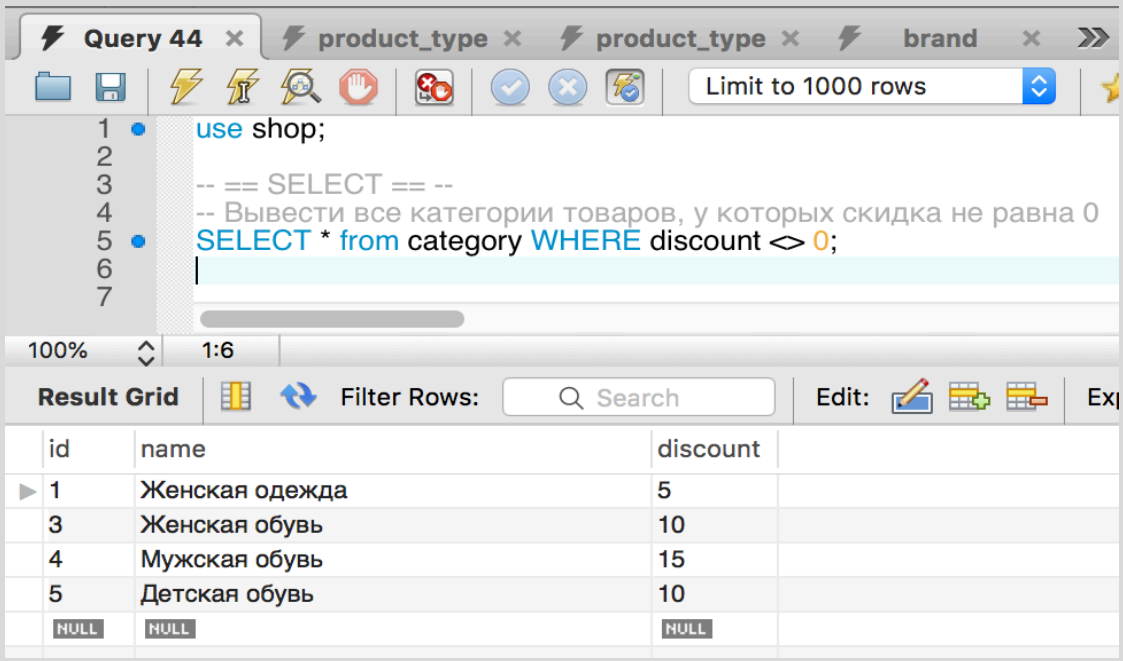
Lesson 6

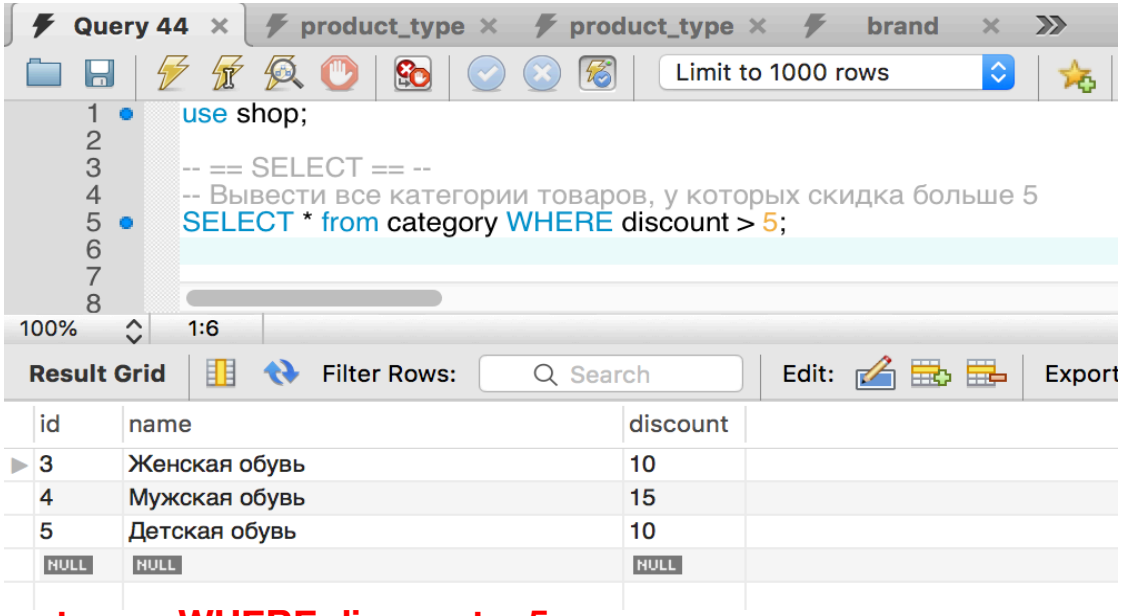
Урок 6. SQL-команды SELECT и WHERE для вывода данных из базы/ SQL commands SELECT and WHERE

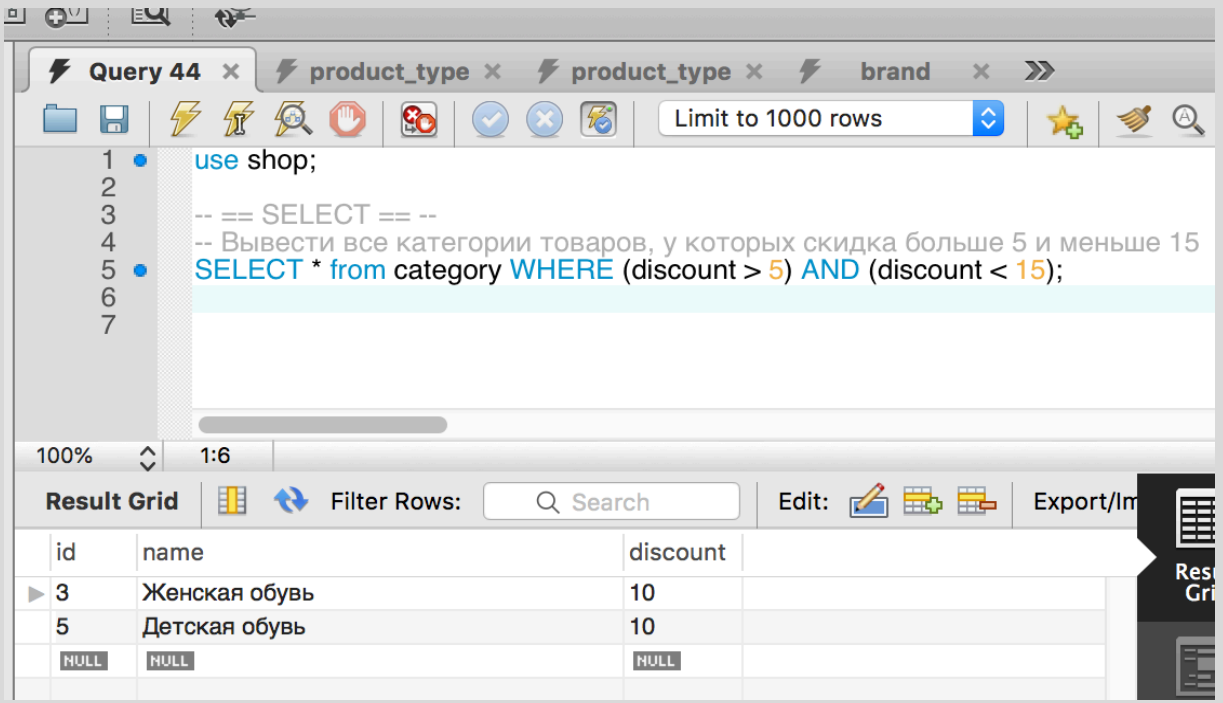
№	Действие	Команда на языке SQL и скрин результата
1	All categories Вывести все категории товаров	**use shop;** **SELECT * from category;** 

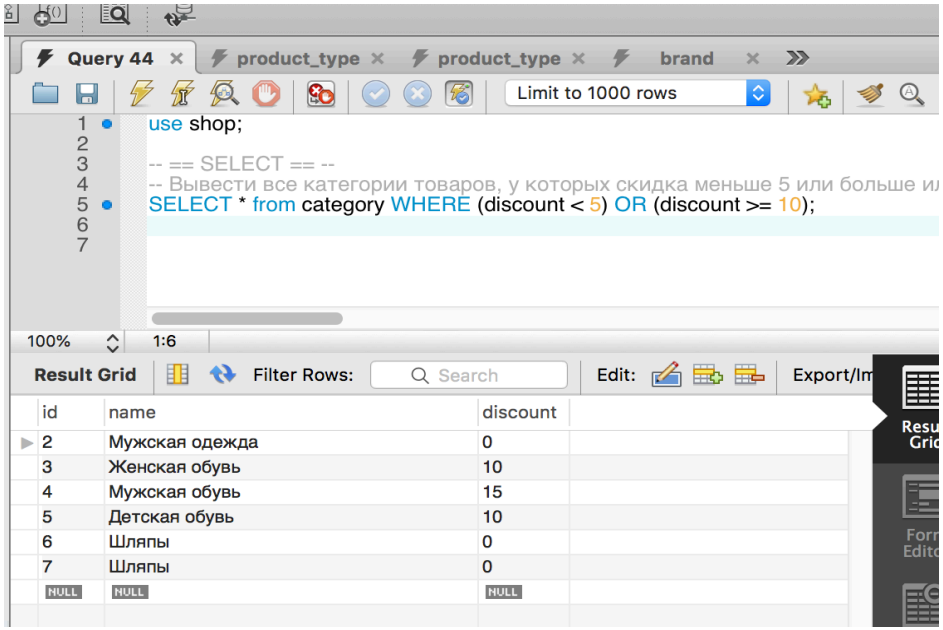
2	По значению	 **use shop;** **SELECT * from category WHERE id = 3;**

3	По условию Вывести все категории товаров, у которых скидка не равна 0	**use shop;** **SELECT * from category WHERE discount <> 0;**

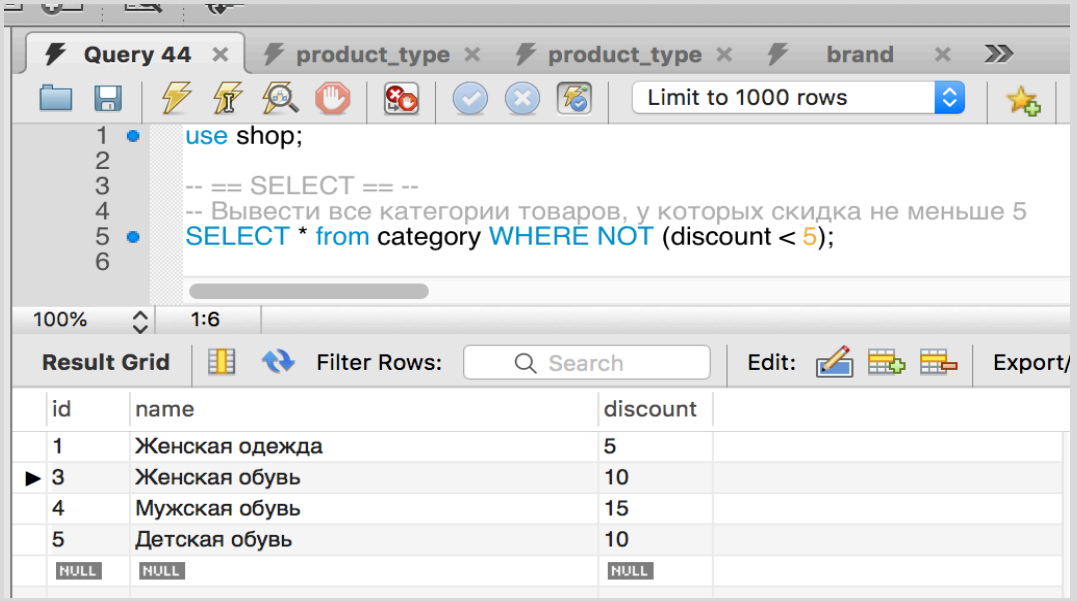


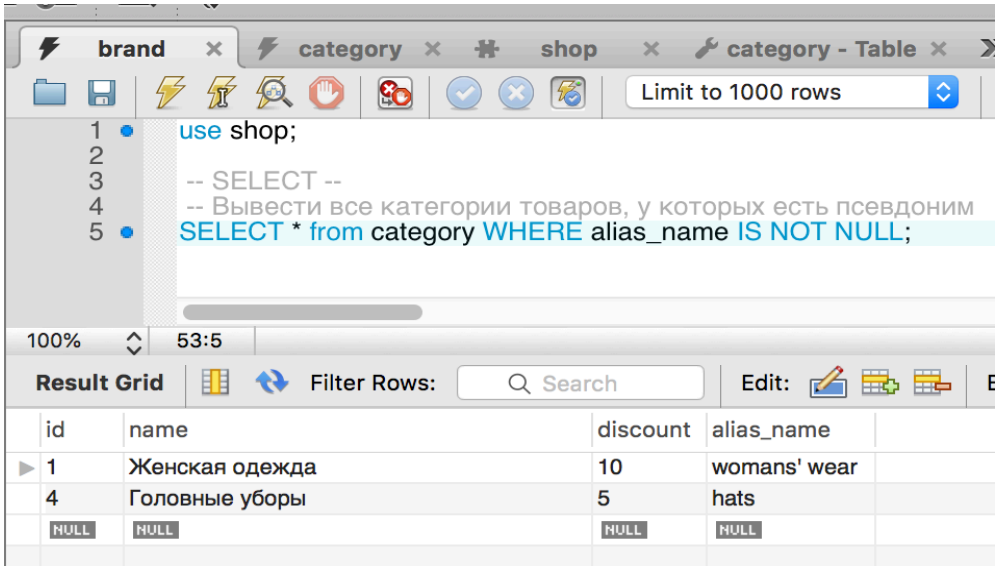
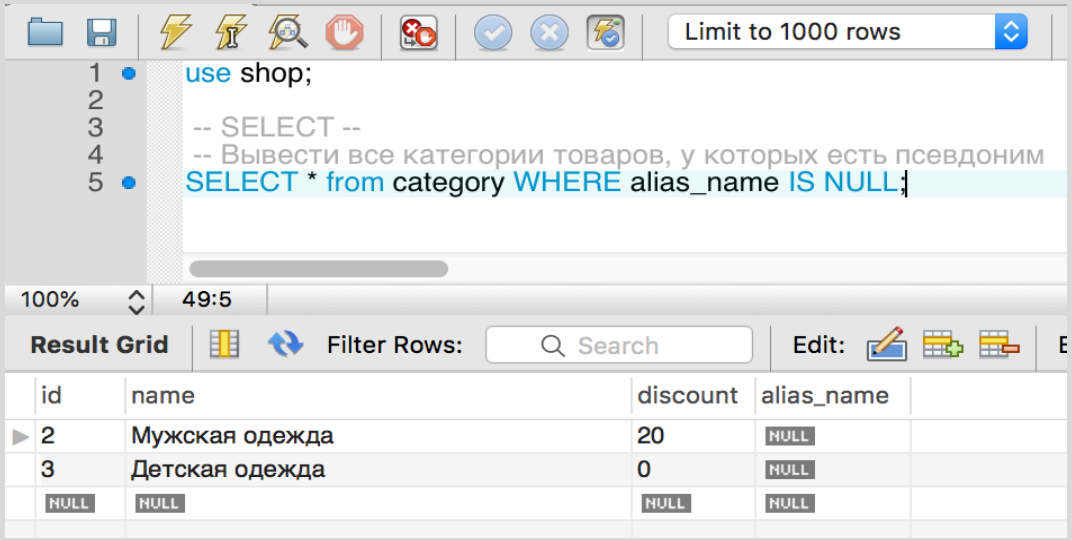
4	Вывести все категории товаров, у которых скидка больше 5	 **use shop;** **SELECT * from category WHERE discount > 5;**

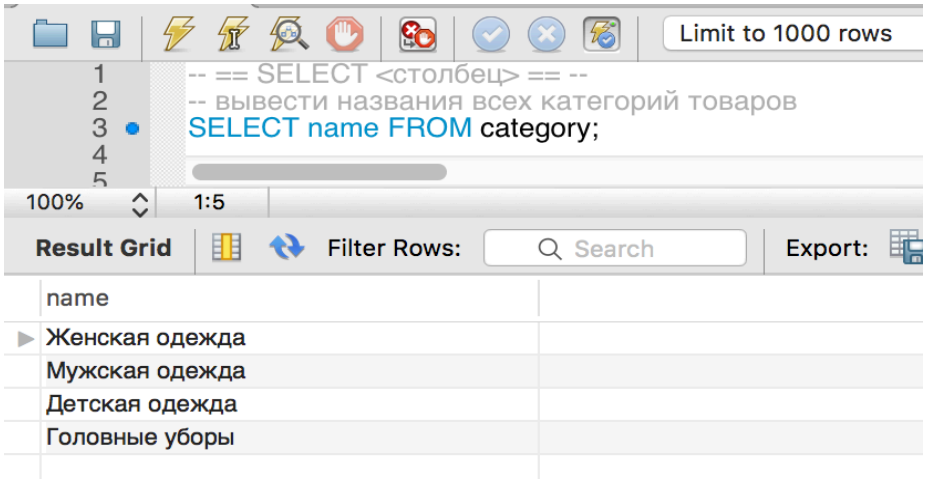
5	Вывести все категории товаров, у которых скидка больше 5 и меньше 15	**use shop;**  **SELECT * from category WHERE (discount > 5) AND (discount < 15);**

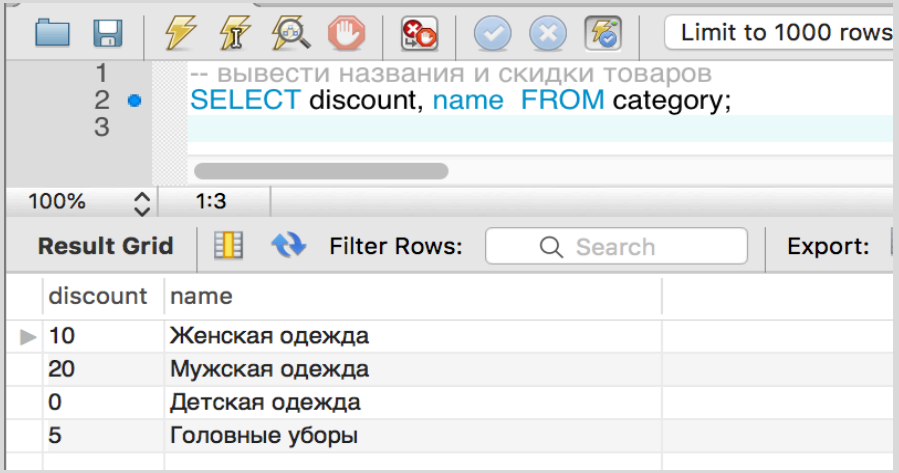
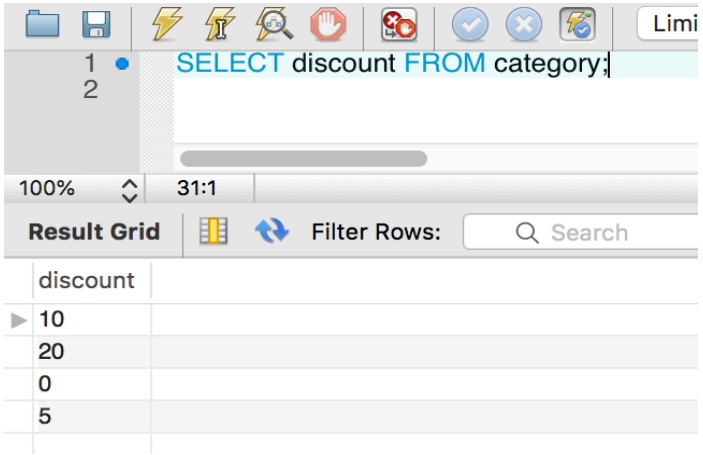
6	Вывести все категории товаров, у которых скидка меньше 5 или больше или равна 10	**use shop;**  **SELECT * from category WHERE (discount < 5) OR (discount >= 10);**

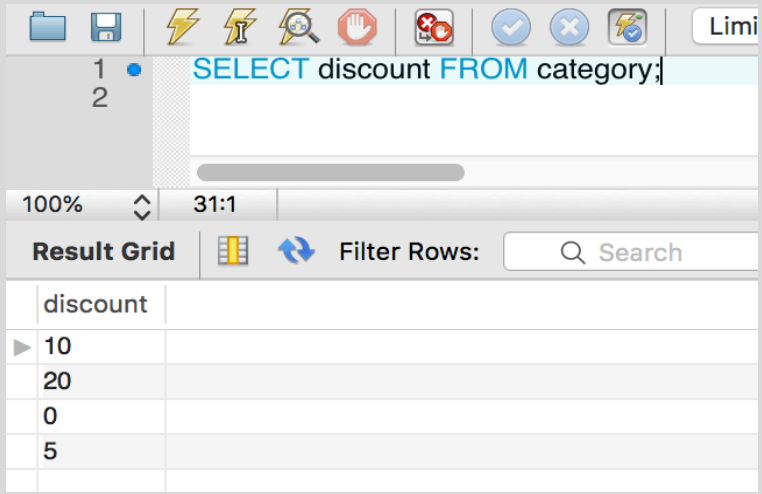
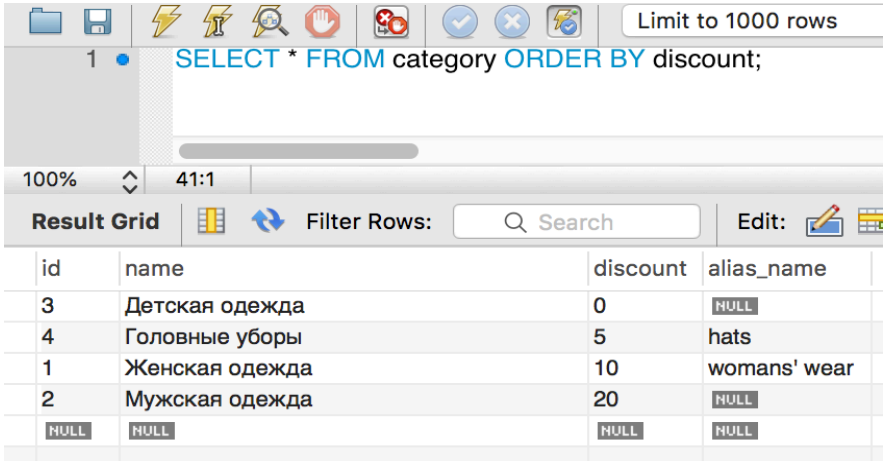
7	Вывести все категории товаров, у которых скидка не меньше 5	**use shop;** **SELECT * from category WHERE NOT (discount < 5);**

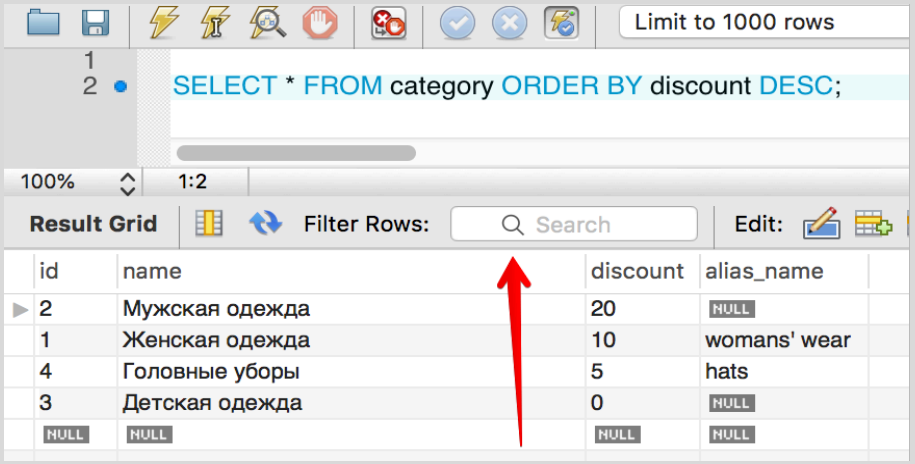
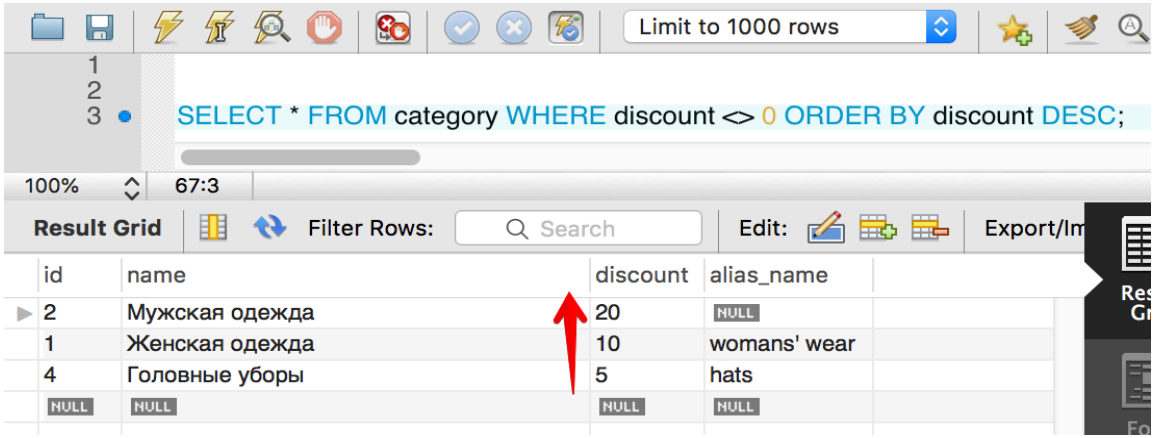


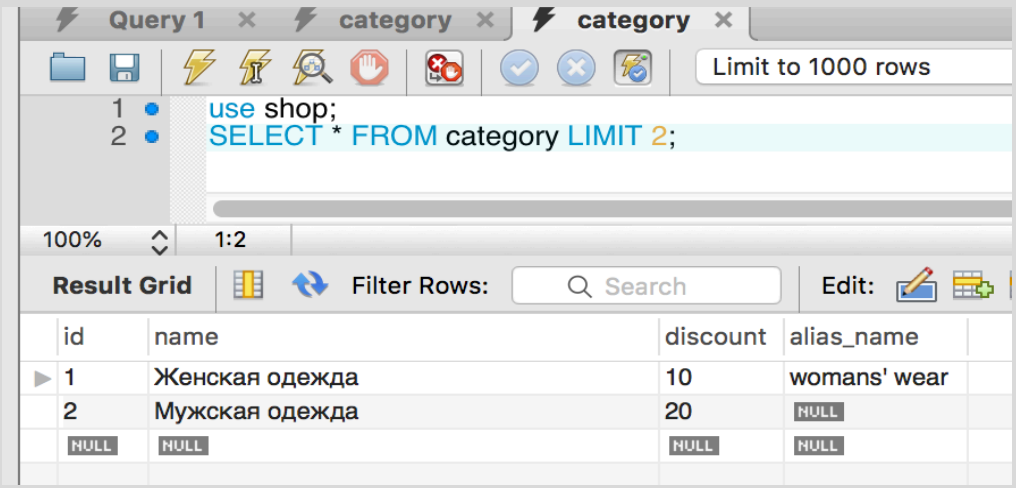
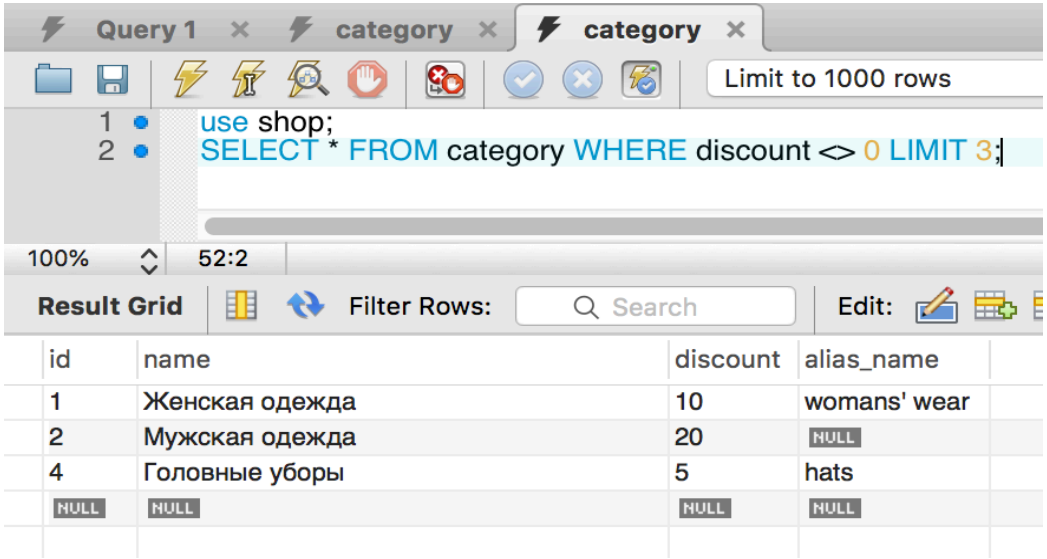
8	Вывести тех у кого есть значение в alias_name Вывести все категории товаров, у которых есть псевдоним	**use shop;**  **SELECT * from category WHERE alias_name IS NOT NULL;**
9	Вывести тех у кого НЕТ значения в alias_name Вывести все категории товаров, у которых есть псевдоним	**use shop;**  **SELECT * from category WHERE alias_name IS NULL;**

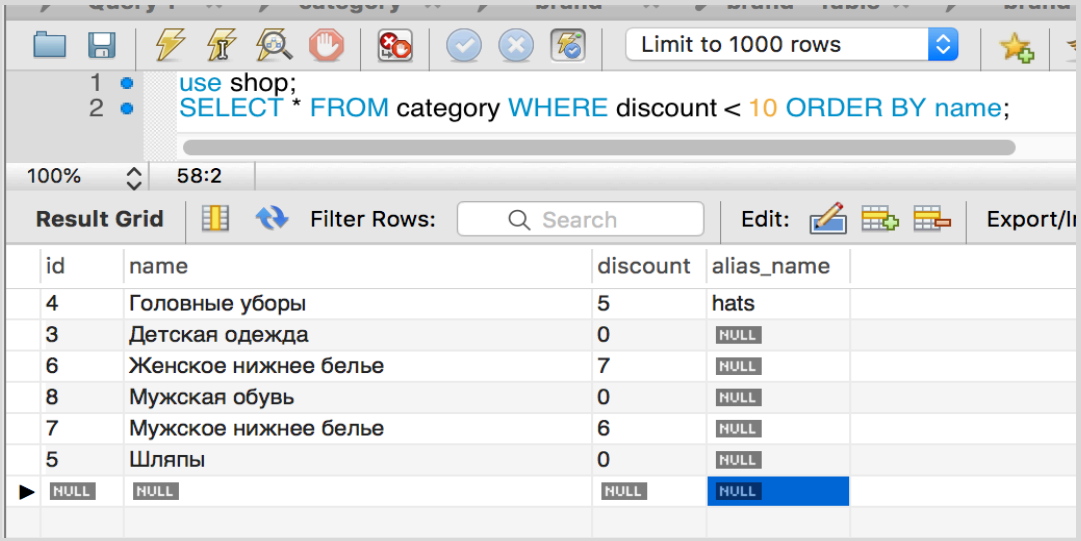
Lesson 7		Урок 7. SQL-команды DISTINCT, ORDER BY, LIMIT • Получение данных: • SELECT • WHERE • DISTINCT • ORDER BY (ASC, DESC) • LIMIT
№	Действие	Команда на языке SQL и скрин результата
1	вывести названия всех категорий товаров	 use shop; SELECT name FROM category;

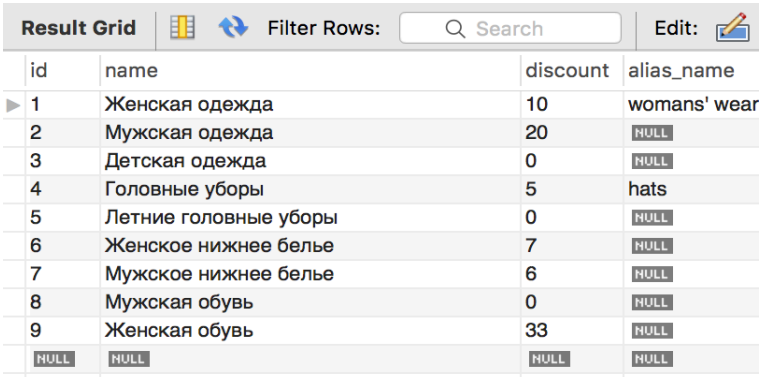
2	вывести названия и скидки товаров	 **use shop; SELECT discount, name FROM category;**
3	вывести все скидки	 **use shop; SELECT discount FROM category;**

4	**DISTINCT** вывести все **уникальные** значения скидок	 **use shop; SELECT DISTINCT discount FROM category;**
5	**ORDER BY** вывести все категории товаров, и **отсортировать** их по размеру скидки	 **use shop; SELECT * FROM category ORDER BY discount;**

6	ORDER BY вывести все категории товаров, и **отсортировать их по размеру скидки в обратном порядке**	 **use shop; SELECT * FROM category ORDER BY discount DESC;**
7	вывести все категории товаров **с ненулевой скидкой, и отсортировать их по размеру скидки** в обратном порядке	 **use shop; SELECT * FROM category WHERE discount <> 0 ORDER BY discount DESC;**

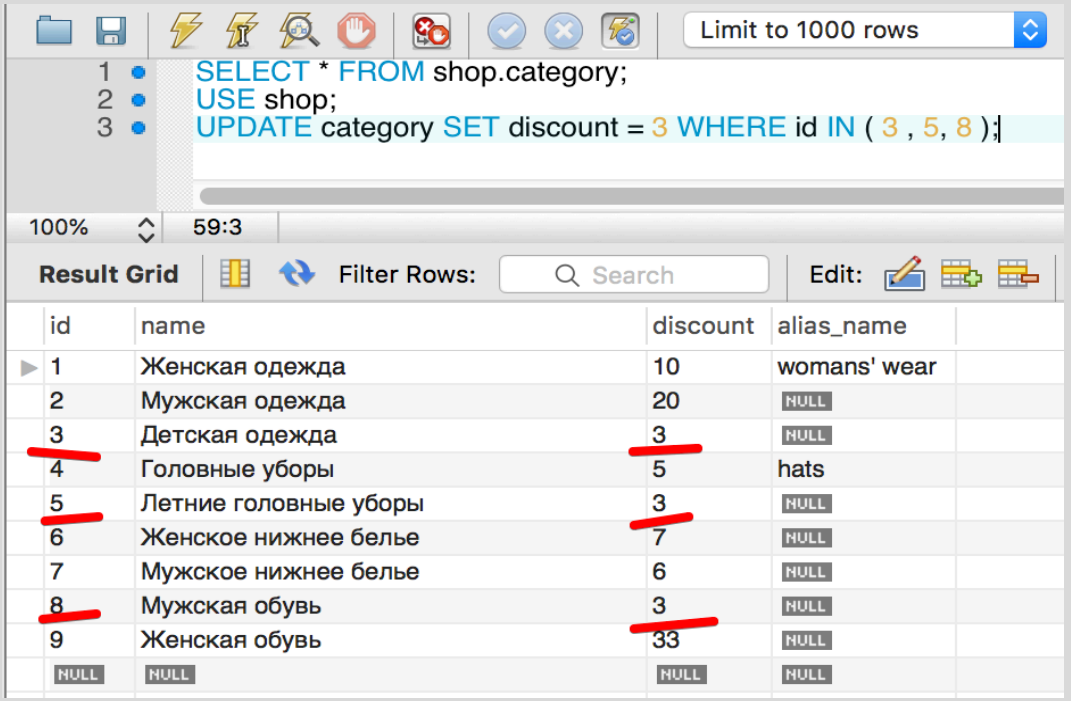
8	LIMIT вывести **первые 2 категории товара**	 **use shop; SELECT * FROM category LIMIT 2;**
9	вывести первые 3 категории товара со скидкой не равной нулю	 **use shop;** **SELECT * FROM category WHERE discount <> 0 LIMIT 3;**

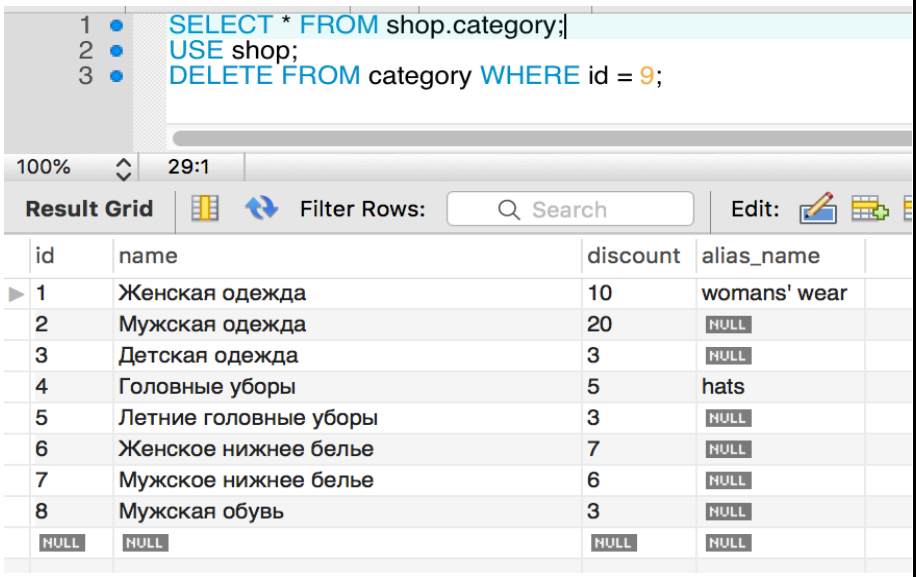
10	Получить все категории товаров со скидкой < 10%, и отсортировать их по названию	 **use shop;** **SELECT * FROM category WHERE discount < 10 ORDER BY name;**

Lesson 8		Урок 8. SQL-команды DELETE и UPDATE
№	Действие	Команда на языке SQL и скрин результата
1	Изменить name Шляпы на Летние головные уборы по ID (Было ID 5 Шляпы, станет ID 5 Летние головные уборы)	USE shop;  UPDATE category SET name = 'Летние головные уборы' WHERE id = 5;

Result Grid | Filter Rows: Search | Edit:

id	name	discount	alias_name
1	Женская одежда	10	womans' wear
2	Мужская одежда	20	NULL
3	Детская одежда	0	NULL
4	Головные уборы	5	hats
5	Летние головные уборы	0	NULL
6	Женское нижнее белье	7	NULL
7	Мужское нижнее белье	6	NULL
8	Мужская обувь	0	NULL
9	Женская обувь	33	NULL
NULL	NULL	NULL	NULL

2	В ID: 3.Детская одежда, 5. Летние уборы, 8. Мужская обувь - **изменим скидку на 3%**	 **USE shop;** **UPDATE category SET discount = 3 WHERE id IN (3 , 5, 8);**

3	**Удалить** категорию Женская Обувь по ID 9	 **USE shop;** **DELETE FROM category WHERE id = 9;**

9 lesson	Урок 9. Согласованность данных

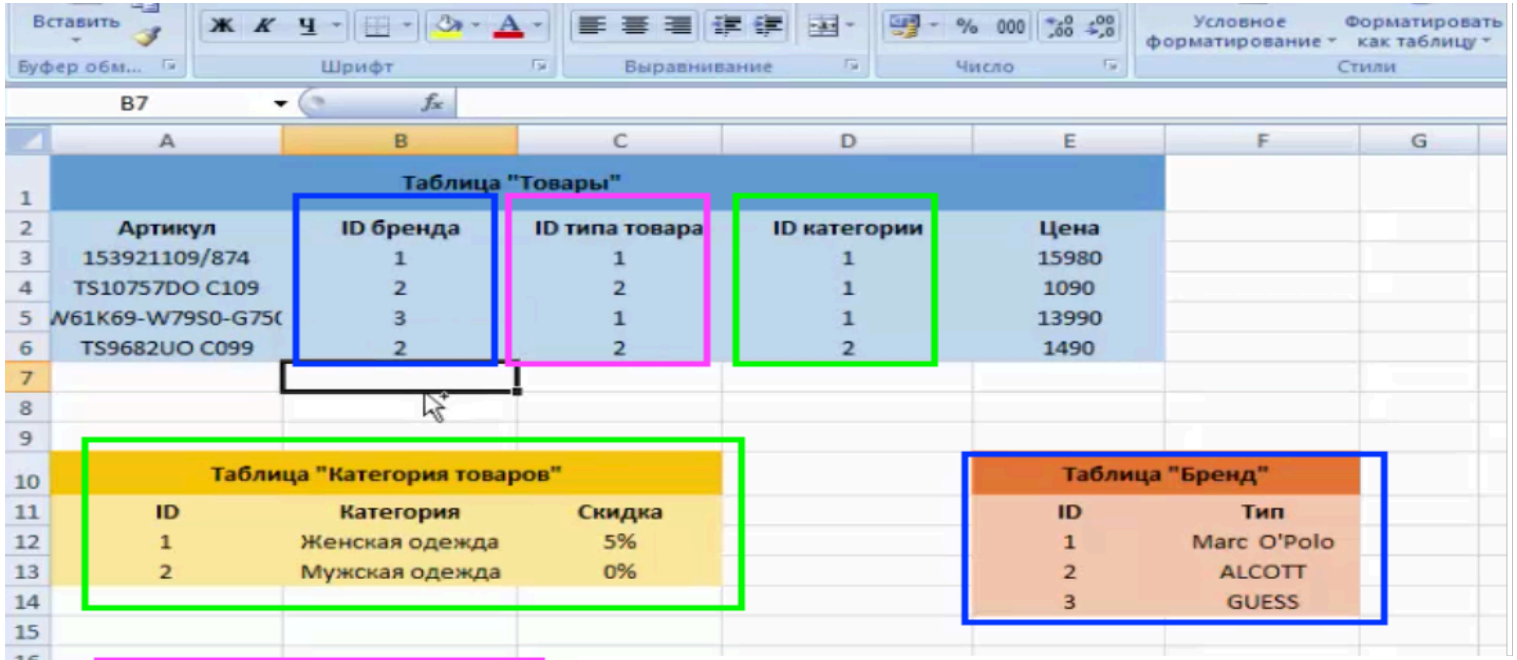


Таблица "Товары"

Артикул	ID бренда	ID типа товара	ID категории	Цена
153921109/874	1	1	1	15980
TS10757DO C109	2	2	1	1090
W61K69-W79S0-G75(	3	1	1	13990
TS9682UO C099	2	2	2	1490

Таблица "Категория товаров"

ID	Категория	Скидка
1	Женская одежда	5%
2	Мужская одежда	0%

Таблица "Бренд"

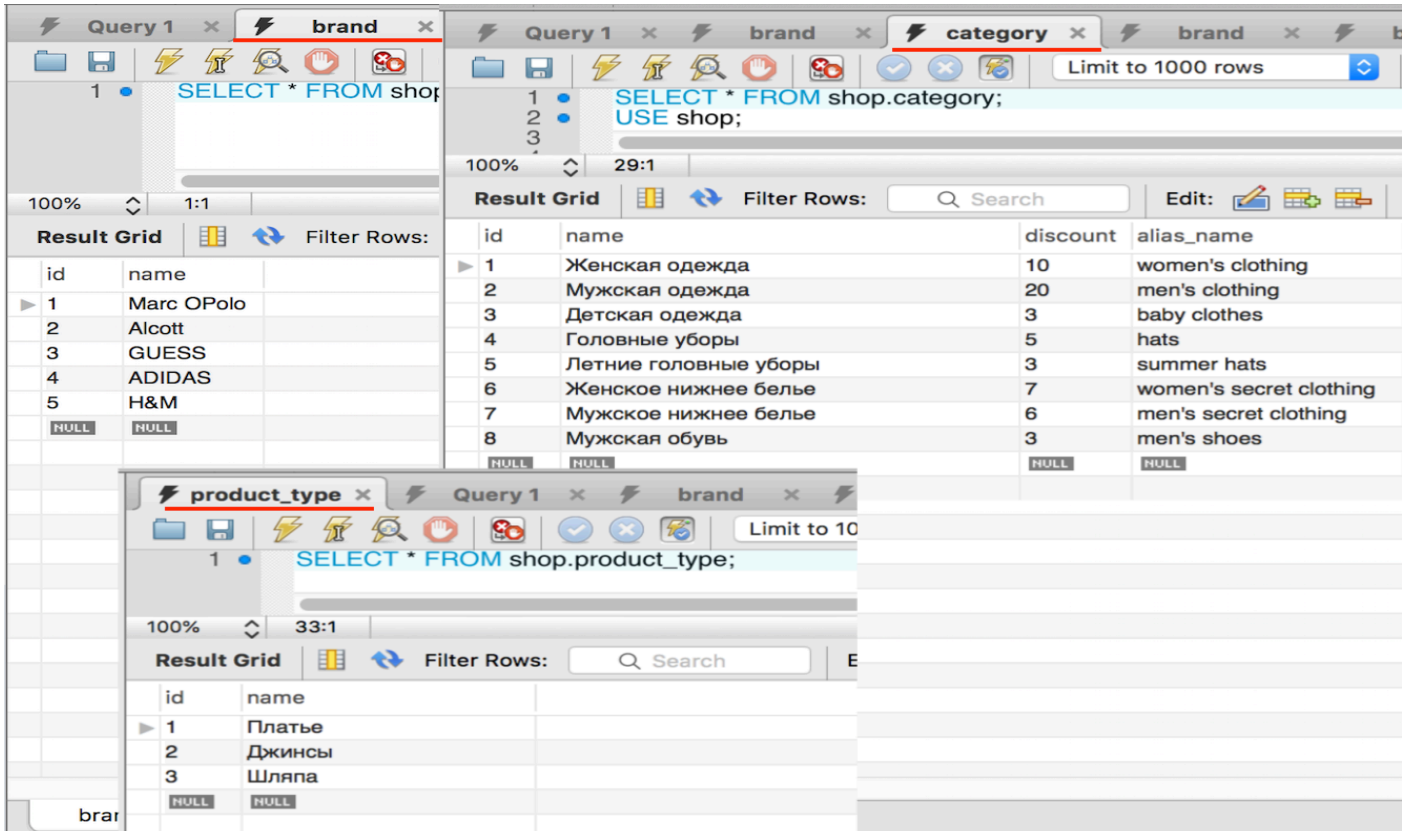
ID	Тип
1	Marc O'Polo
2	ALCOTT
3	GUESS

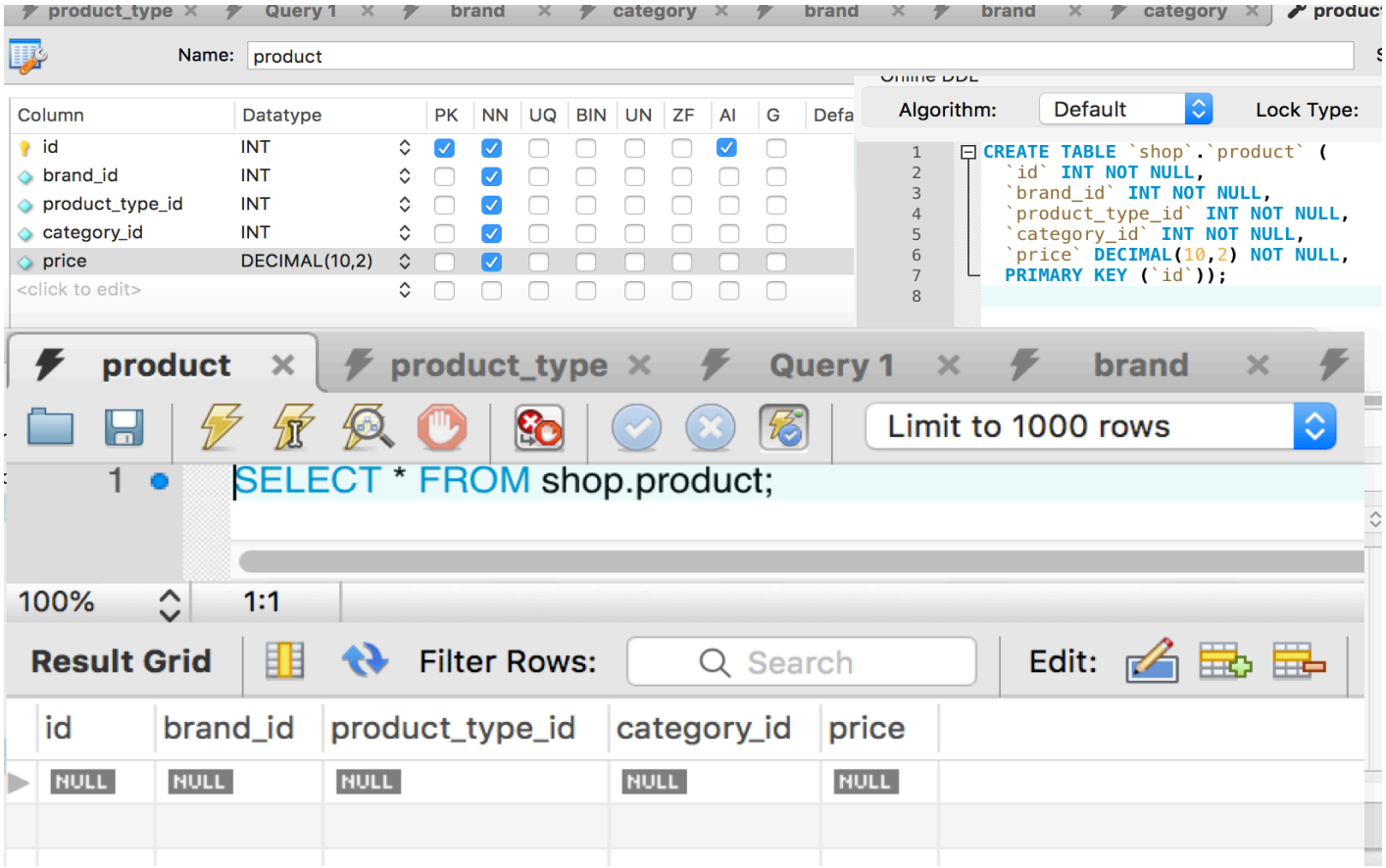
Таблица "Тип товара"

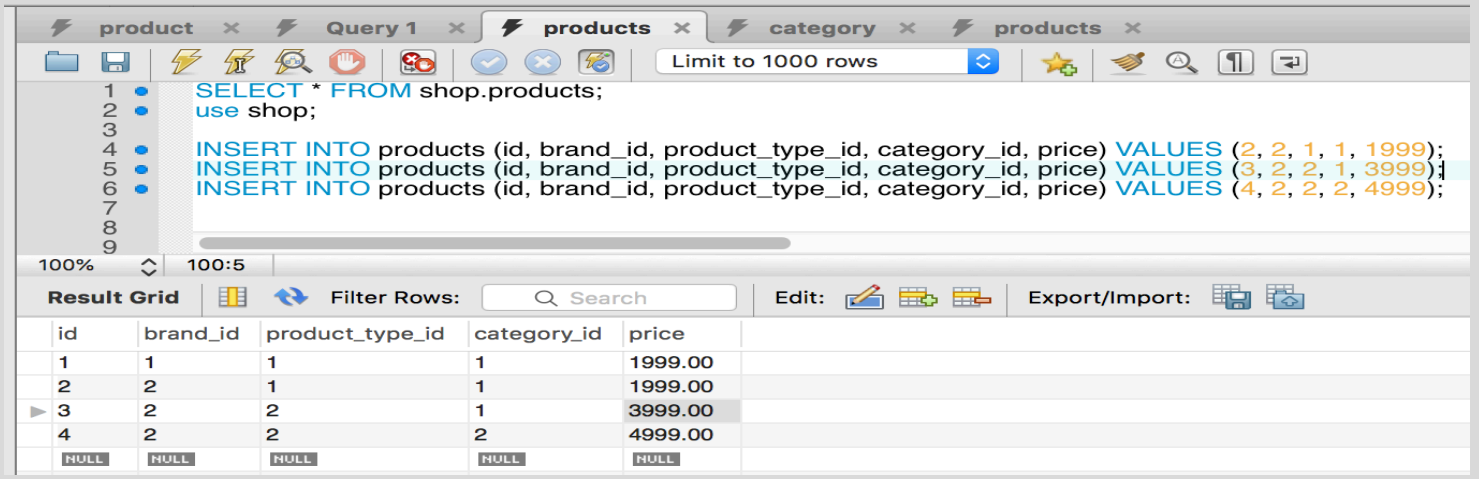
ID	Тип
1	Платье
2	Футболка

Консистентность данных

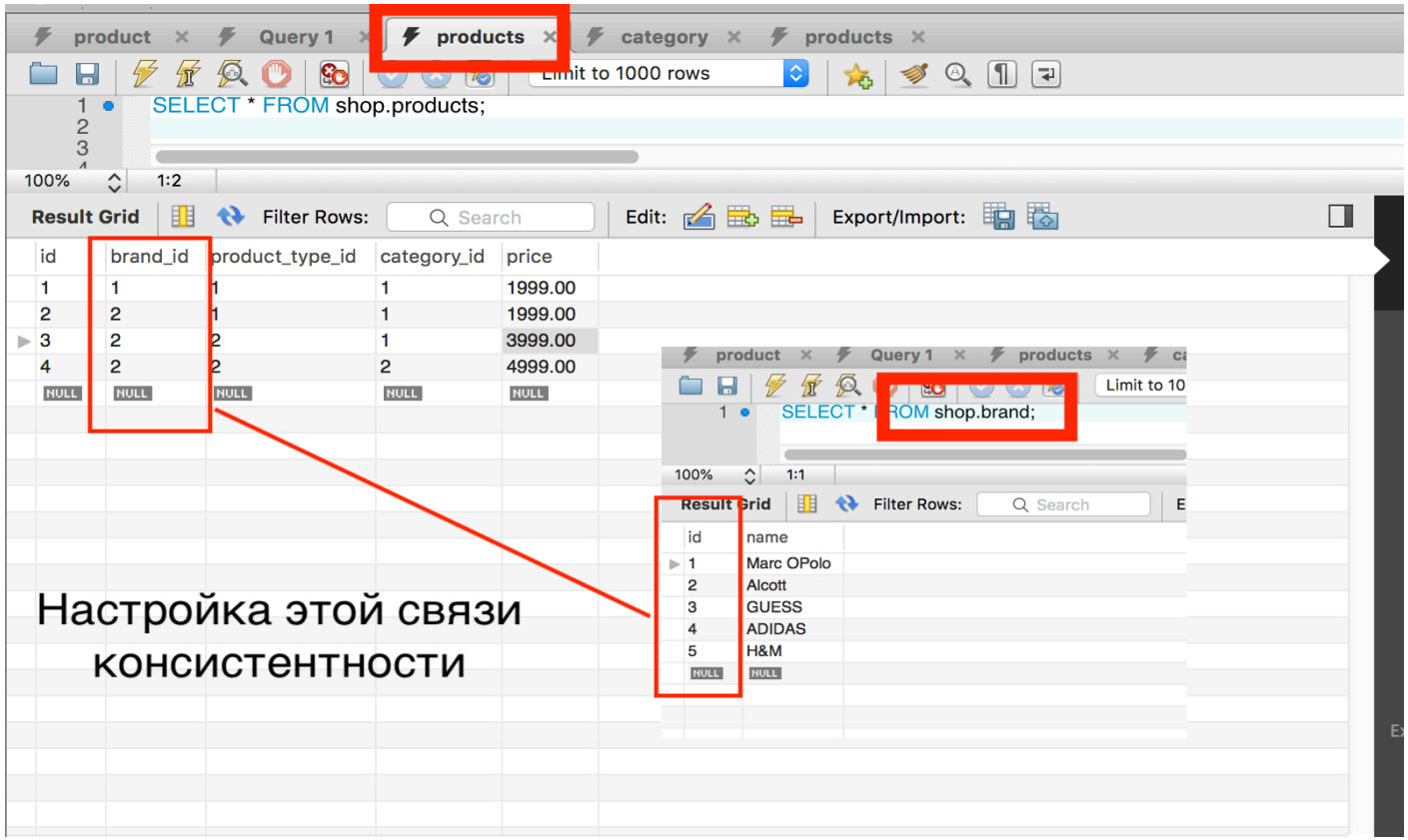
Консистентность данных (англ. *data consistency* или *data validity*) — это согласованность данных друг с другом, целостность данных, а также внутренняя непротиворечивость. Множество всех условий, налагаемых на данные определяется моделью

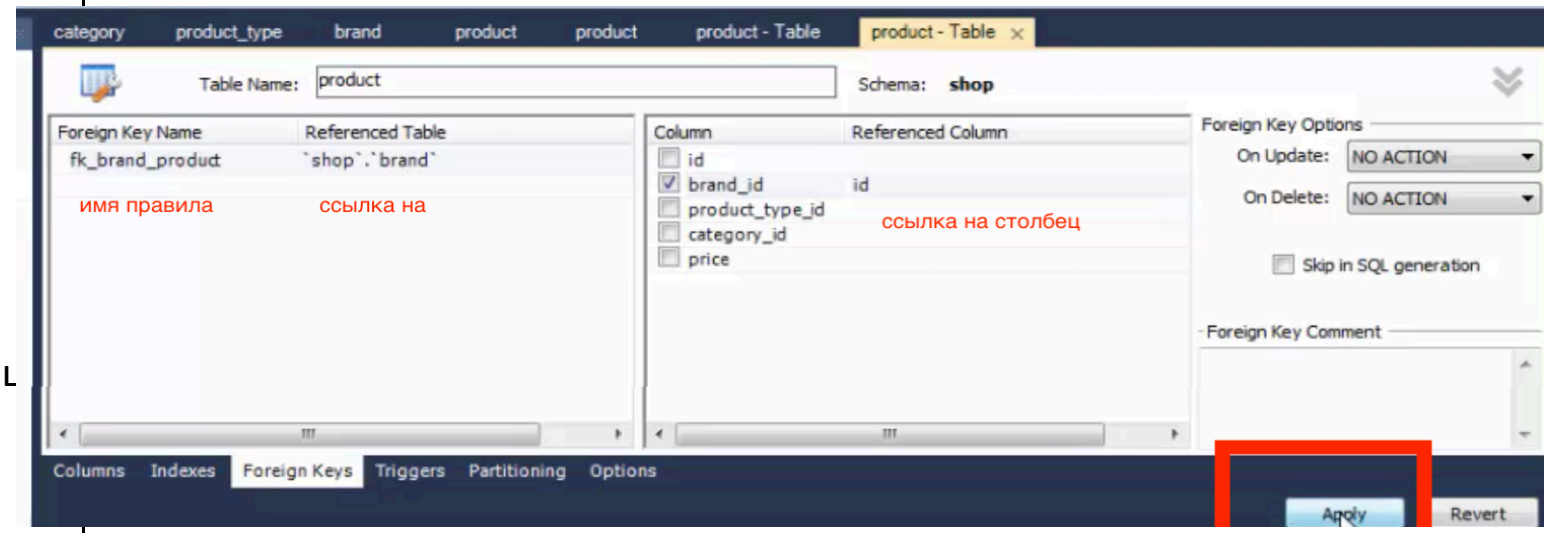

Query 1
brand
1 SELECT * FROM shop
100%
1:1
Result Grid
Filter Rows:
id
name
1
Marc OPolo
2
Alcott
3
GUESS
4
ADIDAS
5
H&M
NULL
NULL
Query 1
brand
category
brand
Limit to 1000 rows
1 SELECT * FROM shop.category;
2 USE shop;
3
100%
29:1
Result Grid
Filter Rows:
Search
Edit:
id
name
discount
alias_name
1
Женская одежда
10
women's clothing
2
Мужская одежда
20
men's clothing
3
Детская одежда
3
baby clothes
4
Головные уборы
5
hats
5
Летние головные уборы
3
summer hats
6
Женское нижнее белье
7
women's secret clothing
7
Мужское нижнее белье
6
men's secret clothing
8
Мужская обувь
3
men's shoes
NULL
NULL
NULL
NULL
product_type
Query 1
brand
Limit to 10
1 SELECT * FROM shop.product_type;
100%
33:1
Result Grid
Filter Rows:
Search
id
name
1
Платье
2
Джинсы
3
Шляпа
NULL
NULL

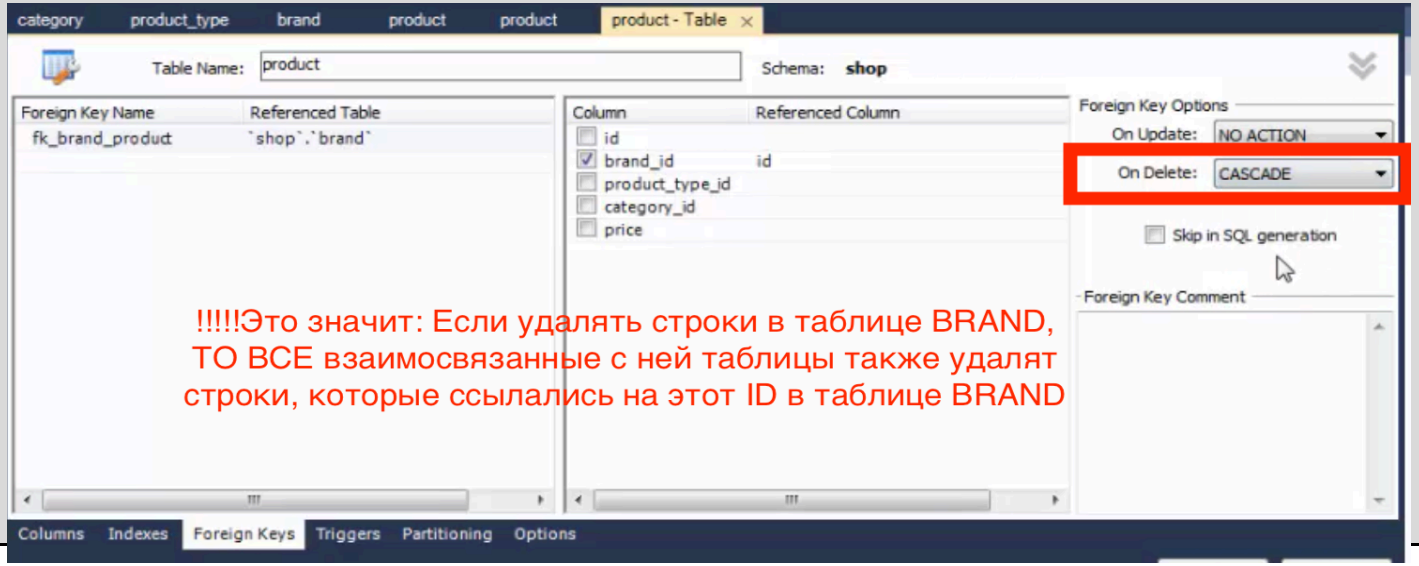
№	Действие	Команда на языке SQL и скрин результата
1	Создать таблицу Products	

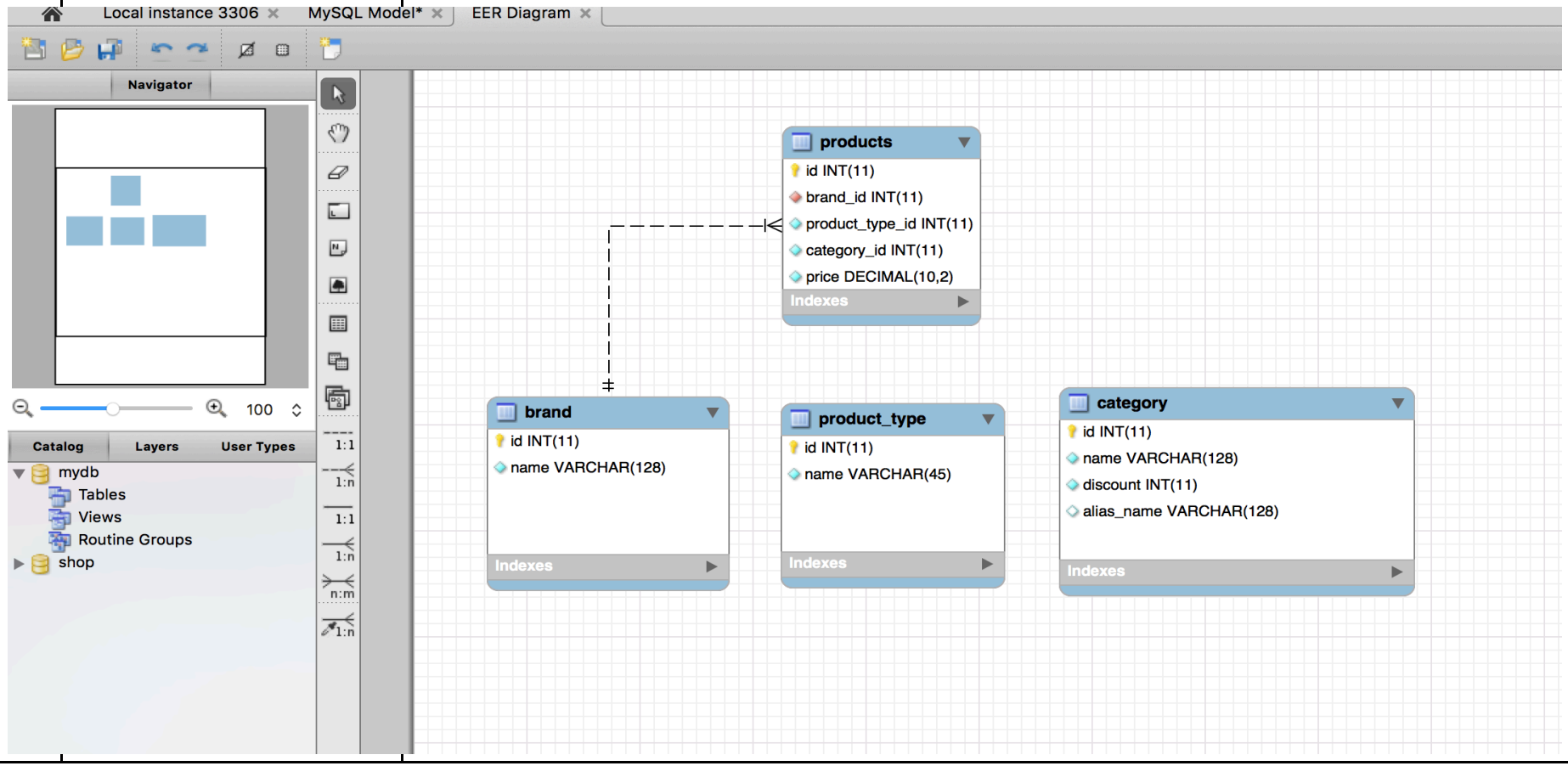
2	Заполнить таблицу Products	 **use shop;** **INSERT INTO products (id, brand_id, product_type_id, category_id, price) VALUES (2, 2, 1, 1, 1999);** **INSERT INTO products (id, brand_id, product_type_id, category_id, price) VALUES (3, 2, 2, 1, 3999);** **INSERT INTO products (id, brand_id, product_type_id, category_id, price) VALUES (4, 2, 2, 2, 4999);**

10 lesson		**УРОК 10. Настройка согласованности данных ВНЕШНИЙ КЛЮЧ - FOREIGN KEY constraints**

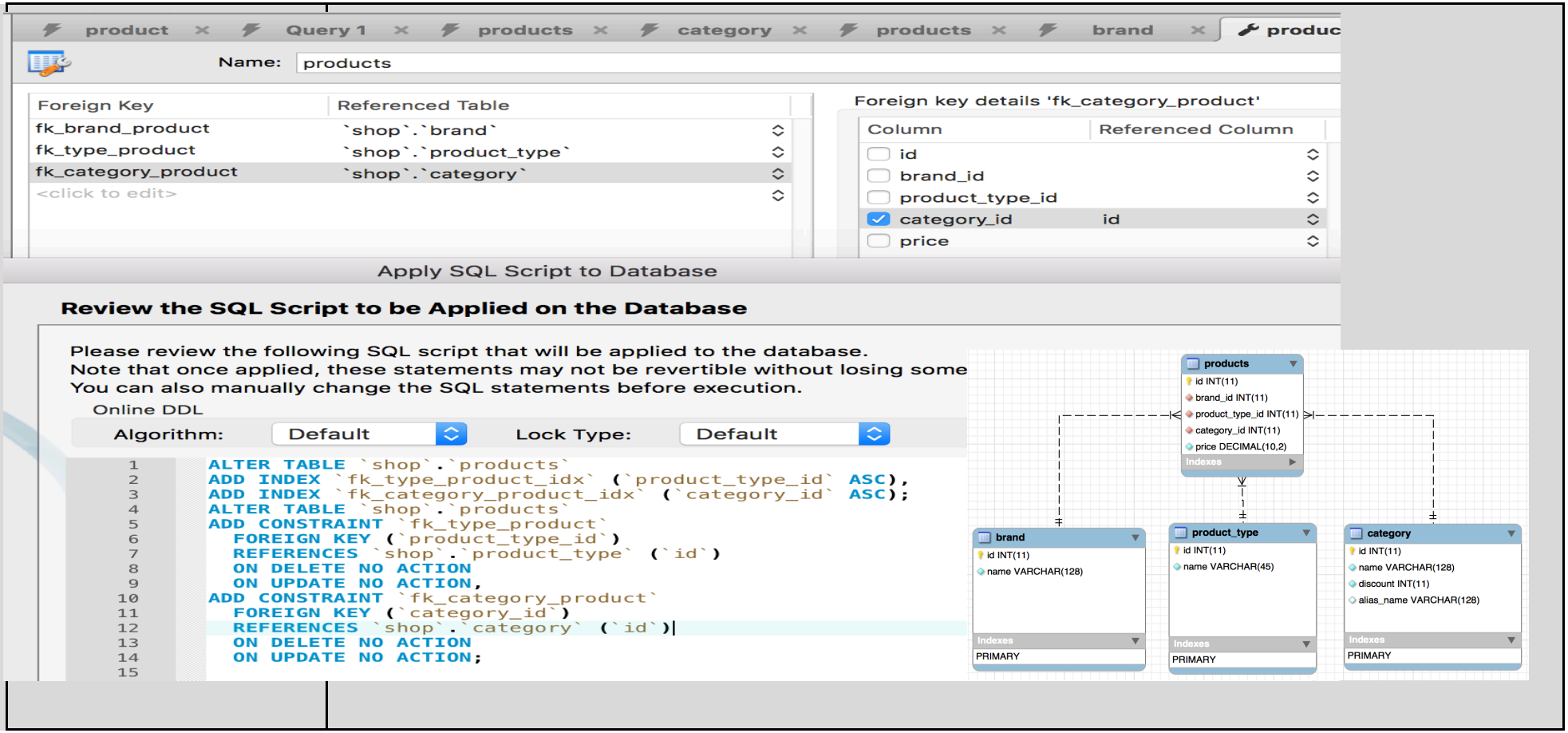


1	Настройка связи между столбцом brand id в таб. Products со столбц id в таб. Brand	 **shop - products (прав.кл. мыши) - Alter Table - вкладка внизу Foreign Keys -**

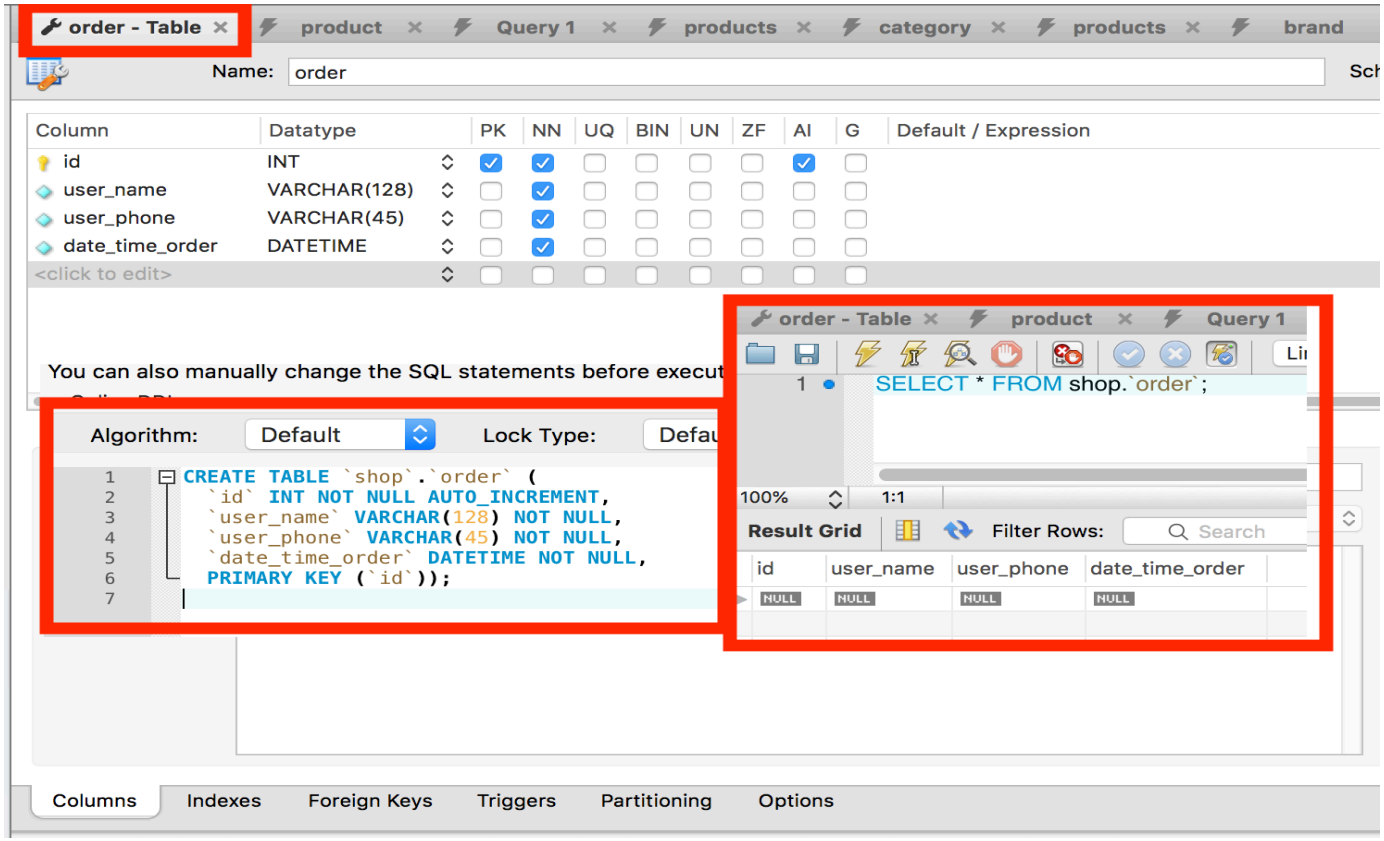
2	!!!!!!!Настройка опциональности	
3	Визуализация связей	**Основное меню программы - Database - Reverse Engineer - (выбираем нашу базу данных) будет такой результат. Где видны связи между таблицами**

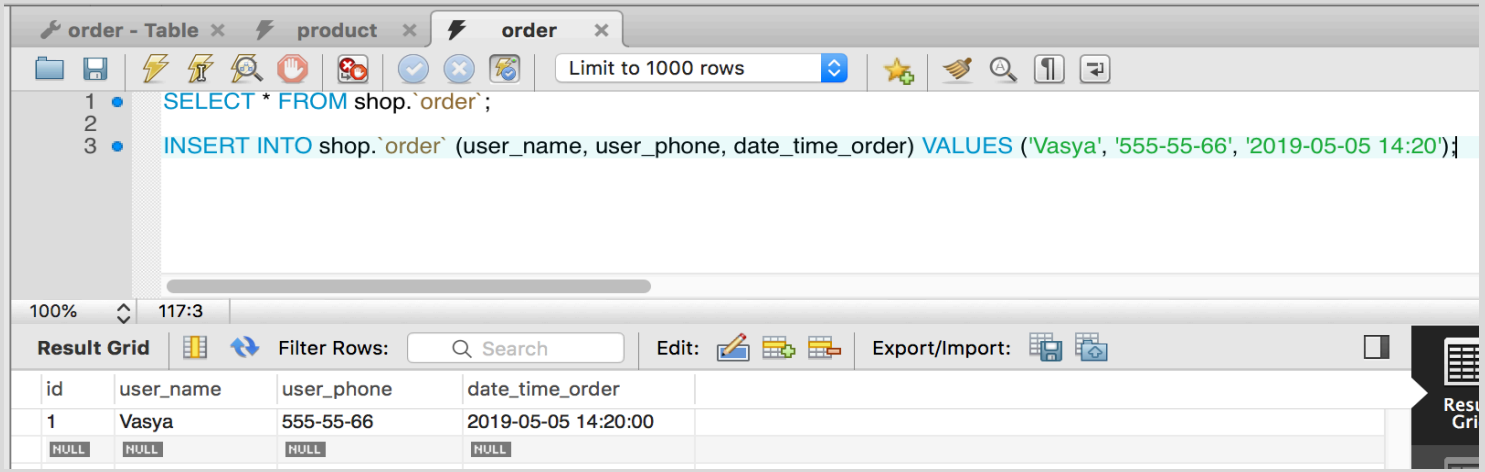

Local instance 3306
MySQL Model*
EER Diagram
Navigator
100
Catalog
Layers
User Types
mydb
Tables
Views
Routine Groups
shop
1:1
1:n
1:1
1:n
n:m
1:n
products
id INT(11)
brand_id INT(11)
product_type_id INT(11)
category_id INT(11)
price DECIMAL(10,2)
Indexes
brand
id INT(11)
name VARCHAR(128)
Indexes
product_type
id INT(11)
name VARCHAR(45)
Indexes
category
id INT(11)
name VARCHAR(128)
discount INT(11)
alias_name VARCHAR(128)
Indexes

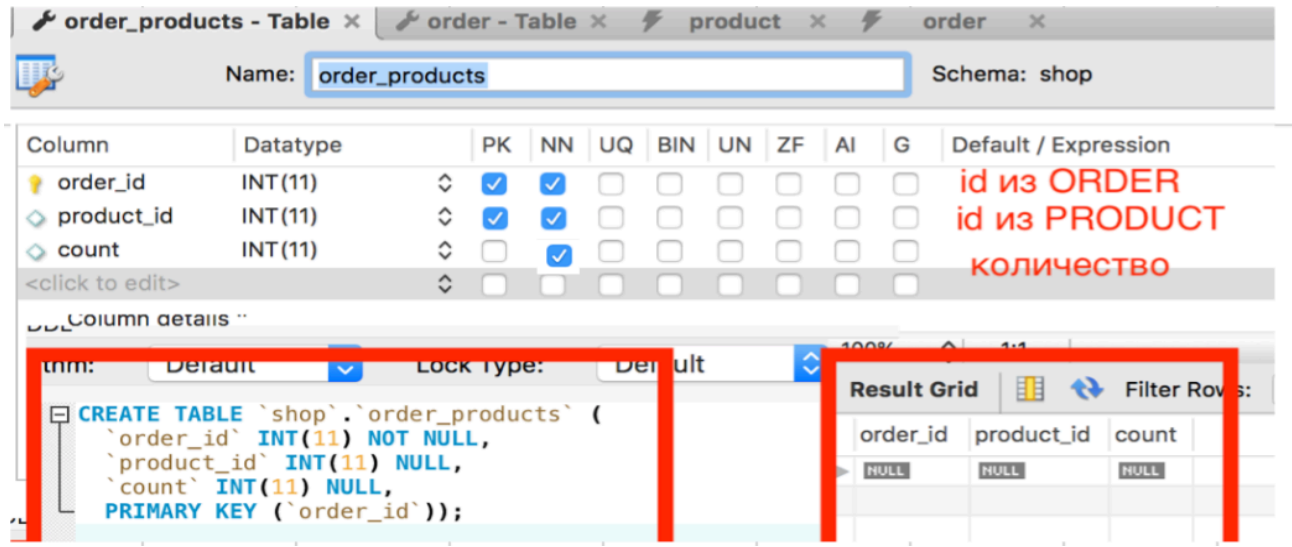
4	Настройка связей между таб. Products с таблицами type, category	ALTER TABLE `shop`.`products` ADD INDEX `fk_type_product_idx` (`product_type_id` ASC), ADD INDEX `fk_category_product_idx` (`category_id` ASC); ALTER TABLE `shop`.`products` ADD CONSTRAINT `fk_type_product` FOREIGN KEY (`product_type_id`) REFERENCES `shop`.`product_type` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION, ADD CONSTRAINT `fk_category_product` FOREIGN KEY (`category_id`) REFERENCES `shop`.`category` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION;

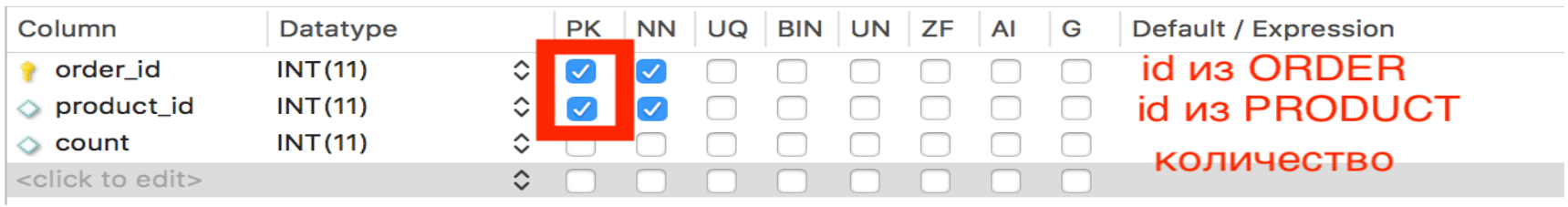

product
Query 1
products
category
products
brand
Name: products
Foreign Key
Referenced Table
fk_brand_product
`shop`.`brand`
fk_type_product
`shop`.`product_type`
fk_category_product
`shop`.`category`
<click to edit>
Foreign key details 'fk_category_product'
Column
Referenced Column
id
brand_id
product_type_id
category_id
id
price
Apply SQL Script to Database
Review the SQL Script to be Applied on the Database
Please review the following SQL script that will be applied to the database.
Note that once applied, these statements may not be revertible without losing some
You can also manually change the SQL statements before execution.
Online DDL
Algorithm: Default
Lock Type: Default
1 ALTER TABLE `shop`.`products`
2 ADD INDEX `fk_type_product_idx` (`product_type_id` ASC),
3 ADD INDEX `fk_category_product_idx` (`category_id` ASC);
4 ALTER TABLE `shop`.`products`
5 ADD CONSTRAINT `fk_type_product`
6 FOREIGN KEY (`product_type_id`)
7 REFERENCES `shop`.`product_type` (`id`)
8 ON DELETE NO ACTION
9 ON UPDATE NO ACTION,
10 ADD CONSTRAINT `fk_category_product`
11 FOREIGN KEY (`category_id`)
12 REFERENCES `shop`.`category` (`id`)
13 ON DELETE NO ACTION
14 ON UPDATE NO ACTION;
15
products
id INT(11)
brand_id INT(11)
product_type_id INT(11)
category_id INT(11)
price DECIMAL(10,2)
Indexes
brand
id INT(11)
name VARCHAR(128)
Indexes
PRIMARY
product_type
id INT(11)
name VARCHAR(45)
Indexes
PRIMARY
category
id INT(11)
name VARCHAR(128)
discount INT(11)
alias_name VARCHAR(128)
Indexes
PRIMARY

11 lesson		**Урок 11. Создание таблиц с отношением “многие ко многим”. Создаем таблицу ORDER для хранения заказа в интернет магазине, ORDER_PRODUCTS - связывающая таблица заказов из ORDER с товарами из PRODUCTS**
1	Создаем **таблицу ЗАКАЗОВ ORDER** в которой будет храниться информация о заказе	**CREATE TABLE `shop`.`order` (** **`id` INT NOT NULL AUTO_INCREMENT,** **`user_name` VARCHAR(128) NOT NULL,** **`user_phone` VARCHAR(45) NOT NULL,** **`date_time_order` DATETIME NOT NULL,** **PRIMARY KEY (`id`));**

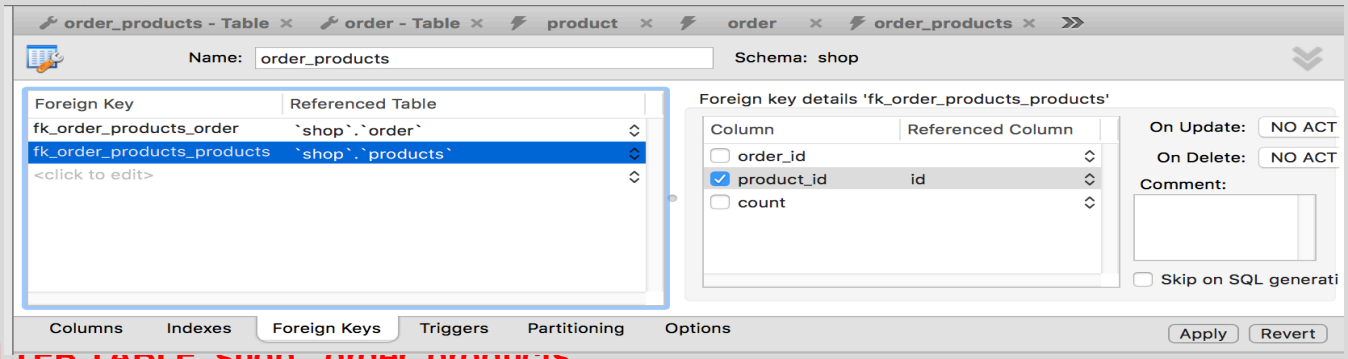

order - Table
product
Query 1
products
category
products
brand
Name: order
Column
Datatype
PK
NN
UQ
BIN
UN
ZF
AI
G
Default / Expression
id
INT
user_name
VARCHAR(128)
user_phone
VARCHAR(45)
date_time_order
DATETIME
<click to edit>
You can also manually change the SQL statements before execut
Algorithm:
Default
Lock Type:
Defau
1 CREATE TABLE `shop`.`order` (
2 `id` INT NOT NULL AUTO_INCREMENT,
3 `user_name` VARCHAR(128) NOT NULL,
4 `user_phone` VARCHAR(45) NOT NULL,
5 `date_time_order` DATETIME NOT NULL,
6 PRIMARY KEY (`id`));
7
order - Table
product
Query 1
1 SELECT * FROM shop.`order`;
100%
1:1
Result Grid
Filter Rows:
Search
id
user_name
user_phone
date_time_order
NULL
NULL
NULL
NULL
Columns
Indexes
Foreign Keys
Triggers
Partitioning
Options

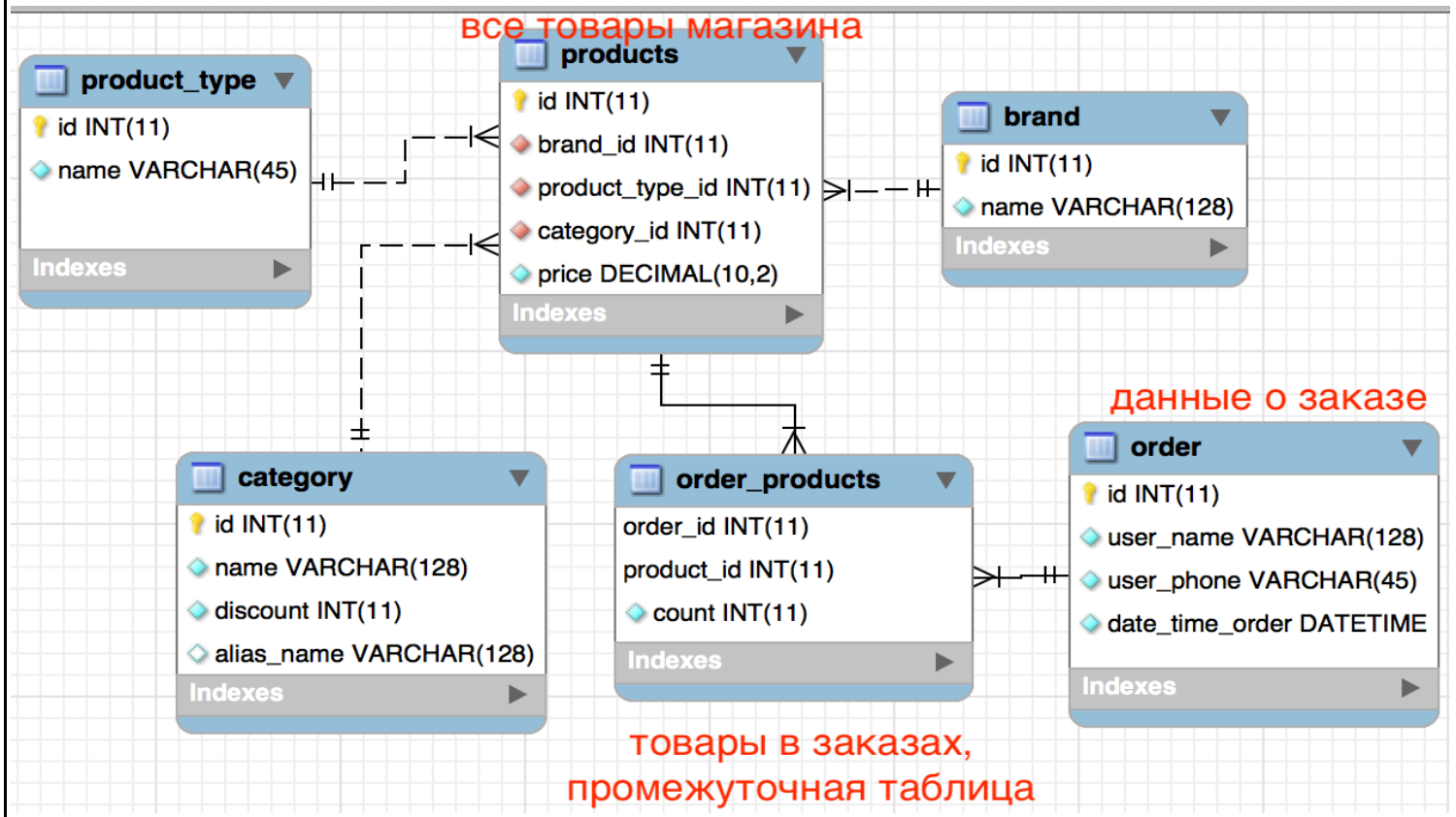
2	Заполняем ORDER. В нем храним данные по user и НЕ храним данные по выбранному товару	 **INSERT INTO shop.`order` (user_name, user_phone, date_time_order) VALUES ('Vasya', '555-55-66', '2019-05-05 14:20');**

3	Создаем **таблицу взаимосвязи заказов и товаров ORDER_PRODUCTS** в которой будет храниться информация о выбранном в заказе товаре user-ом	 **CREATE TABLE `shop`.`order_products` (`order_id` INT(11) NOT NULL, `product_id` INT(11) NULL, `count` INT(11) NULL, PRIMARY KEY (`order_id`));**

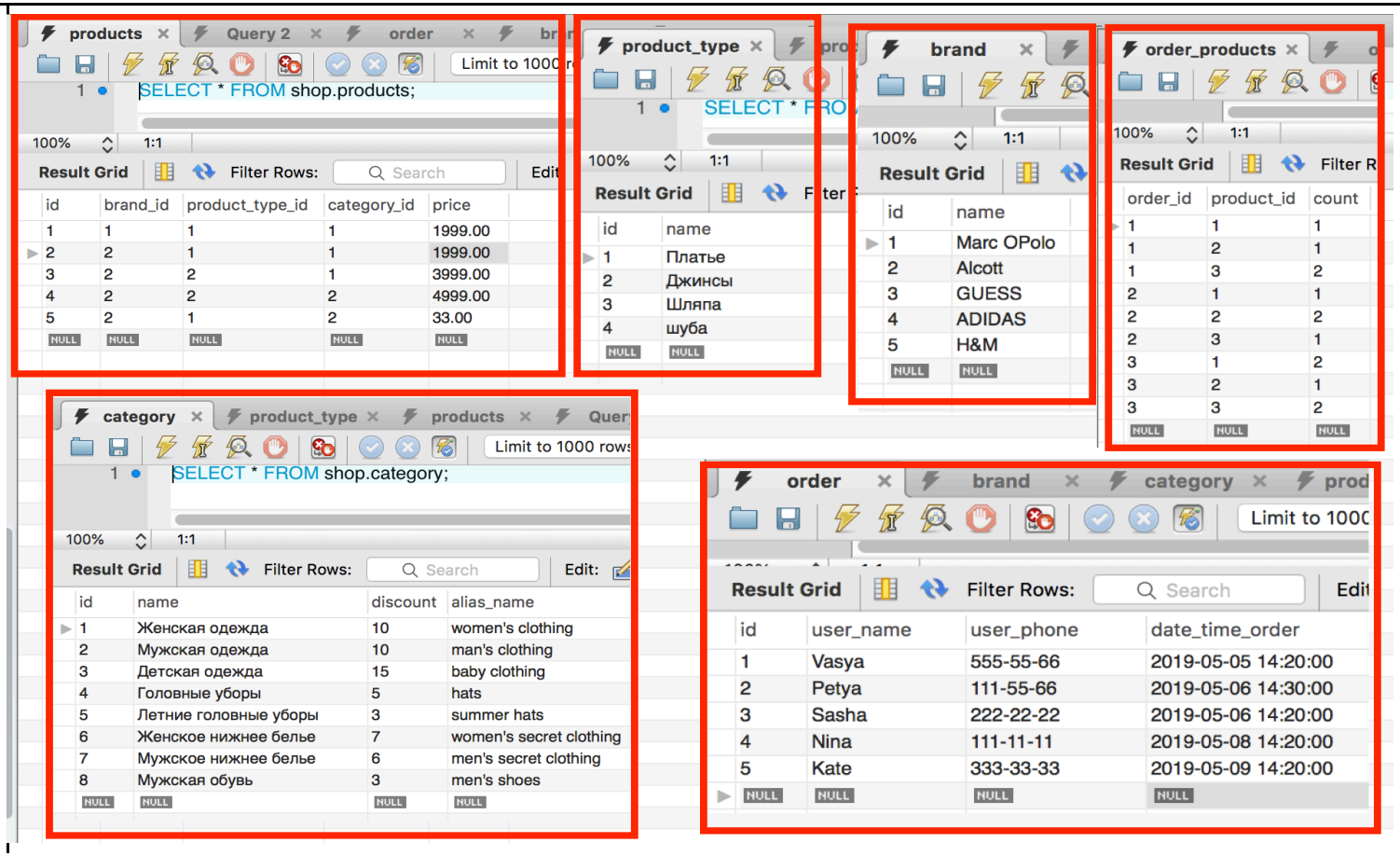


Составной первичный ключ, состоит из двух и более столбцов. Это значит что в таблице не может быть две строки с одинаковым ORDER_ID и PRODUCT_ID

12 lesson		**Урок 12. Составной первичный ключ. Создаем таблицу ORDER_PRODUCTS - связывающая таблица заказов из ORDER с товарами из PRODUCTS**
4	Настройка связей в таблице ORDER_PRODUCTS устанавливаем ключи и связываем ее с PRODUCTS и ORDER (настройка внешних ключей foreign keys)	 **ALTER TABLE `shop`.`order_products`** **ADD INDEX `fk_order_products_products_idx` (`product_id` ASC);** **ALTER TABLE `shop`.`order_products`** **ADD CONSTRAINT `fk_order_products_order`** **FOREIGN KEY (`order_id`)** **REFERENCES `shop`.`order` (`id`)** **ON DELETE NO ACTION** **ON UPDATE NO ACTION,** **ADD CONSTRAINT `fk_order_products_products`** **FOREIGN KEY (`product_id`)** **REFERENCES `shop`.`products` (`id`)** **ON DELETE NO ACTION** **ON UPDATE NO ACTION;**

5	Визуализируем связи	**Основное меню программы - Database - Reverse Engineer - (выбираем нашу базу данных) будет такой результат. Где видны связи между таблицами**
		

My Database


products
Query 2
order
SELECT * FROM shop.products;
Result Grid
Filter Rows:
Search
id brand_id product_type_id category_id price
1 1 1 1 1999.00
2 2 1 1 1999.00
3 2 2 1 3999.00
4 2 2 2 4999.00
5 2 1 2 33.00
product_type
SELECT * FROM shop.product_type;
id name
1 Платье
2 Джинсы
3 Шляпа
4 шуба
brand
SELECT * FROM shop.brand;
id name
1 Marc OPolo
2 Alcott
3 GUESS
4 ADIDAS
5 H&M
order_products
SELECT * FROM shop.order_products;
order_id product_id count
1 1 1
1 2 1
1 3 2
2 1 1
2 2 2
2 3 1
3 1 2
3 2 1
3 3 2
category
product_type
products
SELECT * FROM shop.category;
id name discount alias_name
1 Женская одежда 10 women's clothing
2 Мужская одежда 10 man's clothing
3 Детская одежда 15 baby clothing
4 Головные уборы 5 hats
5 Летние головные уборы 3 summer hats
6 Женское нижнее белье 7 women's secret clothing
7 Мужское нижнее белье 6 men's secret clothing
8 Мужская обувь 3 men's shoes
order
brand
category
SELECT * FROM shop.order;
id user_name user_phone date_time_order
1 Vasya 555-55-66 2019-05-05 14:20:00
2 Petya 111-55-66 2019-05-06 14:30:00
3 Sasha 222-22-22 2019-05-06 14:20:00
4 Nina 111-11-11 2019-05-08 14:20:00
5 Kate 333-33-33 2019-05-09 14:20:00

13 lesson		Урок 13. Объединение данных из нескольких таблиц (INNER JOIN, LEFT JOIN \ RIGHT JOIN, FULL OUTER JOIN)
		в лекции немного теории\логики на примере xls

A14 | 3

Таблица "Товары"

Артикул	ID бренда	ID типа товара	ID категории	Цена
153921109/874	1	1	1	15980
TS10757DO C109	2	2	1	1090
W61K69-W79S0-G75(	3	1	1	13990
TS9682UO C099	2	2	2	1490

ID	Категория	Скидка
1	Женская одежда	5%
1	Женская одежда	5%
1	Женская одежда	5%
2	Мужская одежда	0%

Таблица "Категория товаров"

ID	Категория	Скидка
1	Женская одежда	5%
2	Мужская одежда	0%
3	Женская обувь	15%

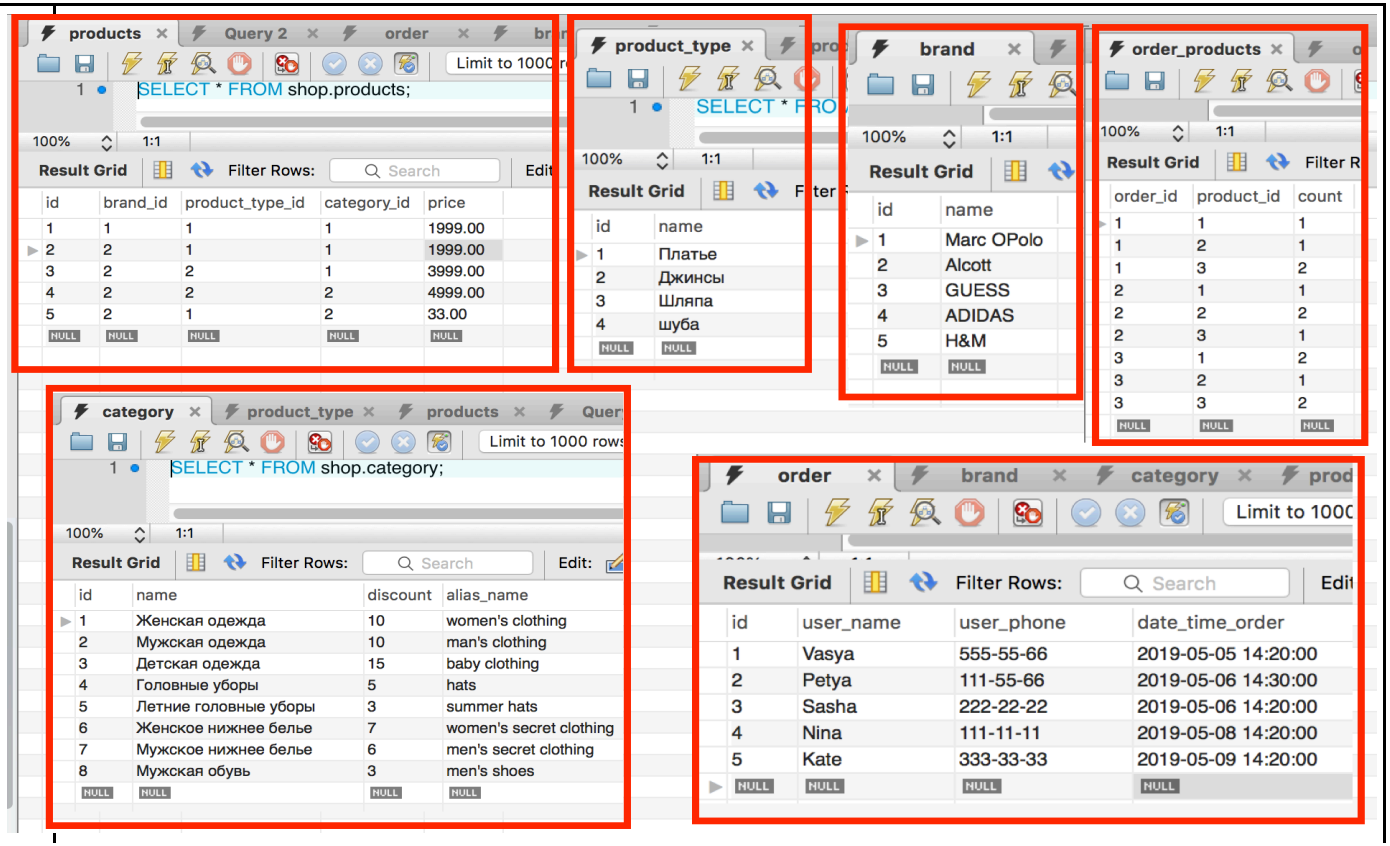
Таблица "Бренд"

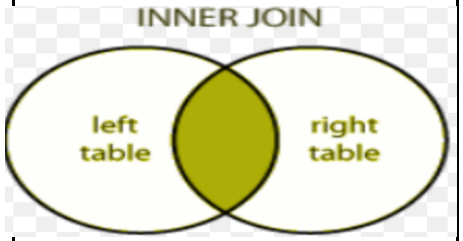
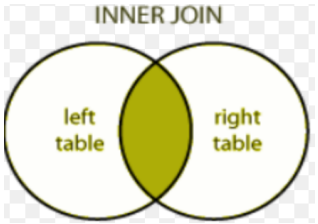
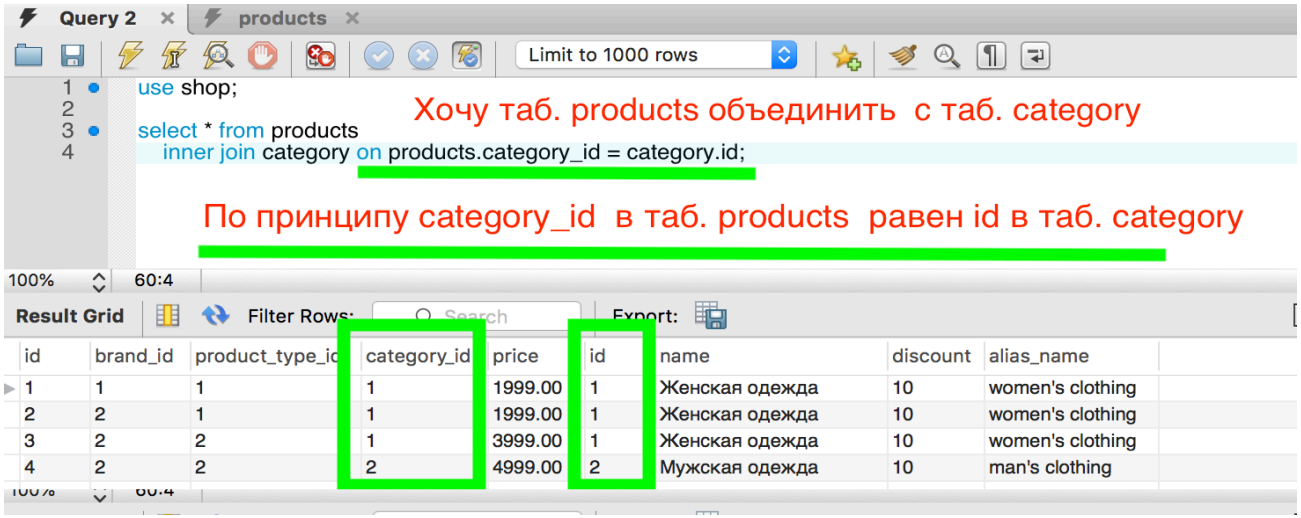
ID	Тип
1	Marc O'Polo
2	ALCOTT
3	GUESS

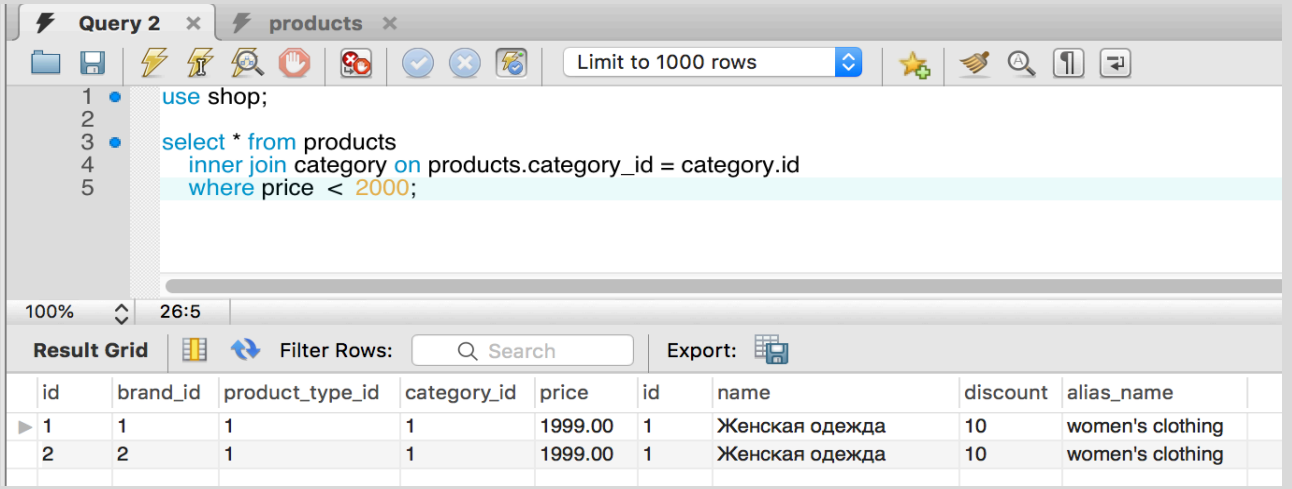
Таблица "Тип товара"

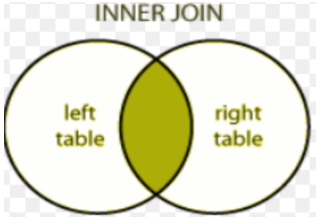
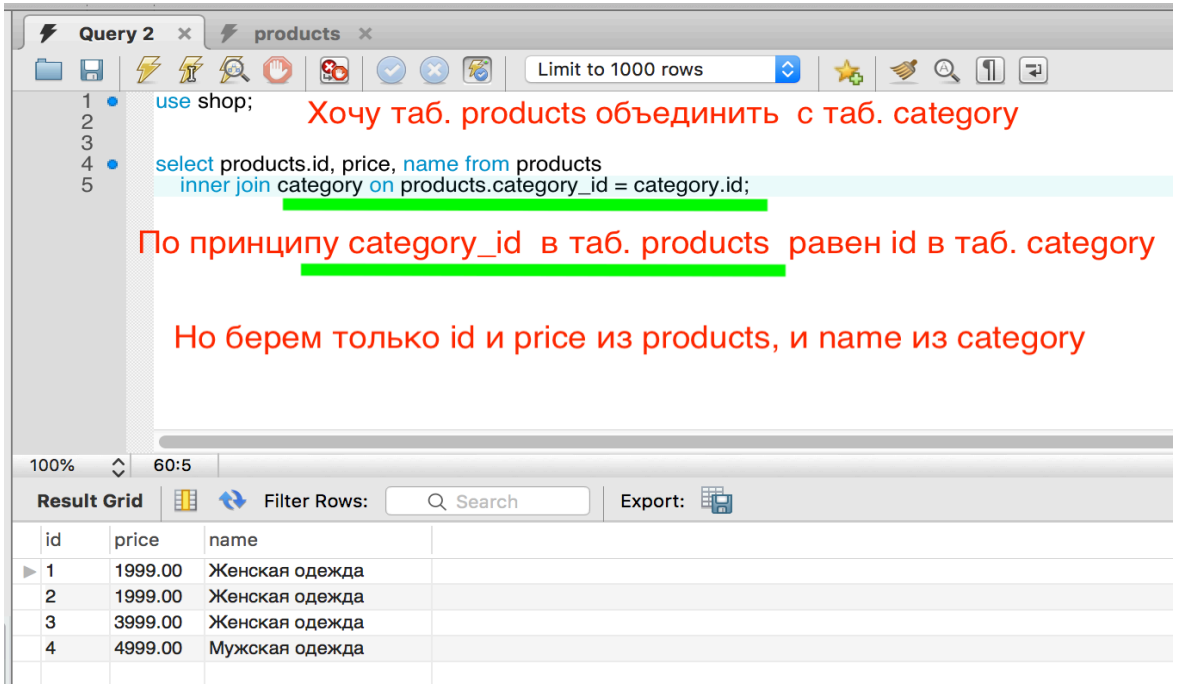
ID	Тип
1	Платье
2	Футболка

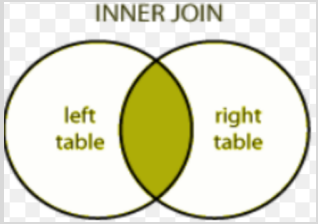
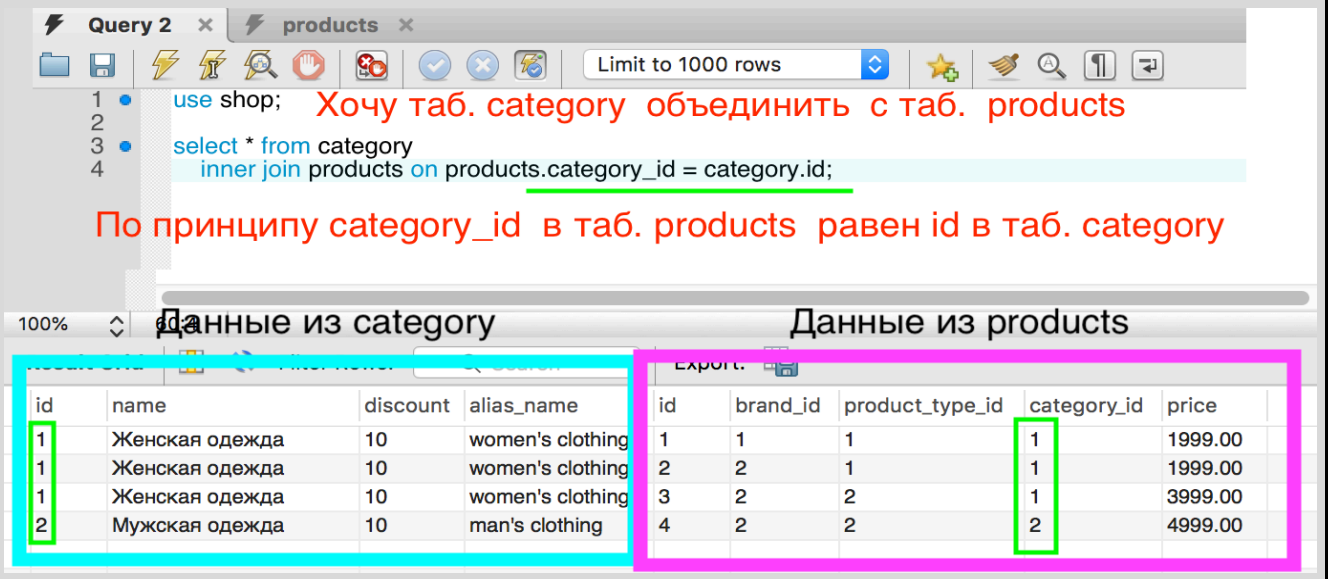
My Database


products
Query 2
order
SELECT * FROM shop.products;
Result Grid
Filter Rows:
Search
id brand_id product_type_id category_id price
1 1 1 1 1999.00
2 2 1 1 1999.00
3 2 2 1 3999.00
4 2 2 2 4999.00
5 2 1 2 33.00
product_type
SELECT * FROM shop.product_type;
id name
1 Платье
2 Джинсы
3 Шляпа
4 шуба
brand
SELECT * FROM shop.brand;
id name
1 Marc OPolo
2 Alcott
3 GUESS
4 ADIDAS
5 H&M
order_products
SELECT * FROM shop.order_products;
order_id product_id count
1 1 1
1 2 1
1 3 2
2 1 1
2 2 2
2 3 1
3 1 2
3 2 1
3 3 2
category
product_type
products
SELECT * FROM shop.category;
id name discount alias_name
1 Женская одежда 10 women's clothing
2 Мужская одежда 10 man's clothing
3 Детская одежда 15 baby clothing
4 Головные уборы 5 hats
5 Летние головные уборы 3 summer hats
6 Женское нижнее белье 7 women's secret clothing
7 Мужское нижнее белье 6 men's secret clothing
8 Мужская обувь 3 men's shoes
order
brand
category
SELECT * FROM shop.order;
id user_name user_phone date_time_order
1 Vasya 555-55-66 2019-05-05 14:20:00
2 Petya 111-55-66 2019-05-06 14:30:00
3 Sasha 222-22-22 2019-05-06 14:20:00
4 Nina 111-11-11 2019-05-08 14:20:00
5 Kate 333-33-33 2019-05-09 14:20:00

14 lesson		Урок 14. Объединение данных из нескольких таблиц (INNER JOIN)
№	Действие	Команда на языке SQL и скрин результата
		**ОПЕРАТОР INNER JOIN -Отображает то, что является пересичением двух таблиц по условию описанному в ON**
1	Объединение данных в таблице PRODUCTS с данными из CATEGORY. Знак * - означает вывести ВСЕ ДАННЫЕ 	 **use shop;** **select * from products** **inner join category on products.category_id = category.id;**

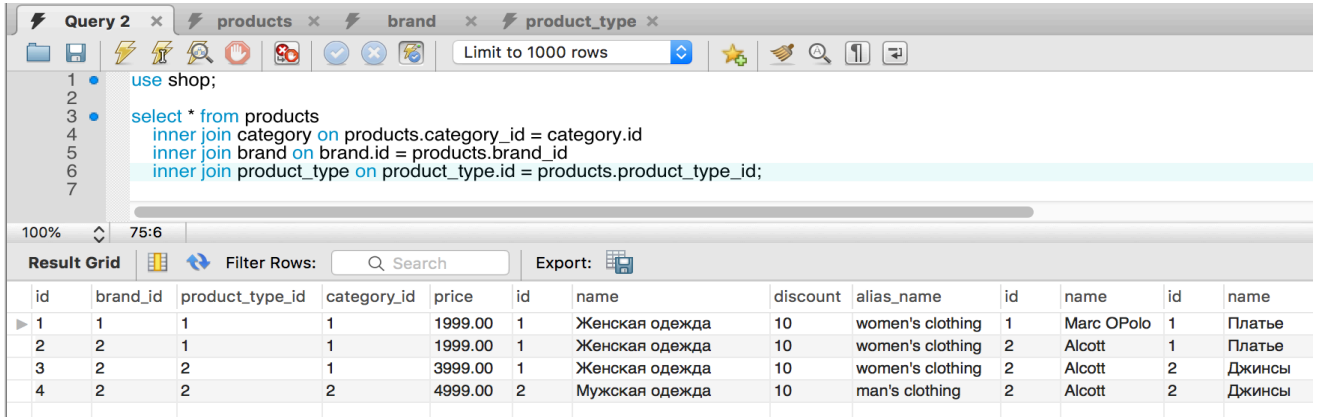
4	Объединение данных в таблице PRODUCTS с данными из CATEGORY. Знак * - означает вывести ВСЕ ДАННЫЕ НО!!!!!! Только те, где price < 2000	 **use shop;** **select * from products** **inner join category on products.category_id = category.id** **where price < 2000;**

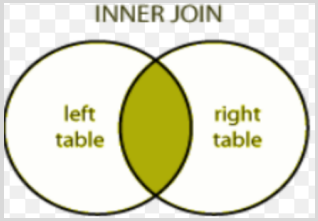
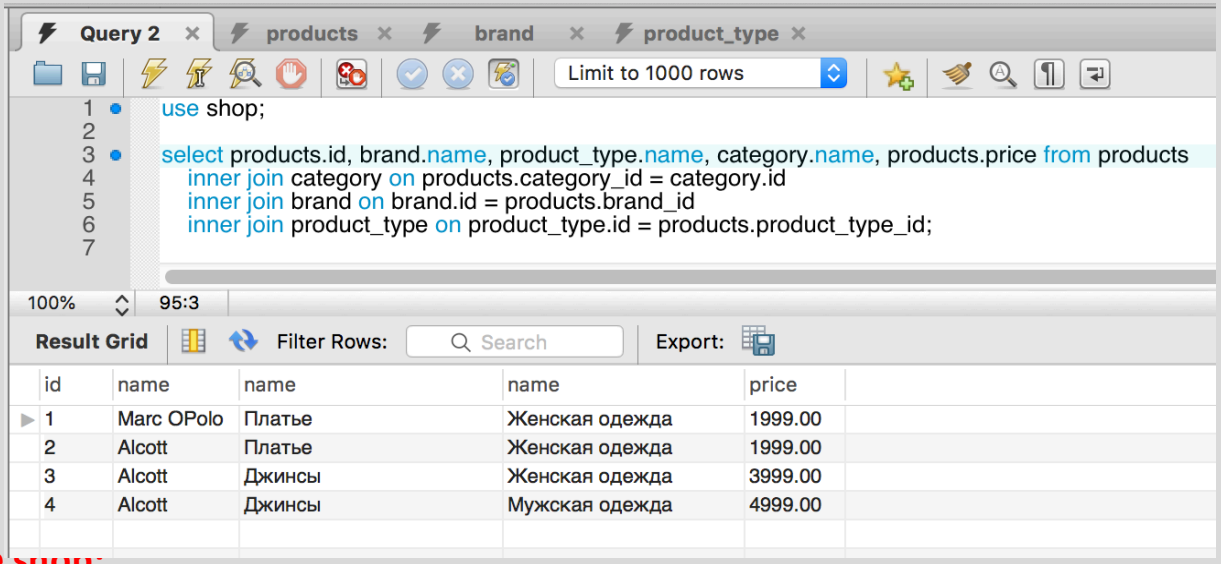
2	Объединение данных в таблице PRODUCTS с данными из CATEGORY. ПО ВЫБОРОЧНЫМ ПОЛЯМ 	 **use shop;** **select products.id, price, name from products** **inner join category on products.category_id = category.id;**

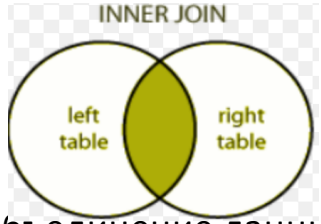
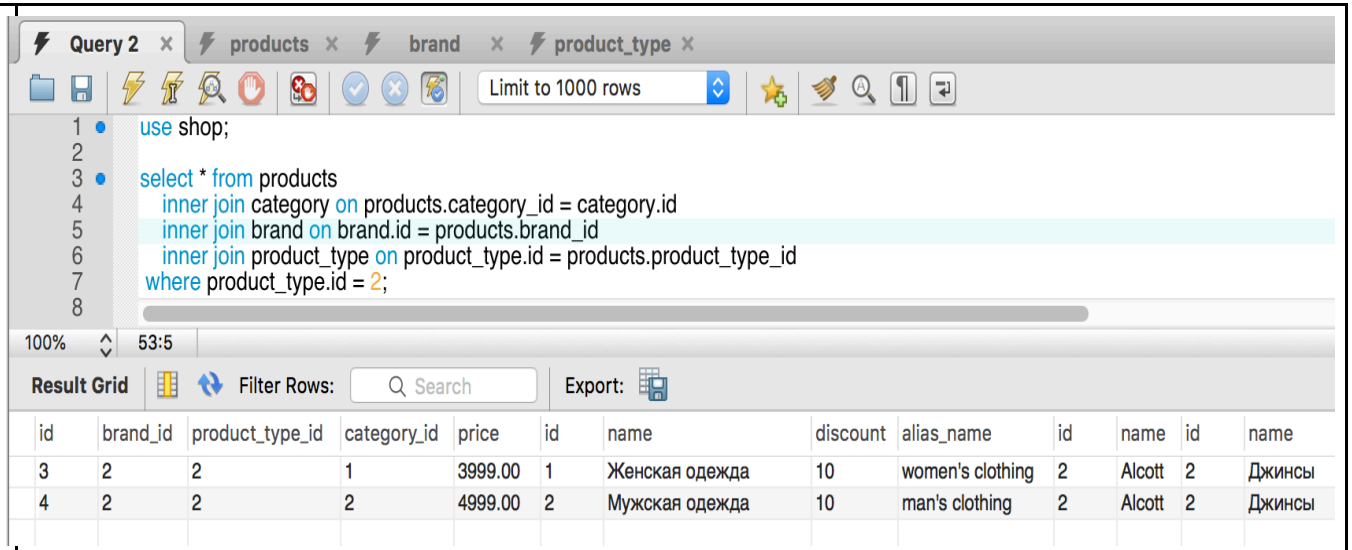
3	 Объединение данных в таблице CATEGORY с данными из PRODUCTS. Знак * - означает вывести ВСЕ ДАННЫЕ	 **use shop;** **select * from category** **inner join products on products.category_id = category.id;**

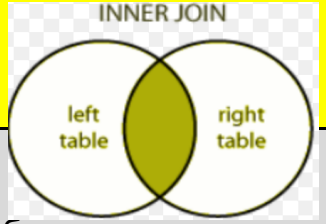
Объединение данных в таблице CATEGORY с данными из PRODUCTS. Знак * - означает вывести ВСЕ ДАННЫЕ

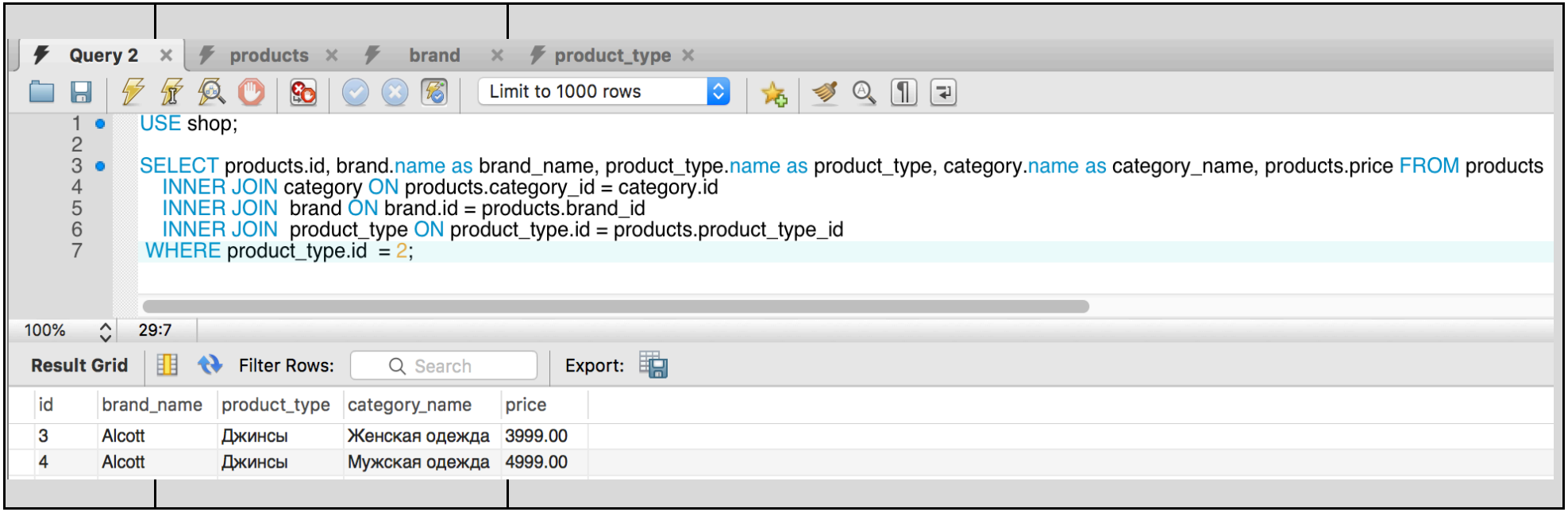

```
use shop;
select * from category
 inner join products on products.category_id = category.id;
```

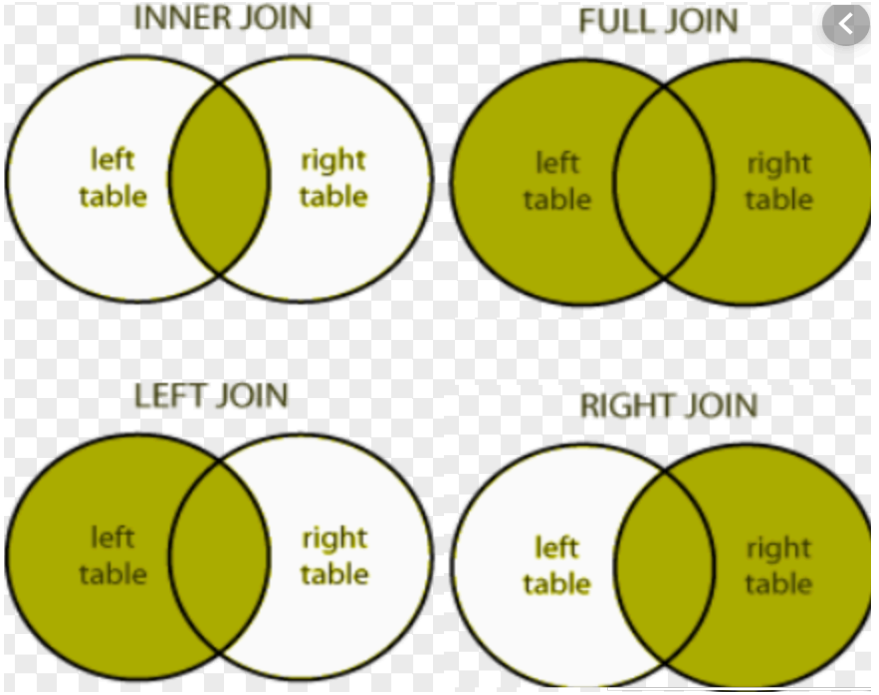
5	Объединение данных в таблице PRODUCTS с данными из CATEGORY, BRAND, PRODUCT_TYPE. Знак * - означает вывести ВСЕ ДАННЫЕ	 **use shop;** **select * from products** **inner join category on products.category_id = category.id** **inner join brand on brand.id = products.brand_id** **inner join product_type on product_type.id = products.product_type_id;**

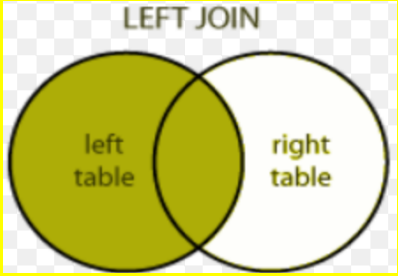
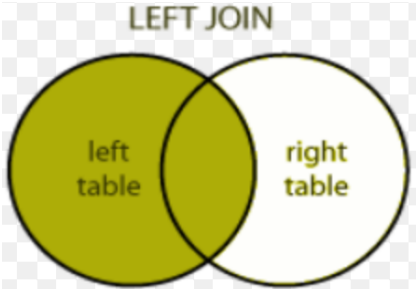
6	 Объединение данных в таблице PRODUCTS с данными из CATEGORY, BRAND, PRODUCT_TYPE. **ВЫБОРОЧНЫЕ СТОЛБЦЫ**	 **use shop;** **select products.id, brand.name, product_type.name, category.name, products.price from products** **inner join category on products.category_id = category.id** **inner join brand on brand.id = products.brand_id** **inner join product_type on product_type.id = products.product_type_id;**

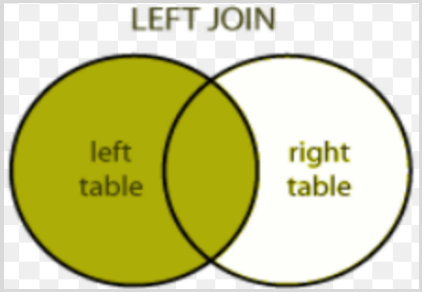
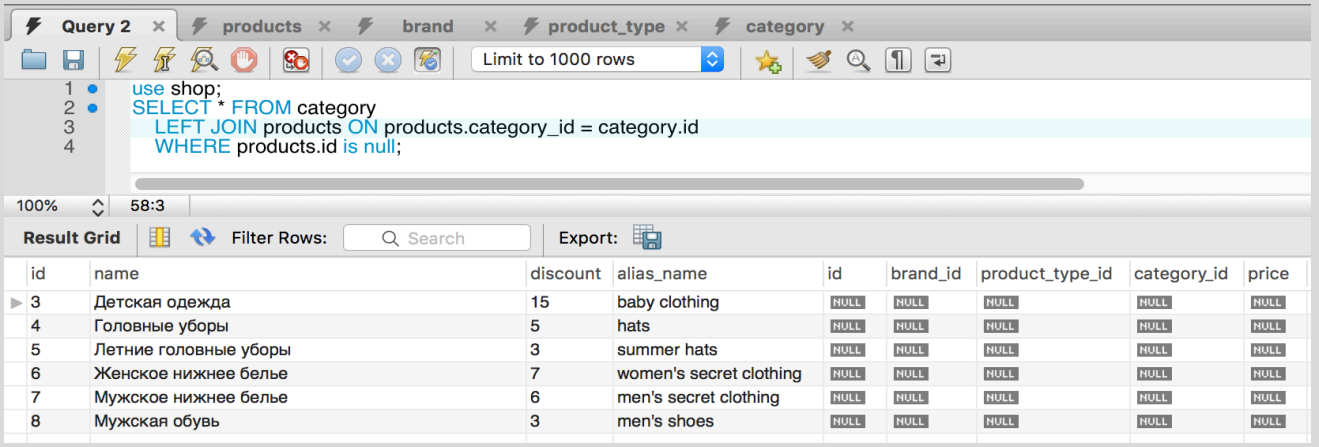
7 ДЗ	 Объединение данных в таблице PRODUCTS с данными из CATEGORY, BRAND, PRODUCT_TYPE. Знак * - означает вывести ВСЕ ДАННЫЕ, НО!!!!! **Только ДЖИНСЫ!!!!!**	 id brand_id product_type_id category_id price id name discount alias_name id name id name 3 2 2 1 3999.00 1 Женская одежда 10 women's clothing 2 Alcott 2 Джинсы 4 2 2 2 4999.00 2 Мужская одежда 10 man's clothing 2 Alcott 2 Джинсы **use shop;** **select * from products** **inner join category on products.category_id = category.id** **inner join brand on brand.id = products.brand_id** **inner join product_type on product_type.id = products.product_type_id** **where product_type.id = 2;**

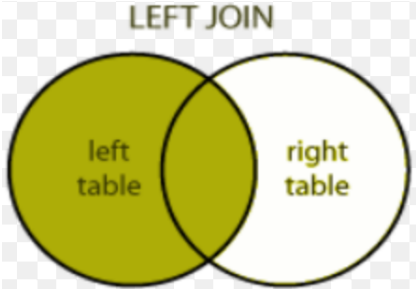
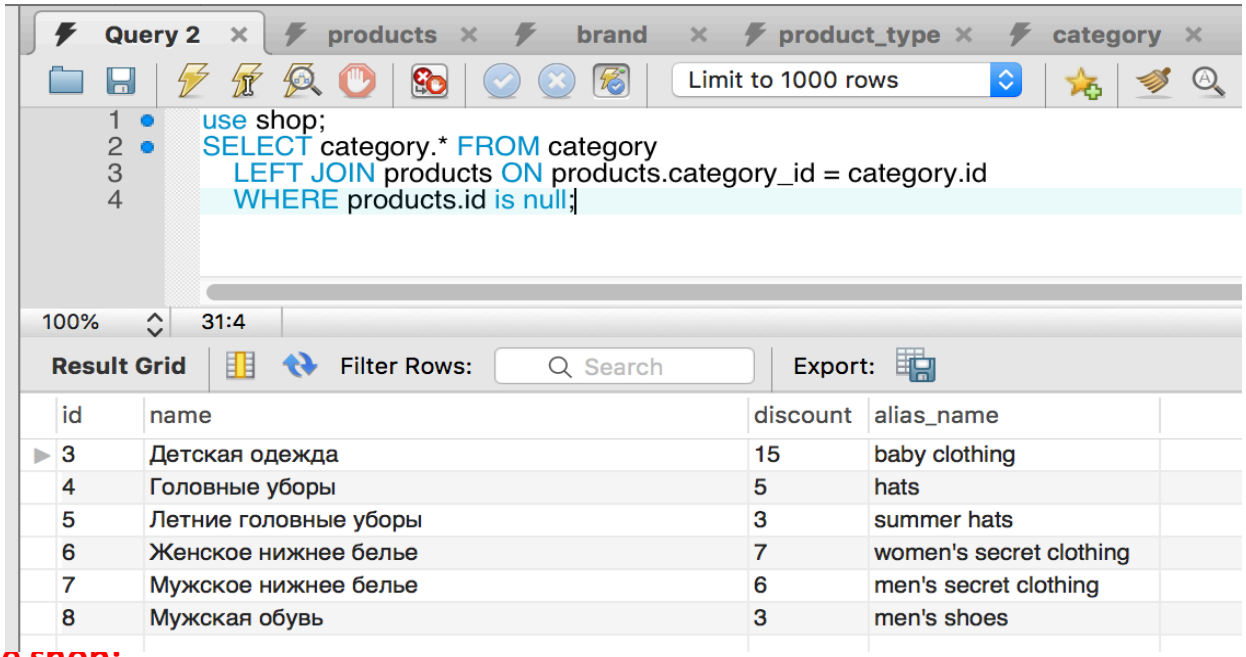
15 lesson		**Урок 15. Объединение данных из нескольких таблиц (Операторы LEFT JOIN и RIGHT JOIN)**
8 ДЗ с урока 14	Объединение данных в таблице PRODUCTS с данными из CATEGORY, BRAND, PRODUCT_TYPE. ВЫБОРОЧНЫЕ СТОЛБЦЫ и !!!! **Только ДЖИНСЫ (id 2 in product_type)!!!!!**	**USE shop;** **SELECT products.id, brand.name as brand_name, product_type.name as product_type, category.name as category_name, products.price FROM products** **INNER JOIN category ON products.category_id = category.id** **INNER JOIN brand ON brand.id = products.brand_id** **INNER JOIN product_type ON product_type.id = products.product_type_id** **WHERE product_type.id = 2;**

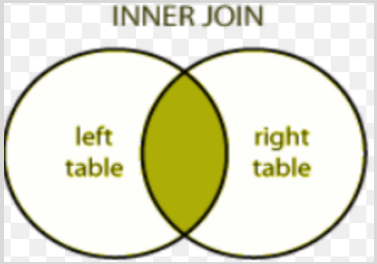
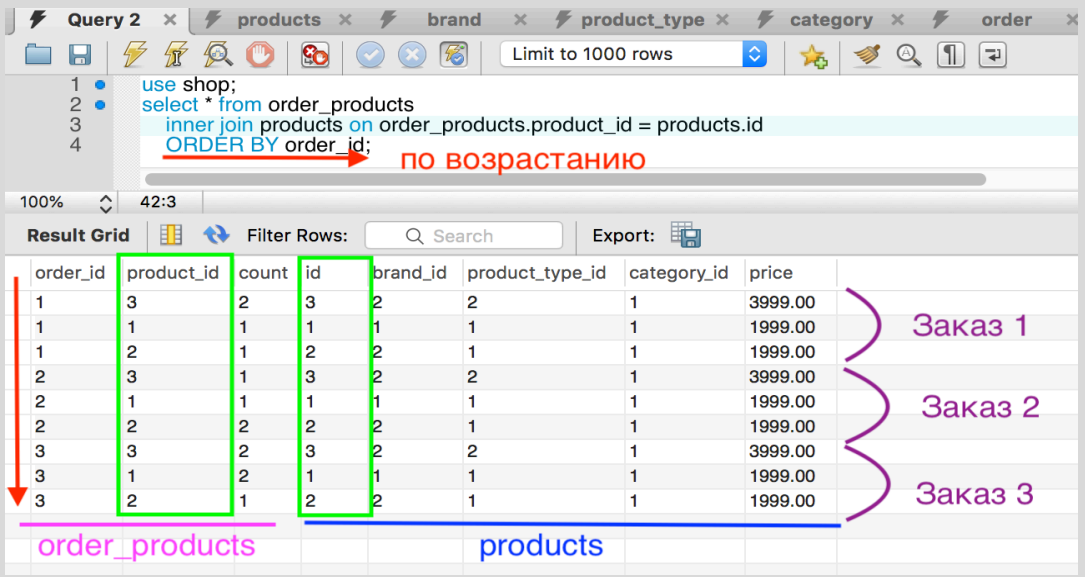

Query 2
products
brand
product_type
Limit to 1000 rows
USE shop;
SELECT products.id, brand.name as brand_name, product_type.name as product_type, category.name as category_name, products.price FROM products
INNER JOIN category ON products.category_id = category.id
INNER JOIN brand ON brand.id = products.brand_id
INNER JOIN product_type ON product_type.id = products.product_type_id
WHERE product_type.id = 2;
100%
29:7
Result Grid
Filter Rows:
Search
Export:
id
brand_name
product_type
category_name
price
3
Alcott
Джинсы
Женская одежда
3999.00
4
Alcott
Джинсы
Мужская одежда
4999.00

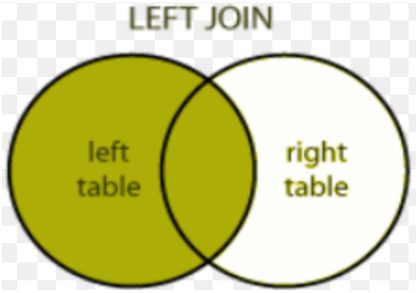
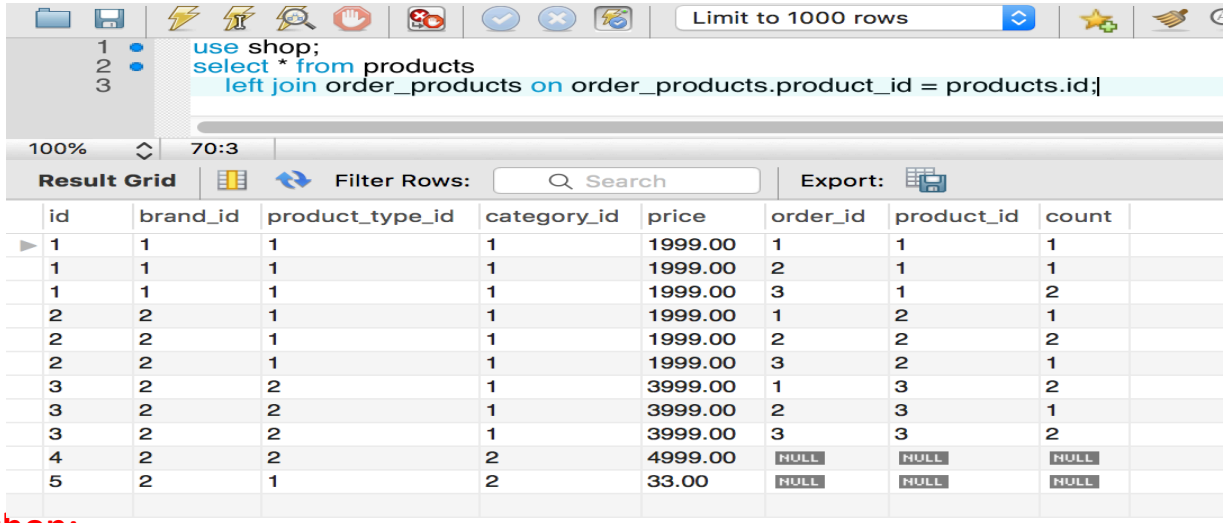
		 INNER JOIN left table right table FULL JOIN left table right table LEFT JOIN left table right table RIGHT JOIN left table right table

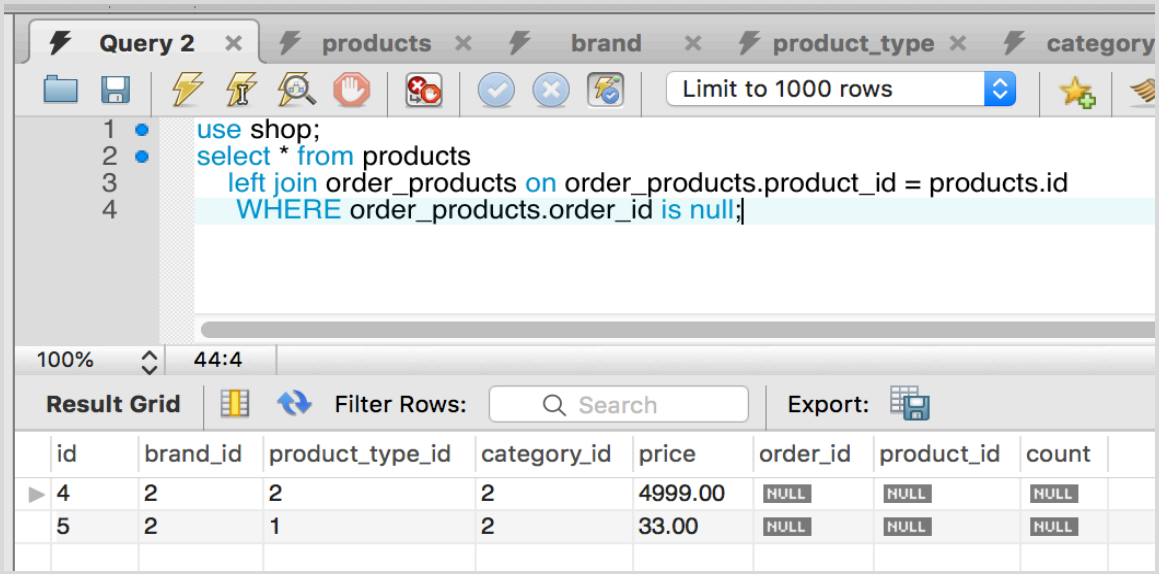
		**ОПЕРАТОР LEFT JOIN - с его помощью можно посмотреть категории товаров, которых НЕТ в PRODUCTS.** **Это объединение при котором в результирующий запрос попадает все из left table и строки из left table в которых есть пересечение с right table - при этом недостающие строчки заполняются NULL**
1	Покажи все из CATEGORY даже если этих продуктов НЕТ в PRODUCTS 	Query 2 × products × brand × product_type × category × Limit to 1000 rows 1 SELECT * FROM category 2 LEFT JOIN products ON products.category_id = category.id; 3 100% 59:3 Result Grid Filter Rows: Search Export: id | name | discount | alias_name | id | brand_id | product_type_id | category_id | price 1 | Женская одежда | 10 | women's clothing | 1 | 1 | 1 | 1 | 1999.00 1 | Женская одежда | 10 | women's clothing | 2 | 2 | 1 | 1 | 1999.00 1 | Женская одежда | 10 | women's clothing | 3 | 2 | 2 | 1 | 3999.00 2 | Мужская одежда | 10 | man's clothing | 4 | 2 | 2 | 2 | 4999.00 3 | Детская одежда | 15 | baby clothing | NULL | NULL | NULL | NULL | NULL 4 | Головные уборы | 5 | hats | NULL | NULL | NULL | NULL | NULL 5 | Летние головные уборы | 3 | summer hats | NULL | NULL | NULL | NULL | NULL 6 | Женское нижнее белье | 7 | women's secret clothing | NULL | NULL | NULL | NULL | NULL 7 | Мужское нижнее белье | 6 | men's secret clothing | NULL | NULL | NULL | NULL | NULL 8 | Мужская обувь | 3 | men's shoes | NULL | NULL | NULL | NULL | NULL **SELECT * FROM category** **LEFT JOIN products ON products.category_id = category.id;**

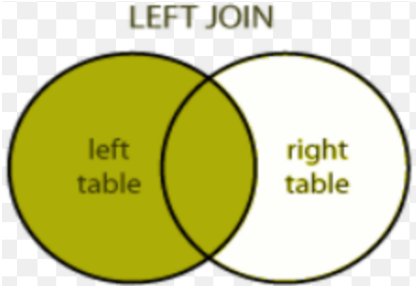
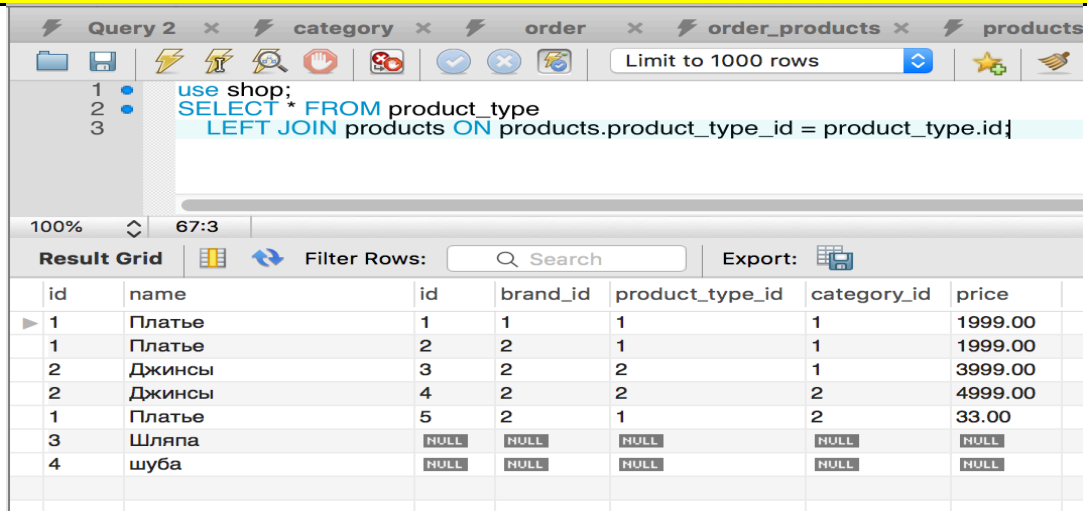
2	Если добавить к предыдущему запросу ограничение WHERE, то получим только те строки, по которым товар в products отсутствует 	 id | name | discount | alias_name | id | brand_id | product_type_id | category_id | price 3 | Детская одежда | 15 | baby clothing | NULL | NULL | NULL | NULL | NULL 4 | Головные уборы | 5 | hats | NULL | NULL | NULL | NULL | NULL 5 | Летние головные уборы | 3 | summer hats | NULL | NULL | NULL | NULL | NULL 6 | Женское нижнее белье | 7 | women's secret clothing | NULL | NULL | NULL | NULL | NULL 7 | Мужское нижнее белье | 6 | men's secret clothing | NULL | NULL | NULL | NULL | NULL 8 | Мужская обувь | 3 | men's shoes | NULL | NULL | NULL | NULL | NULL **use shop;** **SELECT * FROM category** **LEFT JOIN products ON products.category_id = category.id** **WHERE products.id is null;**

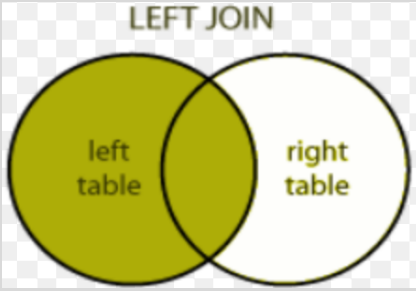
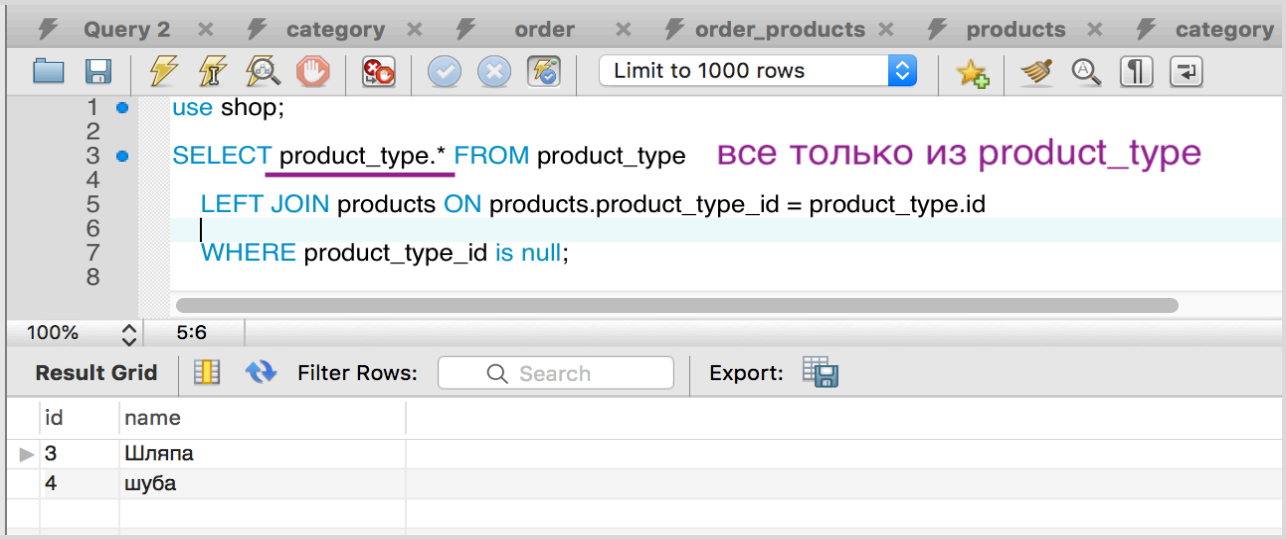
3	Покажи товары из CATEGORY, КОТОРЫХ НЕТ В PRODUCTS. **Столбцы с NULL НЕ ВЫВОДИТЬ** 	 id | name | discount | alias_name 3 | Детская одежда | 15 | baby clothing 4 | Головные уборы | 5 | hats 5 | Летние головные уборы | 3 | summer hats 6 | Женское нижнее белье | 7 | women's secret clothing 7 | Мужское нижнее белье | 6 | men's secret clothing 8 | Мужская обувь | 3 | men's shoes **use shop;** **SELECT category.* FROM category** **LEFT JOIN products ON products.category_id = category.id** **WHERE products.id is null;**

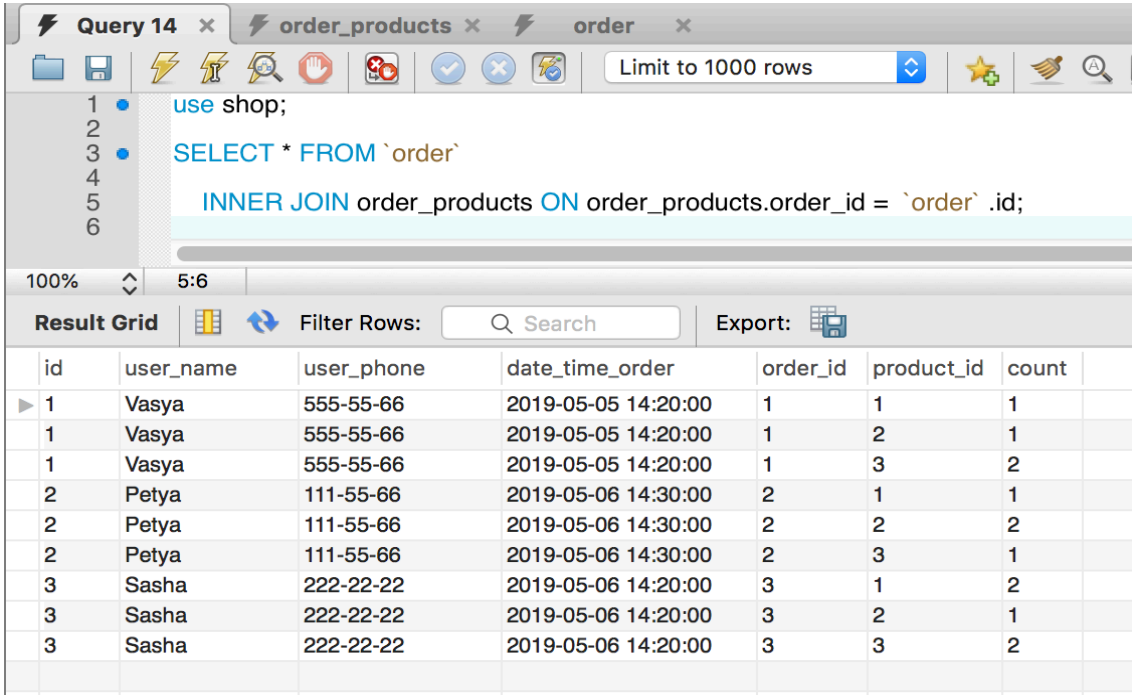
4 дз	Покажи все из ORDER_PRODUCTS, что пересекается с PRODUCTS. Сделай по возрастанию столбец order_id 	 **use shop;** **select * from order_products** **inner join products on order_products.product_id = products.id** **ORDER BY order_id;**

5 дз	Покажи все из PRODUCTS, даже если этих продуктов в заказах ORDER_PRODUCTS НЕТ! 	 id brand_id product_type_id category_id price order_id product_id count 1 1 1 1 1999.00 1 1 1 1 1 1 1 1999.00 2 1 1 1 1 1 1 1999.00 3 1 2 2 2 1 1 1999.00 1 2 1 2 2 1 1 1999.00 2 2 2 2 2 1 1 1999.00 3 2 1 3 2 2 1 3999.00 1 3 2 3 2 2 1 3999.00 2 3 1 3 2 2 1 3999.00 3 3 2 4 2 2 2 4999.00 NULL NULL NULL 5 2 1 2 33.00 NULL NULL NULL **use shop;** **select * from products** **LEFT JOIN order_products on order_products.product_id = products.id;**

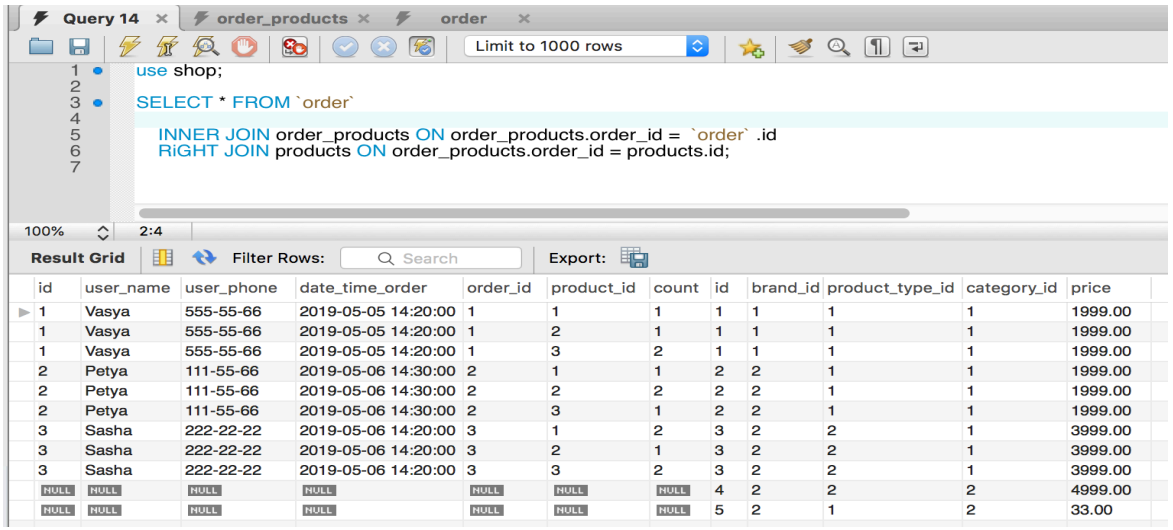
бдз	Если добавить к предыдущему запросу ограничение WHERE, то получим только те строки, по которым товар в ORDER_PRODUCTS отсутствует 	 **use shop;** **select * from products** **LEFT JOIN order_products on order_products.product_id = products.id** **WHERE order_products.order_id is null;**

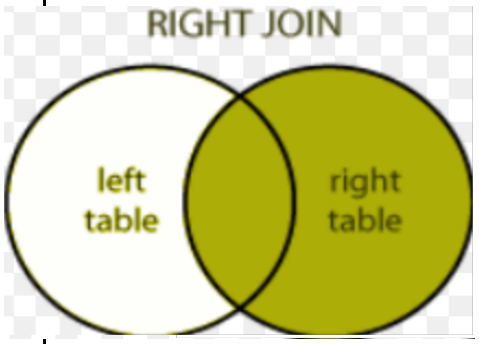
16 lesson		ДЗ с урока 15
1. ДЗ с урока 15	Покажи (объедини) все из PRODUCT_TYPE даже то, чего нет в PRODUCTS 	 id name id brand_id product_type_id category_id price 1 Платье 1 1 1 1 1999.00 1 Платье 2 2 1 1 1999.00 2 Джинсы 3 2 2 1 3999.00 2 Джинсы 4 2 2 2 4999.00 1 Платье 5 2 1 2 33.00 3 Шляпа NULL NULL NULL NULL NULL 4 шуба NULL NULL NULL NULL NULL **use shop;** **SELECT * FROM product_type** ***LEFT JOIN*** **products ON products.product_type_id = product_type.id;**

2. ДЗ с урока 15	Покажи (объедини) все из PRODUCT_TYPE даже то, чего нет в PRODUCTS, но выведи только то, что в условии WHERE  	 **use shop;** **SELECT product_type.* FROM product_type** **LEFT JOIN products ON products.product_type_id = product_type.id** **WHERE product_type_id is null;**

3. ДЗ	Вывести информацию обо всех товарах, которые НЕ ПОПАЛИ НИ В ОДИН ЗАКАЗ, Делаем это поэтапно. СМ. 3 скрина	

id	user_name	user_phone	date_time_order	order_id	product_id	count
1	Vasya	555-55-66	2019-05-05 14:20:00	1	1	1
1	Vasya	555-55-66	2019-05-05 14:20:00	1	2	1
1	Vasya	555-55-66	2019-05-05 14:20:00	1	3	2
2	Petya	111-55-66	2019-05-06 14:30:00	2	1	1
2	Petya	111-55-66	2019-05-06 14:30:00	2	2	2
2	Petya	111-55-66	2019-05-06 14:30:00	2	3	1
3	Sasha	222-22-22	2019-05-06 14:20:00	3	1	2
3	Sasha	222-22-22	2019-05-06 14:20:00	3	2	1
3	Sasha	222-22-22	2019-05-06 14:20:00	3	3	2


Query 14
order_products
order
Limit to 1000 rows
use shop;
SELECT * FROM `order`
INNER JOIN order_products ON order_products.order_id = `order` .id
RiGHT JOIN products ON order_products.order_id = products.id;
100%
2:4
Result Grid
Filter Rows:
Search
Export:
id user_name user_phone date_time_order order_id product_id count id brand_id product_type_id category_id price
1 Vasya 555-55-66 2019-05-05 14:20:00 1 1 1 1 1 1 1 1999.00
1 Vasya 555-55-66 2019-05-05 14:20:00 1 2 1 1 1 1 1 1999.00
1 Vasya 555-55-66 2019-05-05 14:20:00 1 3 2 1 1 1 1 1999.00
2 Petya 111-55-66 2019-05-06 14:30:00 2 1 1 2 2 1 1 1999.00
2 Petya 111-55-66 2019-05-06 14:30:00 2 2 2 2 2 1 1 1999.00
2 Petya 111-55-66 2019-05-06 14:30:00 2 3 1 2 2 1 1 1999.00
3 Sasha 222-22-22 2019-05-06 14:20:00 3 1 2 3 2 2 1 3999.00
3 Sasha 222-22-22 2019-05-06 14:20:00 3 2 1 3 2 2 1 3999.00
3 Sasha 222-22-22 2019-05-06 14:20:00 3 3 2 3 2 2 1 3999.00
NULL NULL NULL NULL NULL NULL NULL 4 2 2 2 4999.00
NULL NULL NULL NULL NULL NULL NULL 5 2 1 2 33.00



Query 14 | order_products | order

Limit to 1000 rows

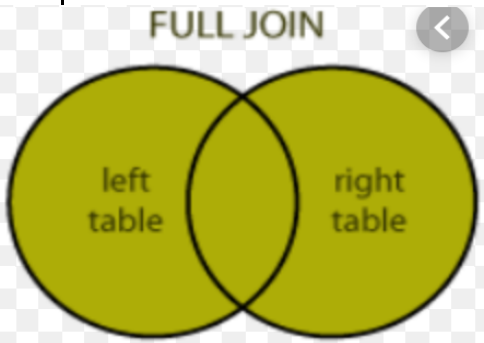
```
1  use shop;
2
3  SELECT products. * FROM `order`
4
5     INNER JOIN order_products ON order_products.order_id =  `order` .id
6     RiGHT JOIN products ON order_products.order_id = products.id
7     WHERE `order` . id is null;
8
```

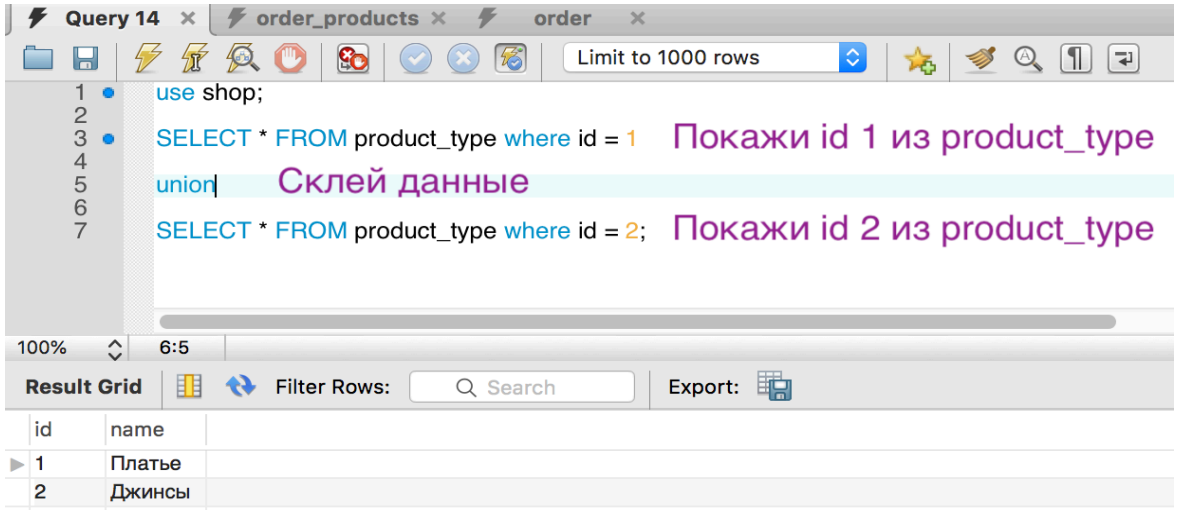
100% 72:5

Result Grid | Filter Rows: Search | Export:

id	brand_...	product_type_id	category_id	price
4	2	2	2	4999.00
5	2	1	2	33.00

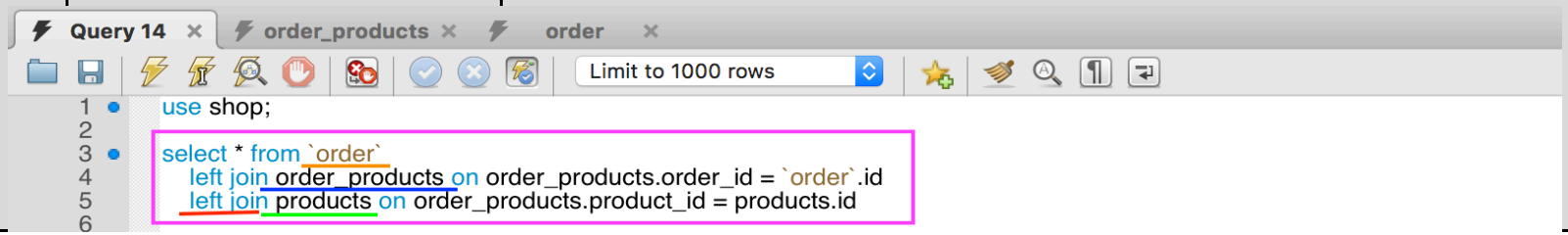
```
use shop;
SELECT products. * FROM `order`
   INNER JOIN order_products ON order_products.order_id =  `order` .id
   RiGHT JOIN products ON order_products.order_id = products.id
WHERE `order` . id is null;
```

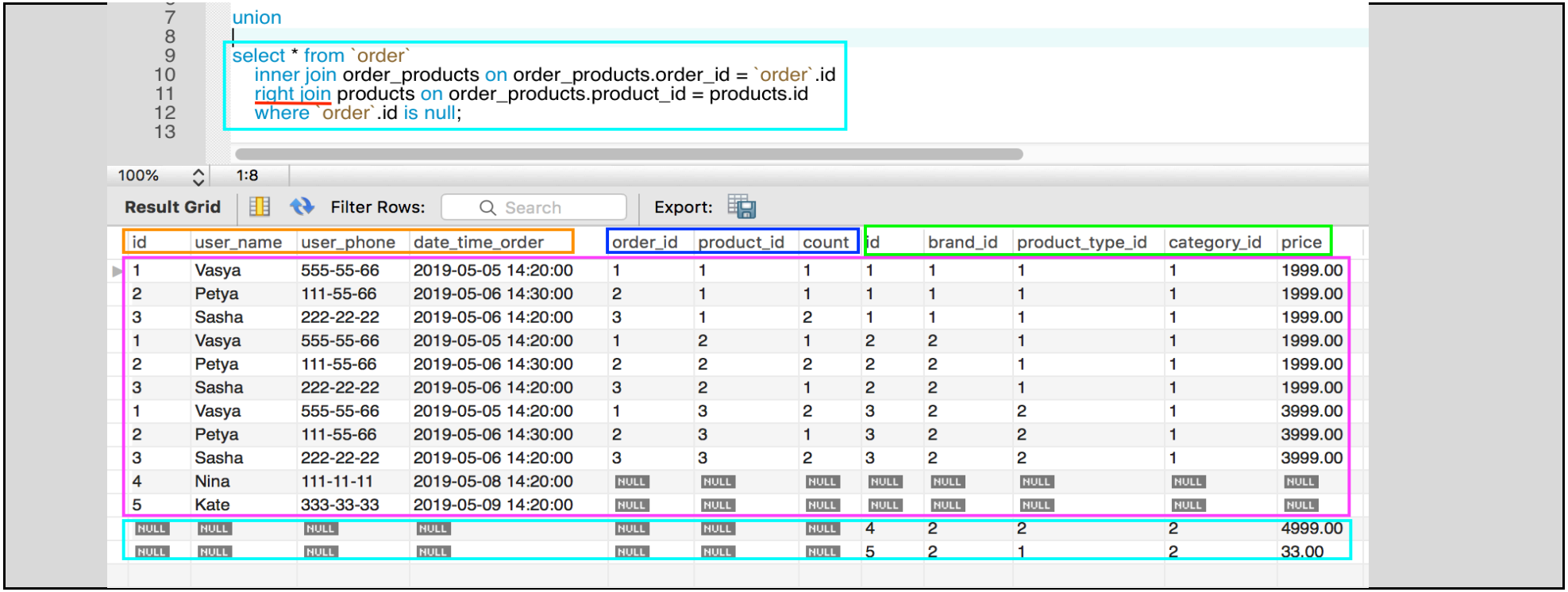
16 lesson		ПОЛНОЕ объединение данных из НЕСКОЛЬКИХ таблиц (FULL JOIN FULL OUTER JOIN). UNION
		Полное объединение данных двух таблиц - ВСЕ без исключения строки обеих таблиц БЕЗ зависимости их пересечения. **В MySQL директивы FULL OUTER JOIN нет (она есть в MS SQL, Oracle, Post greSql). В MySQL используют дерективу UNION**

1	принцип UNION - команда "СКЛЕЙ" строчки запросов, которые до и после **UNION**	 **use shop;** **SELECT * FROM product_type where id = 1** **union** **SELECT * FROM product_type where id = 2;**

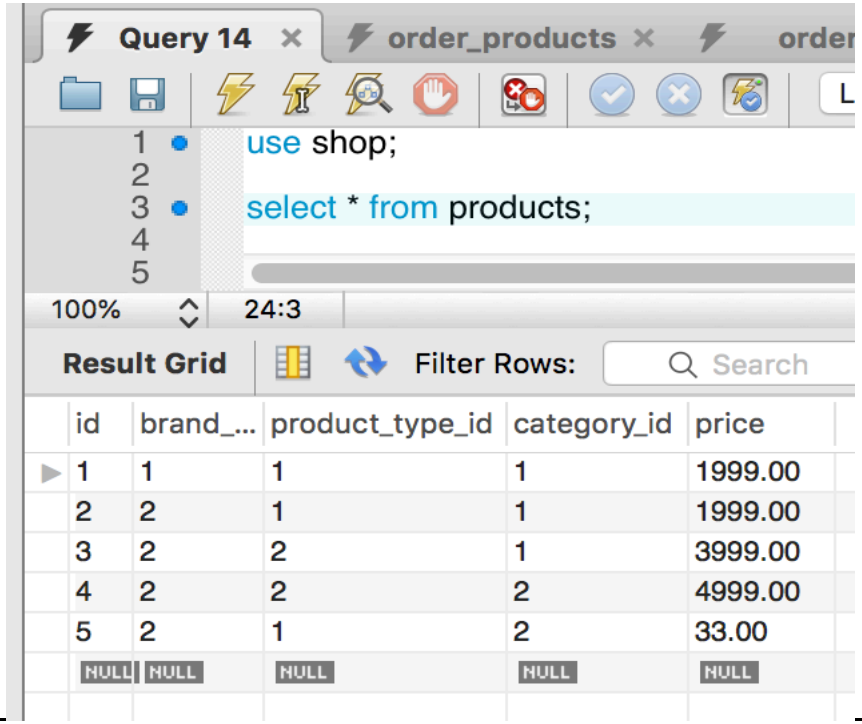
2

```
use shop;
select * from `order`
    left join order_products on order_products.order_id = `order`.id
    left join products on order_products.product_id = products.id

union

select * from `order`
    inner join order_products on order_products.order_id = `order`.id
    right join products on order_products.product_id = products.id
    where `order`.id is null;
```

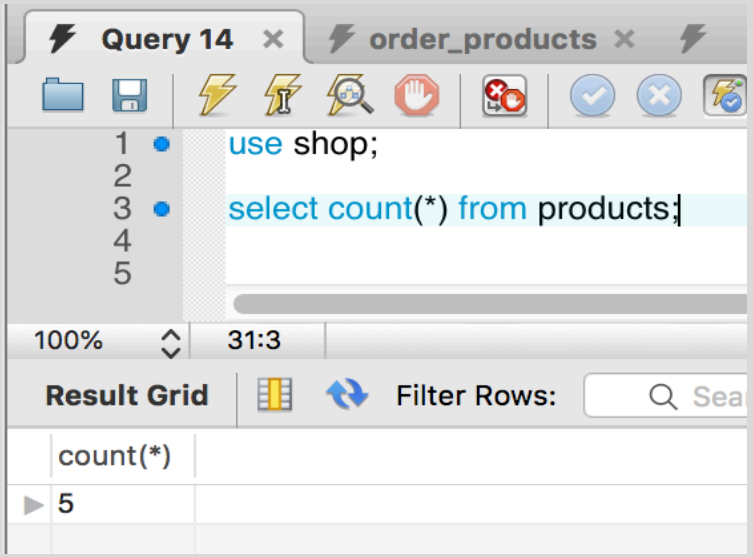


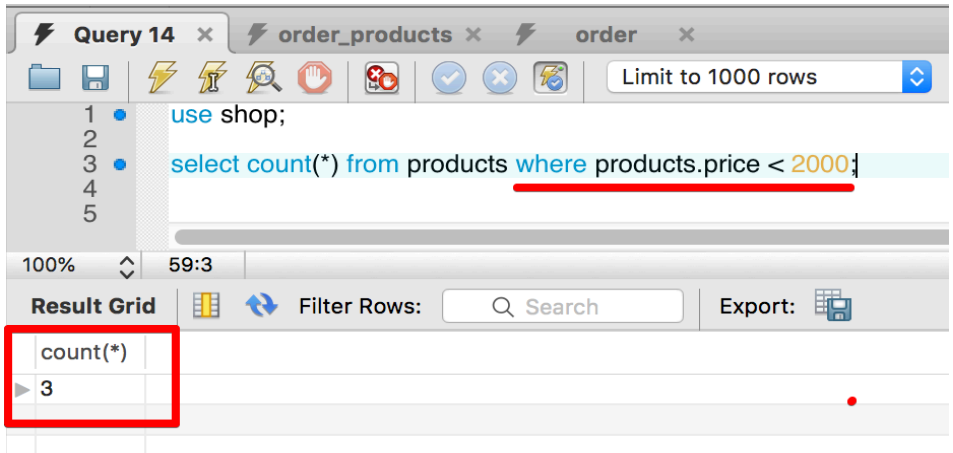
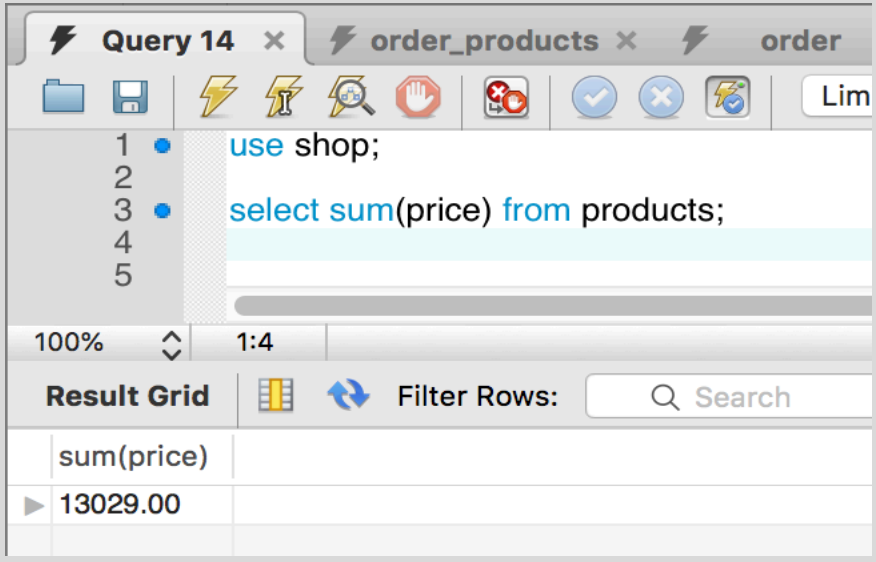


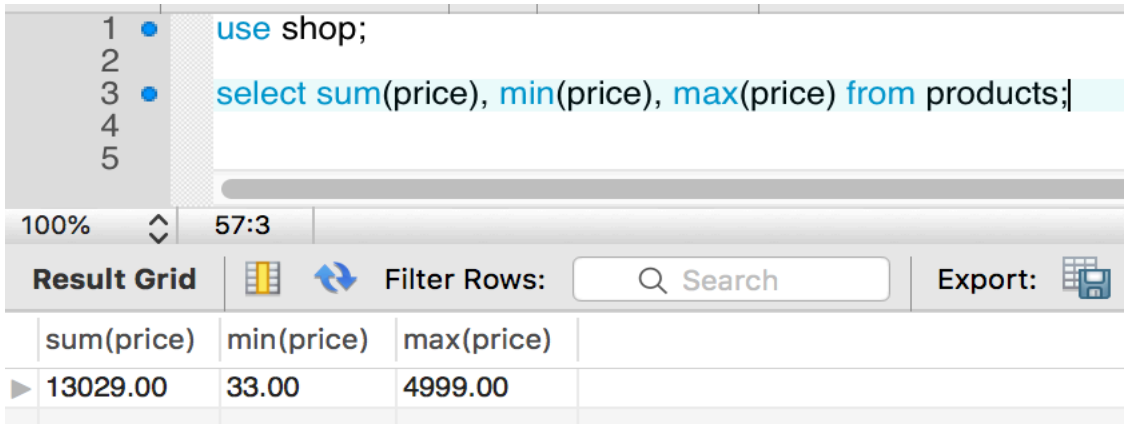
id	user_name	user_phone	date_time_order	order_id	product_id	count	id	brand_id	product_type_id	category_id	price
1	Vasya	555-55-66	2019-05-05 14:20:00	1	1	1	1	1	1	1	1999.00
2	Petya	111-55-66	2019-05-06 14:30:00	2	1	1	1	1	1	1	1999.00
3	Sasha	222-22-22	2019-05-06 14:20:00	3	1	2	1	1	1	1	1999.00
1	Vasya	555-55-66	2019-05-05 14:20:00	1	2	1	2	2	1	1	1999.00
2	Petya	111-55-66	2019-05-06 14:30:00	2	2	2	2	2	1	1	1999.00
3	Sasha	222-22-22	2019-05-06 14:20:00	3	2	1	2	2	1	1	1999.00
1	Vasya	555-55-66	2019-05-05 14:20:00	1	3	2	3	2	2	1	3999.00
2	Petya	111-55-66	2019-05-06 14:30:00	2	3	1	3	2	2	1	3999.00
3	Sasha	222-22-22	2019-05-06 14:20:00	3	3	2	3	2	2	1	3999.00
4	Nina	111-11-11	2019-05-08 14:20:00	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	Kate	333-33-33	2019-05-09 14:20:00	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	4	2	2	2	4999.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	5	2	1	2	33.00

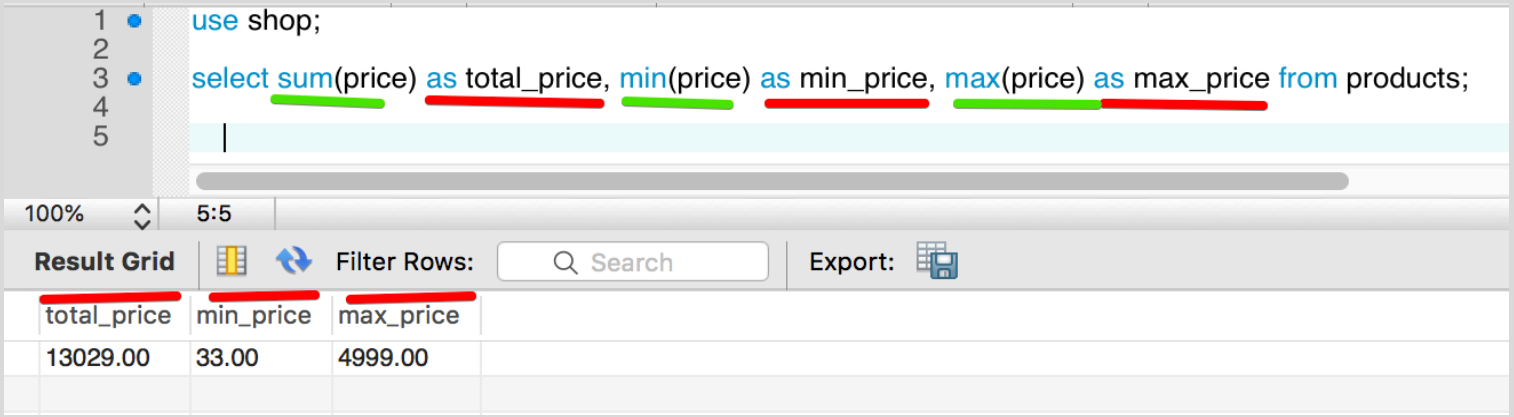
17		Агрегирующие функции (count, sum, max, min)
№	Действие	Команда на языке SQL и скрин результата
1	Покажи **всю таблицу** PRODUCTS	use shop; select * from products; 

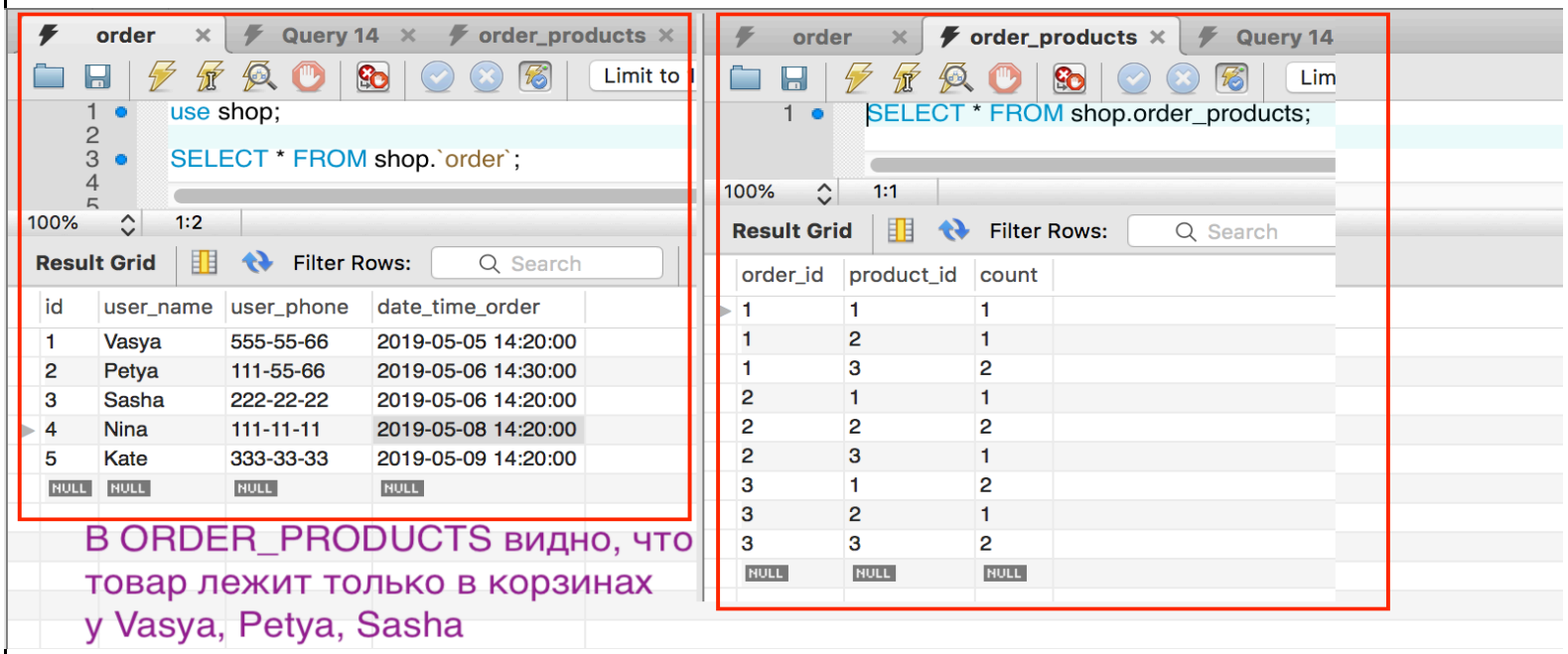
2	Покажи сколько строк в таблице PRODUCTS	**use shop;** **select count(*) from products;**



3	Покажи сколько строк в таблице PRODUCTS, удовлетворяющие условиям WHERE	 use shop; select count(*) from products where products.price < 2000;
4	Покажи сумму столбца. Сумма столбца price в PRODUCTS	 use shop; select sum(price) from products;

5	Покажи sum, min, max столбца price в PRODUCTS	 **use shop;** **select sum(price), min(price), max(price) from products;**

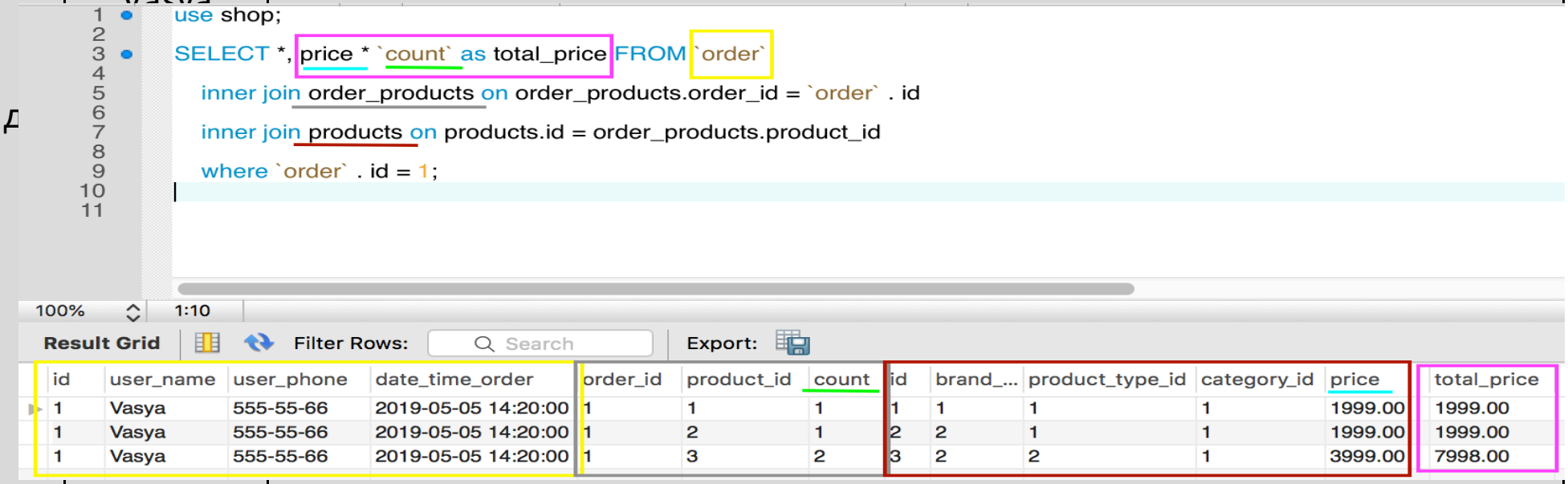
6	Покажи sum, min, max столбца price в PRODUCTS, с помощью as переимену й эти столбцы	 **use shop;** **select sum(price) as total_price, min(price) as min_price, max(price) as max_price from products;**

7 дз	По ORDER и ORDER_PRODUCTS видно, что товар есть только в корзинах Васи, Пети и Саши. У Нины и Кати товаров в	

8 д

Вывести суммарную стоимость заказа у Vasya

```
use shop;
SELECT *, price * `count` as total_price FROM `order`
 inner join order_products on order_products.order_id = `order` . id
   inner join products on products.id = order_products.product_id
   where `order` . id = 1;
```

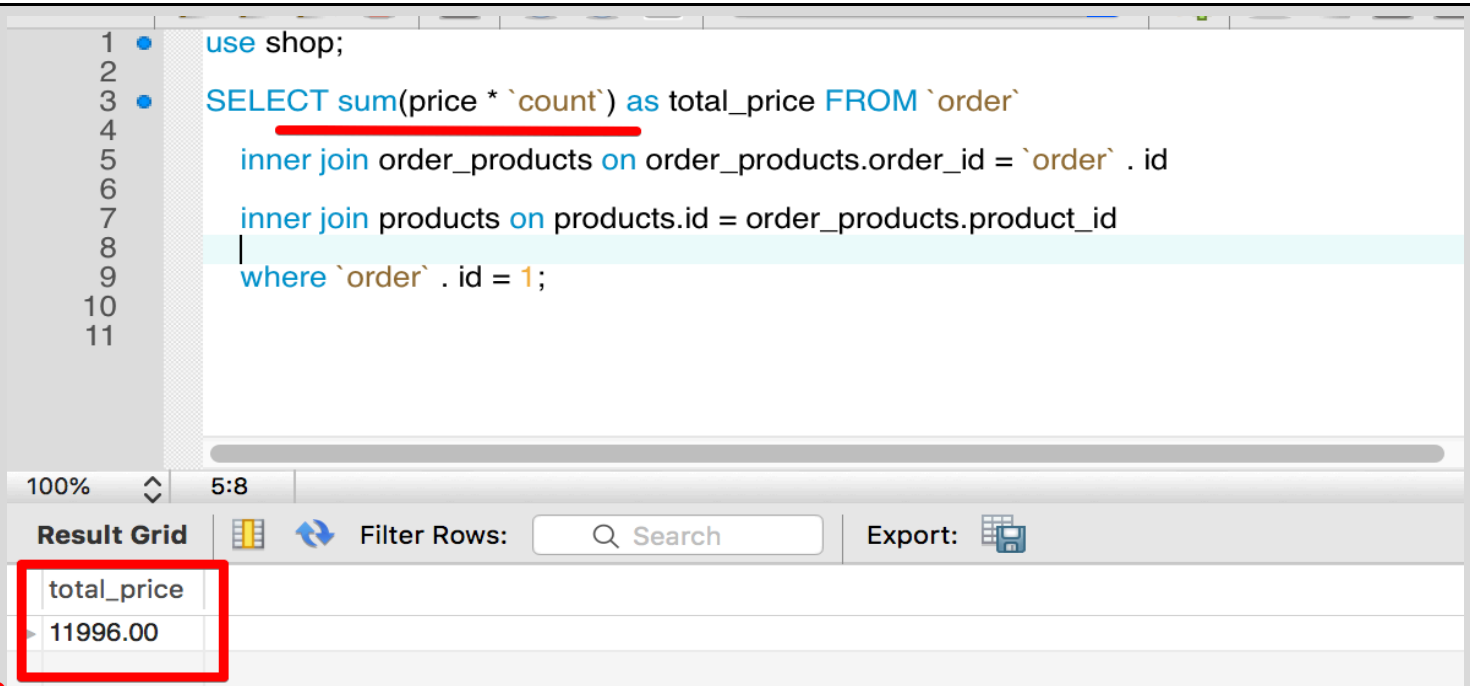


```
use shop;

SELECT *, price * `count` as total_price FROM `order`
   inner join order_products on order_products.order_id = `order` . id
   inner join products on products.id = order_products.product_id
   where `order` . id = 1;
```

100% 1:10

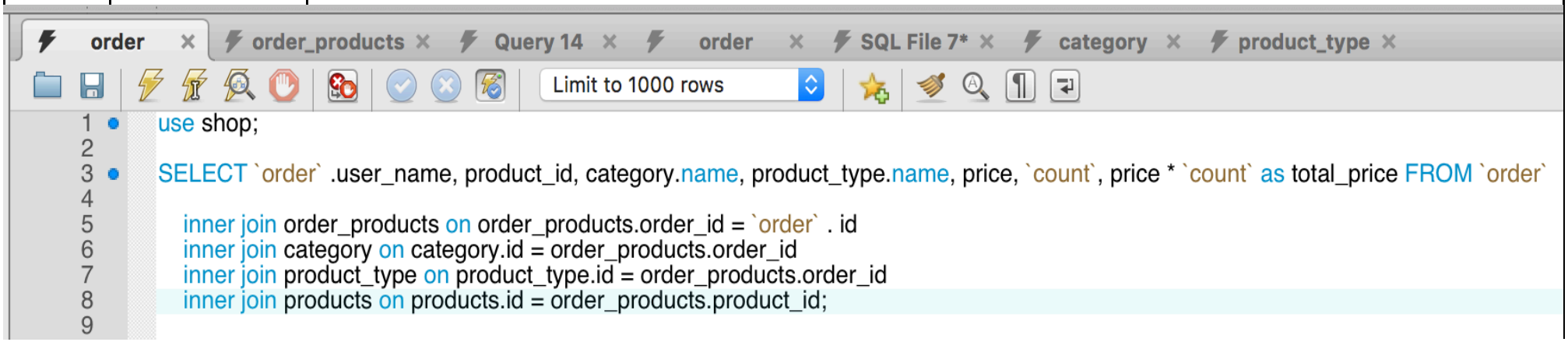
Result Grid | Filter Rows: Search | Export:

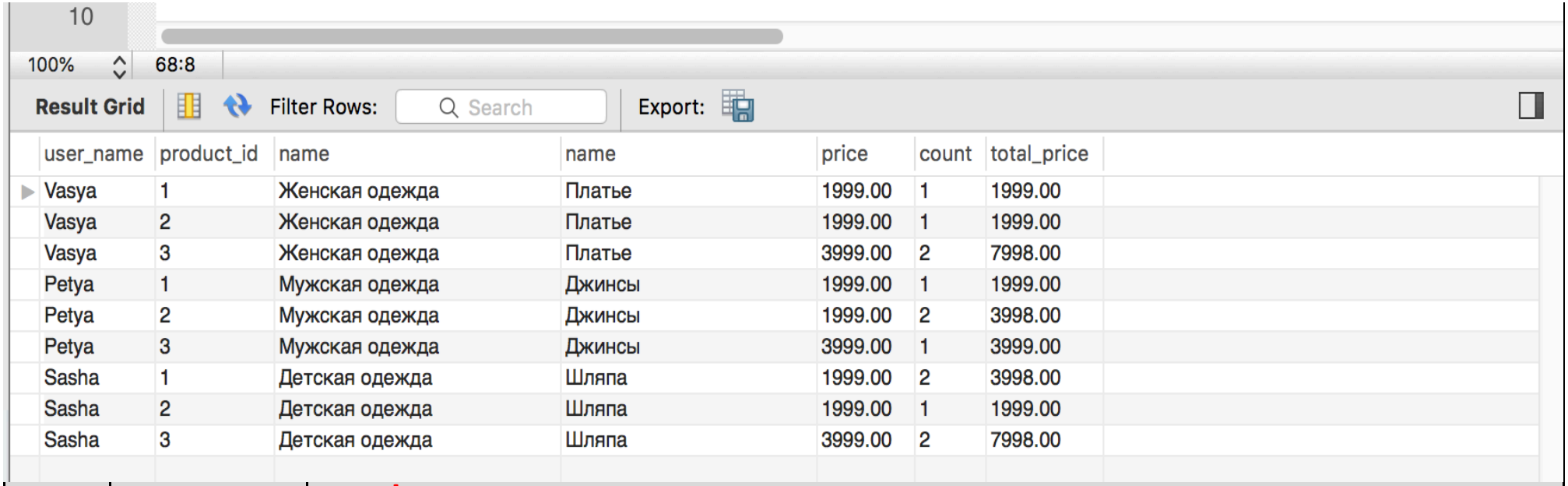
id	user_name	user_phone	date_time_order	order_id	product_id	count	id	brand_...	product_type_id	category_id	price	total_price
1	Vasya	555-55-66	2019-05-05 14:20:00	1	1	1	1	1	1	1	1999.00	1999.00
1	Vasya	555-55-66	2019-05-05 14:20:00	1	2	1	2	2	1	1	1999.00	1999.00
1	Vasya	555-55-66	2019-05-05 14:20:00	1	3	2	3	2	2	1	3999.00	7998.00

9 ДЗ	На основе данных из пункта выше вывести ИТОГО сумму по Васе. Изменилась только в строка 3	 use shop; SELECT sum(price * `count`) as total_price FROM `order` inner join order_products on order_products.order_id = `order` . id inner join products on products.id = order_products.product_id where `order` . id = 1;

18		**Оператор GROUP BY. Объединение по заказчику, вывод итого суммы по заказу, итого количество товаро в заказе (min, max, sum…)**
№	**Действие**	**Команда на языке SQL и скрин результата**
		Директива GROUP BY позволит группировать в одном запросе результаты агрегирующих функций для нескольких строк (получить суммарную стоимость заказа для каждого из пользователей)

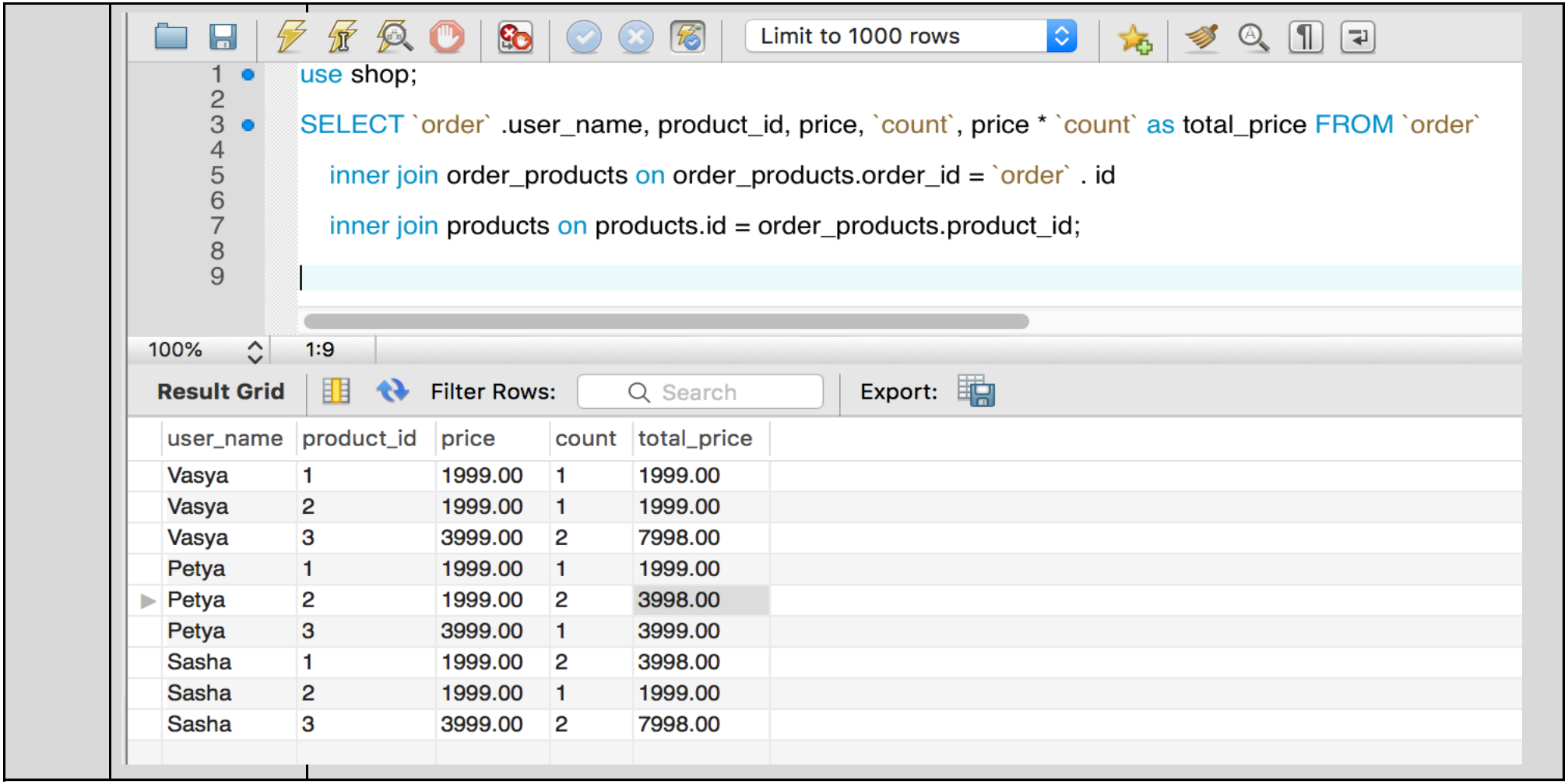
1	Так выглядят заказы, видно, что у каждого покупателя отдельная строка на товар.	use shop; SELECT `order` .user_name, product_id, category.name, product_type.name, price, `count`, price * `count` as total_price FROM `order` inner join order_products on order_products.order_id = `order` . id inner join category on category.id = order_products.order_id inner join product_type on product_type.id = order_products.order_id inner join products on products.id = order_products.product_id;

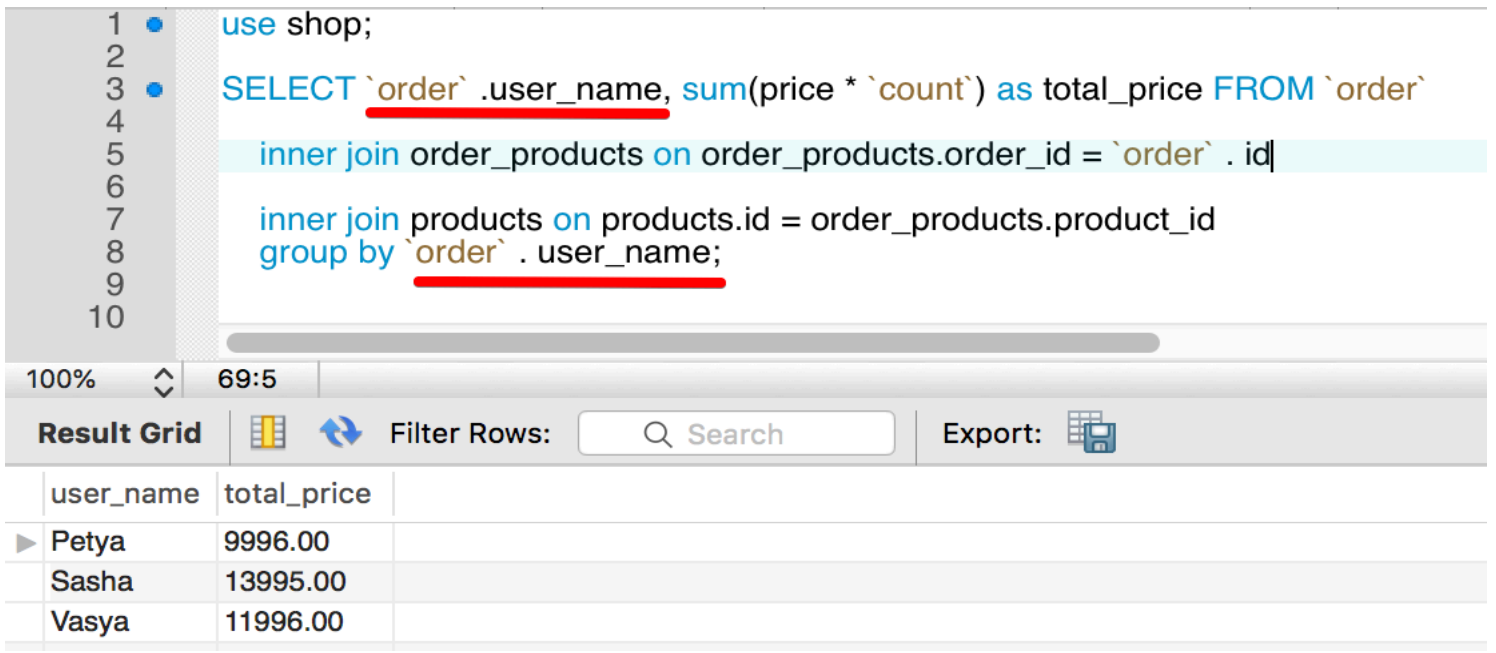


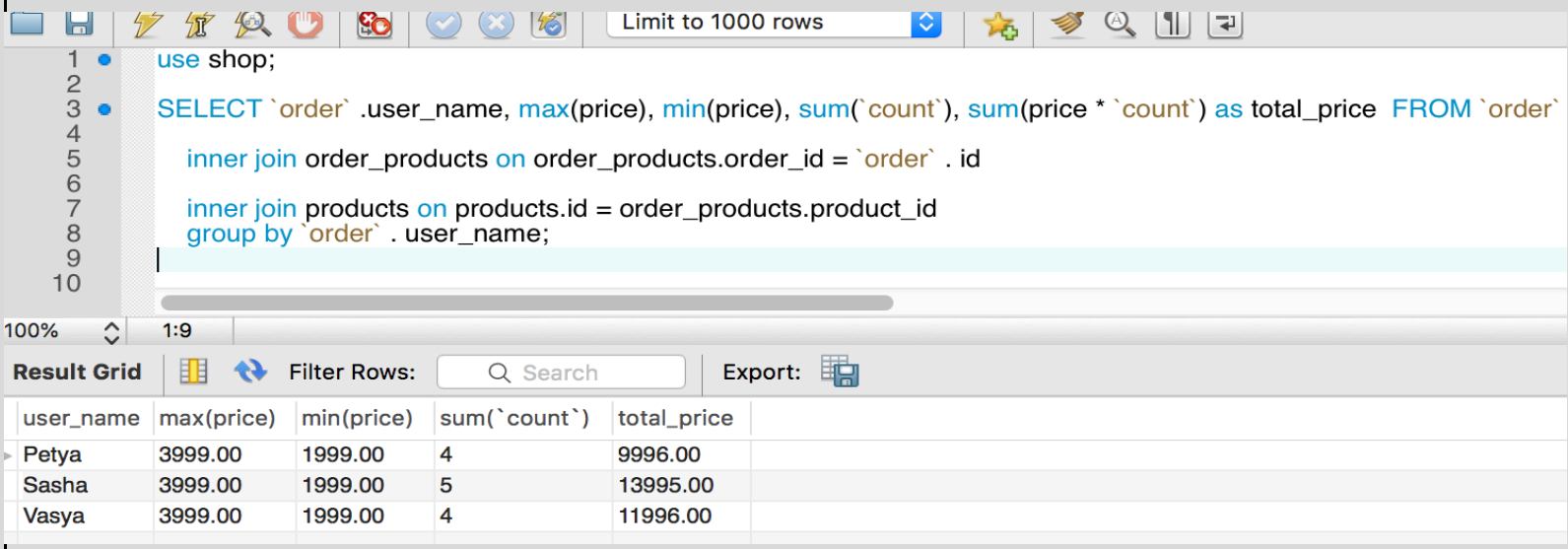


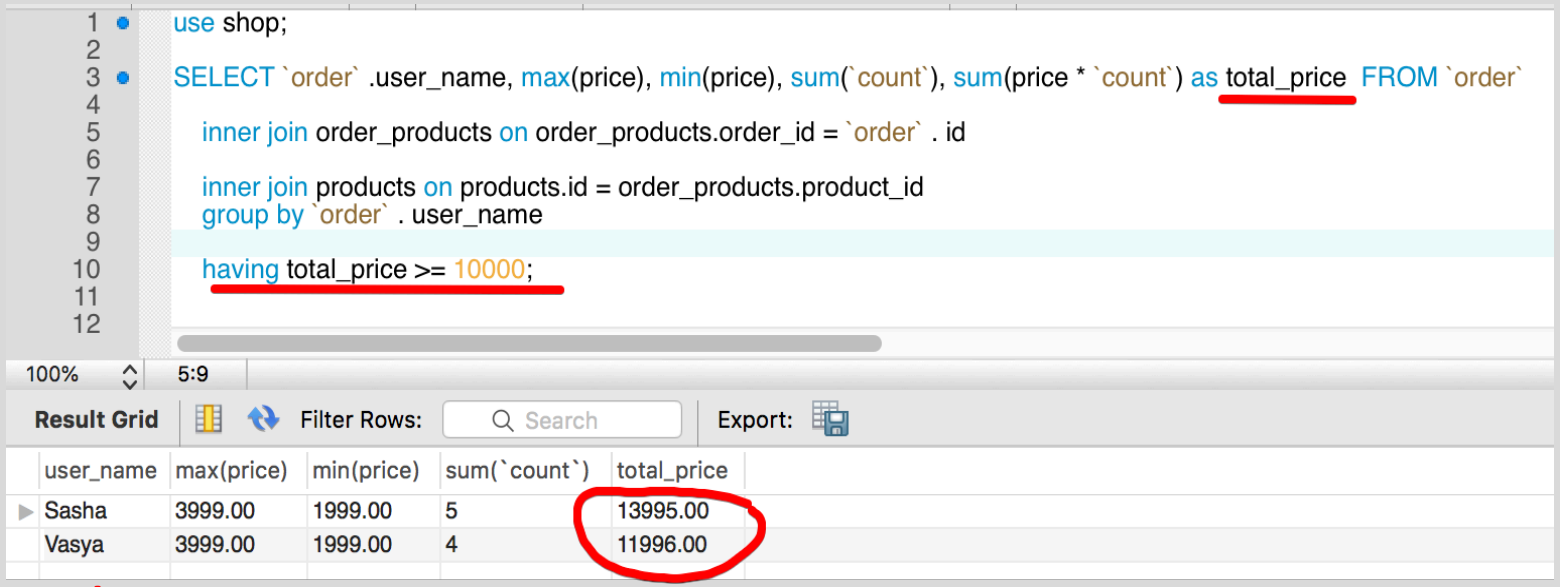
user_name	product_id	name	name	price	count	total_price
Vasya	1	Женская одежда	Платье	1999.00	1	1999.00
Vasya	2	Женская одежда	Платье	1999.00	1	1999.00
Vasya	3	Женская одежда	Платье	3999.00	2	7998.00
Petya	1	Мужская одежда	Джинсы	1999.00	1	1999.00
Petya	2	Мужская одежда	Джинсы	1999.00	2	3998.00
Petya	3	Мужская одежда	Джинсы	3999.00	1	3999.00
Sasha	1	Детская одежда	Шляпа	1999.00	2	3998.00
Sasha	2	Детская одежда	Шляпа	1999.00	1	1999.00
Sasha	3	Детская одежда	Шляпа	3999.00	2	7998.00

2	Или Так выглядят заказы, видно, что у каждого покупателя отдельная строка на товар.	use shop; SELECT `order` .user_name, product_id, price, `count`, price * `count` as total_price FROM `order` inner join order_products on order_products.order_id = `order` . id inner join products on products.id = order_products.product_id;


Limit to 1000 rows
use shop;
SELECT `order` .user_name, product_id, price, `count`, price * `count` as total_price FROM `order`
inner join order_products on order_products.order_id = `order` . id
inner join products on products.id = order_products.product_id;
100%
1:9
Result Grid
Filter Rows:
Search
Export:
user_name product_id price count total_price
Vasya 1 1999.00 1 1999.00
Vasya 2 1999.00 1 1999.00
Vasya 3 3999.00 2 7998.00
Petya 1 1999.00 1 1999.00
Petya 2 1999.00 2 3998.00
Petya 3 3999.00 1 3999.00
Sasha 1 1999.00 2 3998.00
Sasha 2 1999.00 1 1999.00
Sasha 3 3999.00 2 7998.00

3	Объединим по user_name (group by)	 **use shop;** **SELECT `order` .user_name, sum(price * `count`) as total_price FROM `order`** **inner join order_products on order_products.order_id = `order` . id** **inner join products on products.id = order_products.product_id** **group by `order` . user_name;**

4	Объединим с помощью group by по user name, min/max price, количество товаров в корзине, Итого сумма в корзине	 **use shop;** **SELECT `order` .user_name, max(price), min(price), sum(`count`), sum(price * `count`) as total_price FROM `order`** **inner join order_products on order_products.order_id = `order` . id** **inner join products on products.id = order_products.product_id** **group by `order` . user_name;**
19		**Ограничения для агрегированных данных HAVING. ИНДЕКСЫ для оптимизации работы DataBase**
№	**Действие**	**Команда на языке SQL и скрин результата**

1	Ограничения для агрегированных данных HAVING.	

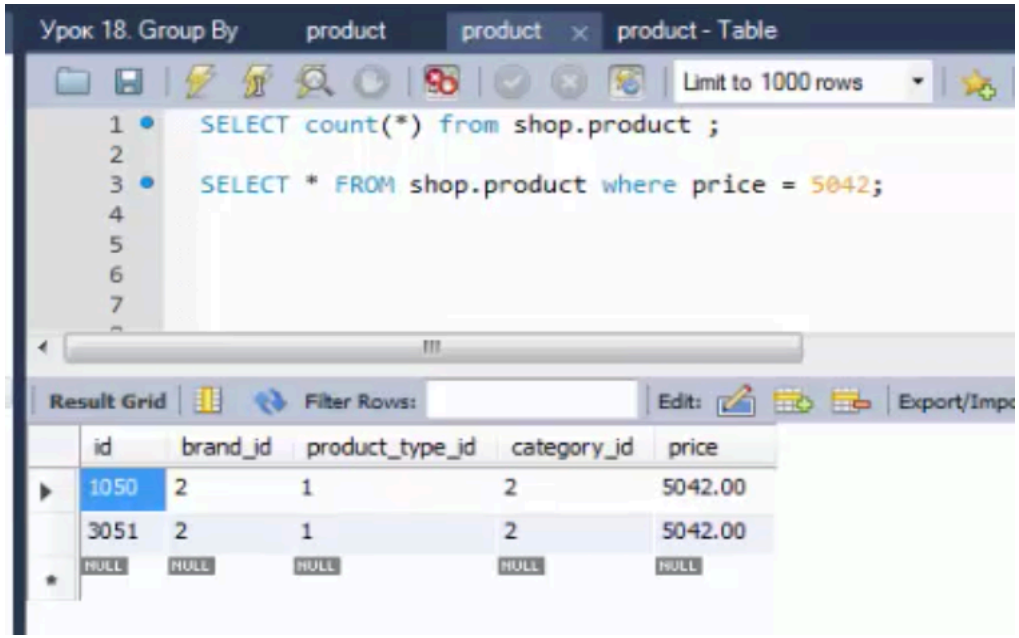
```
use shop;

SELECT `order` .user_name, max(price), min(price), sum(`count`), sum(price * `count`) as total_price  FROM `order`
   inner join order_products on order_products.order_id = `order` . id
   inner join products on products.id = order_products.product_id
   group by `order` . user_name

   having total_price >= 10000;
```

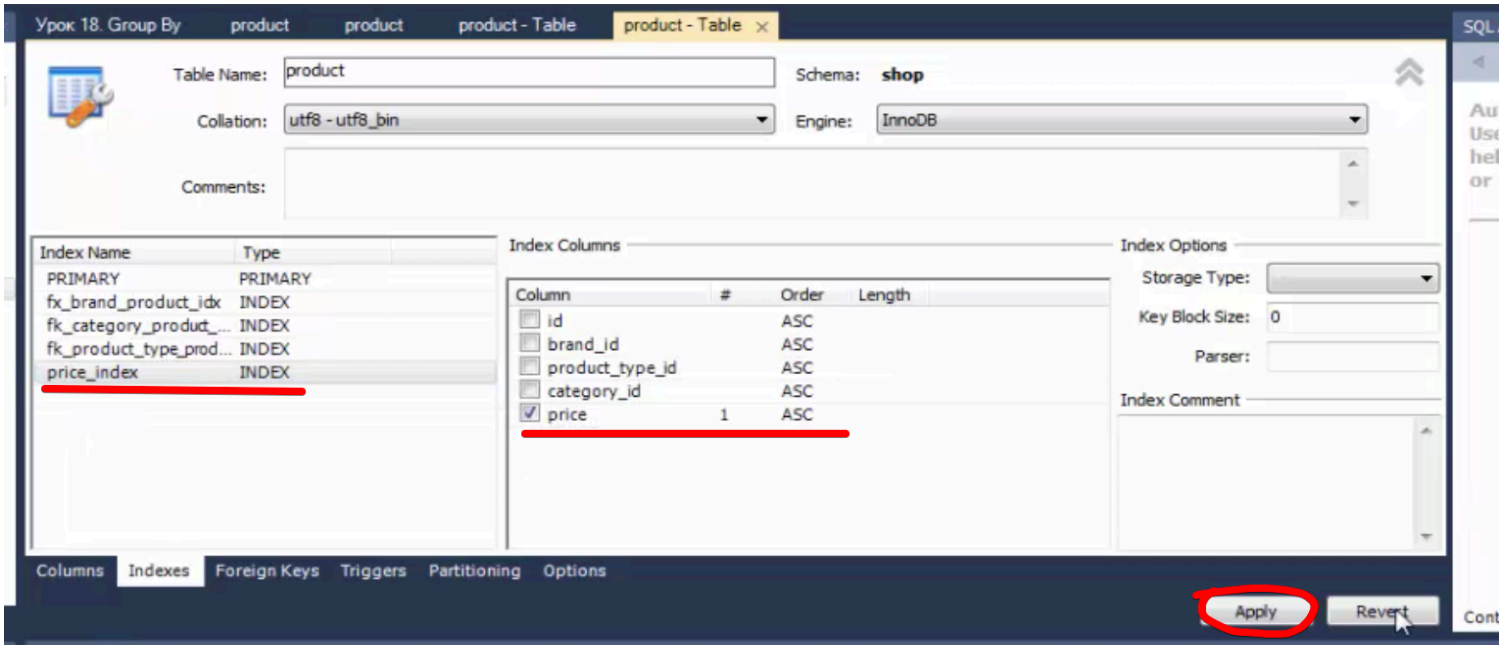
100% 5:9

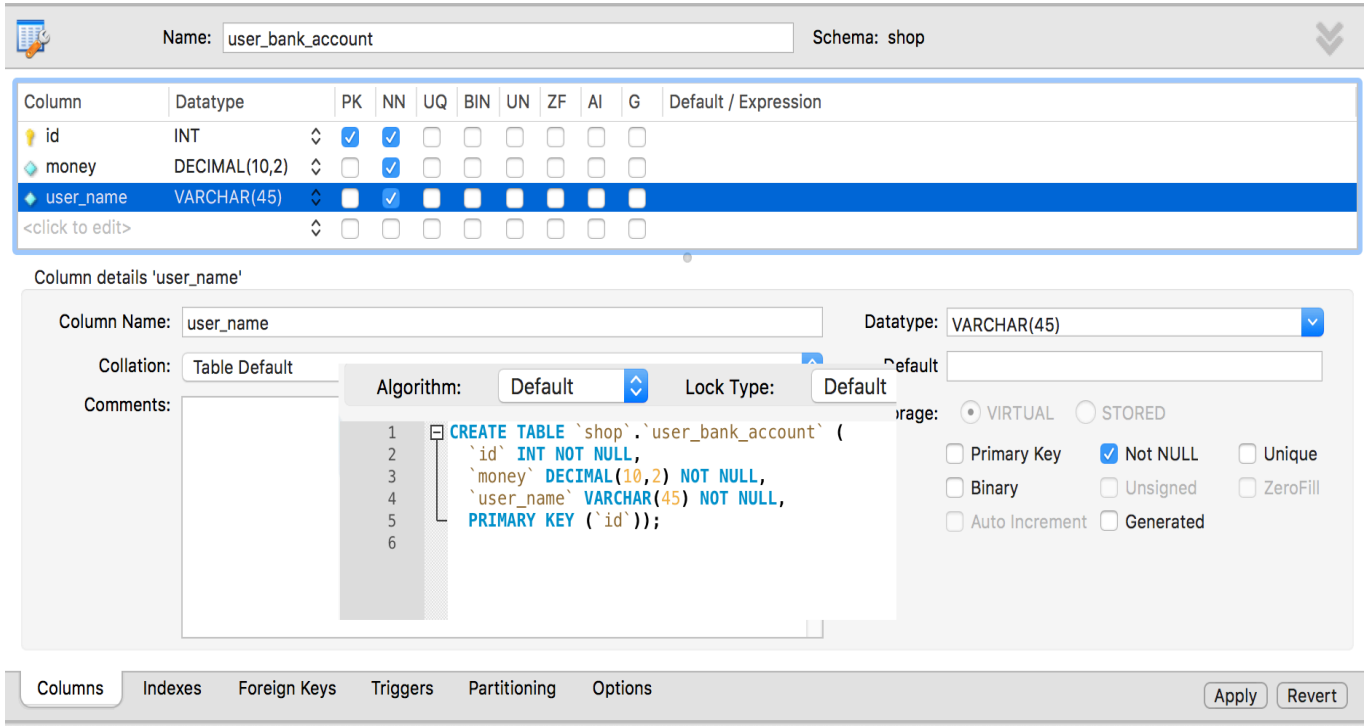
Result Grid | Filter Rows: Search | Export:

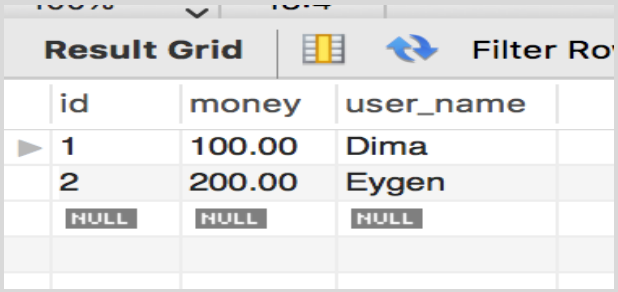
user_name	max(price)	min(price)	sum(`count`)	total_price
Sasha	3999.00	1999.00	5	13995.00
Vasya	3999.00	1999.00	4	11996.00

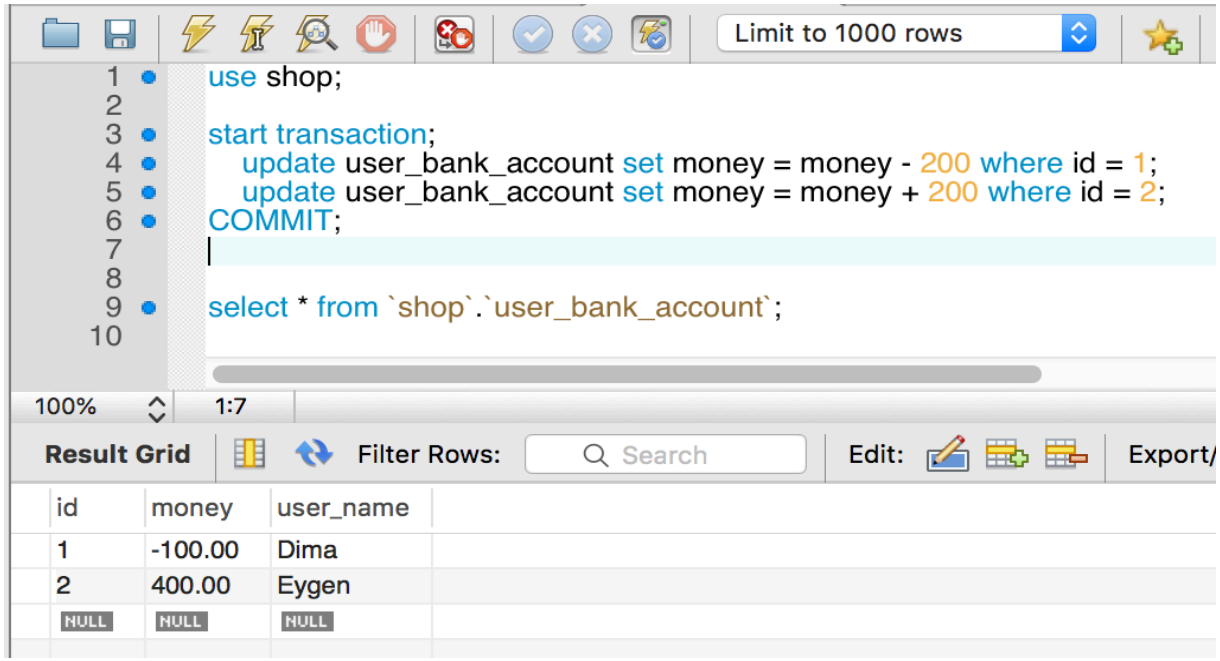
```
use shop;
SELECT `order` .user_name, max(price), min(price), sum(`count`), sum(price * `count`) as total_price  FROM `order`
 inner join order_products on order_products.order_id = `order` . id
      inner join products on products.id = order_products.product_id
      group by `order` . user_name
having total_price >= 10000;
```

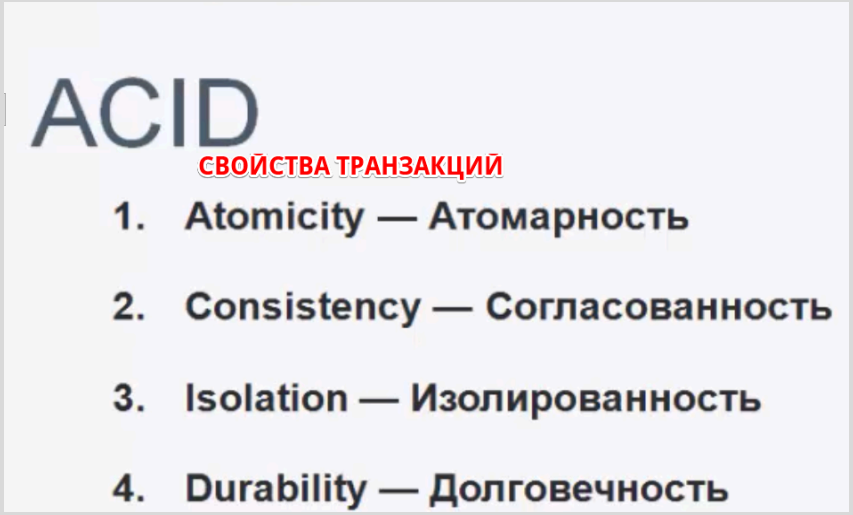
2	Индекс и производительность	Для ускорения производительности можно использовать индексацию.
3	Предположим база увеличилась	Для обработки такого запроса ушло 0,016sec 

4	Добавим индекс в таблицу Products-Altertable-Indexes	 Теперь для обработки запроса из пункта 3 уйдет 0,000 sec

20		Транзакции, на примере банковского перевода от одного Usera другому
№	Действие	Команда на языке SQL и скрин результата
1	Создаем Таблицу USER_BANK_ACCOUNT	

2	Добавим значения в таблицу USER_BANK_ACCOUNT	 **insert into `shop`.`user_bank_account` (id, money, user_name) values ('1', '100', 'Dima');** **insert into `shop`.`user_bank_account` (id, money, user_name) values ('2', '200', 'Eygen');** **select * from `shop`.`user_bank_account`;**

3	Сделаем транзакцию перевода 200 рублей от Димы Евгению.	 **use shop;** **start transaction;** **update user_bank_account set money = money - 200 where id = 1;** **update user_bank_account set money = money + 200 where id = 2;** **COMMIT;** **select * from `shop`.`user_bank_account`;**



Запись этих двух операций в совокупности будет гарантировать нам одновременное свершение транзакции, либо не выполнится ни одна в случае проблемы.