Курс: Основы реляционных баз данных. MySQL
Урок 5. Операторы, фильтрация, сортировка и ограничение. Агрегация данных
Выполнил: Кузнецов Сергей (Факультет Geek University Python-разработки)

---

Домашнее задание:

**Практическое задание по теме «Операторы, фильтрация, сортировка и ограничение»**

1. Пусть в таблице users поля created_at и updated_at оказались незаполненными. Заполните их текущими датой и временем.
2. Таблица users была неудачно спроектирована. Записи created_at и updated_at были заданы типом VARCHAR и в них долгое время помещались значения в формате 20.10.2017 8:10. Необходимо преобразовать поля к типу DATETIME, сохранив введённые ранее значения.
3. В таблице складских запасов storehouses_products в поле value могут встречаться самые разные цифры: 0, если товар закончился и выше нуля, если на складе имеются запасы. Необходимо отсортировать записи таким образом, чтобы они выводились в порядке увеличения значения value. Однако нулевые запасы должны выводиться в конце, после всех.
4. **(по желанию)** Из таблицы users необходимо извлечь пользователей, родившихся в августе и мае. Месяцы заданы в виде списка английских названий ('may', 'august')
5. **(по желанию)** Из таблицы catalogs извлекаются записи при помощи запроса. SELECT * FROM catalogs WHERE id IN (5, 1, 2); Отсортируйте записи в порядке, заданном в списке IN.

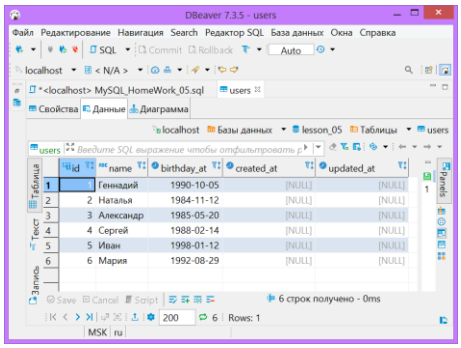
Практическое задание теме «Агрегация данных»

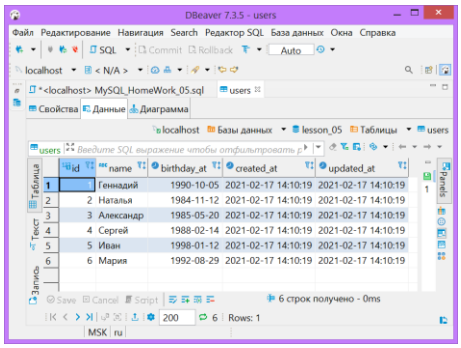
6. Подсчитайте средний возраст пользователей в таблице users.
7. Подсчитайте количество дней рождения, которые приходятся на каждый из дней недели. Следует учесть, что необходимы дни недели текущего года, а не года рождения.
8. (по желанию) Подсчитайте произведение чисел в столбце таблицы.

---

## 1. *Пусть в таблице users поля created_at и updated_at оказались незаполненными. Заполните их текущими датой и временем.*

Прежде всего, все команды работы с БД находятся в файле «*MySQL_HomeWork_05.sql*». Создадим БД «*CREATE DATABASE IF NOT EXISTS lesson_05;*» , сделаем текущей «*USE lesson_05;*» и создав, заполним данные таблицы «*users*». Для заполнения полей, достаточно выполнить оператор «*UPDATE users SET created_at = now(), updated_at = now();*».



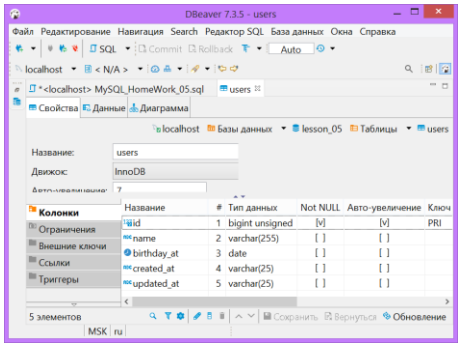
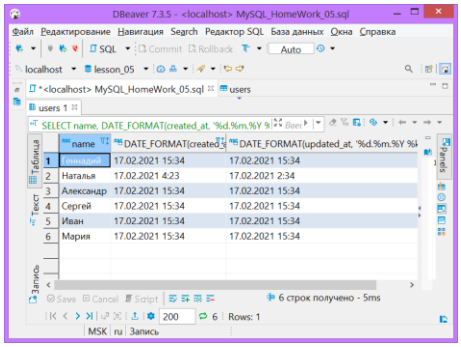
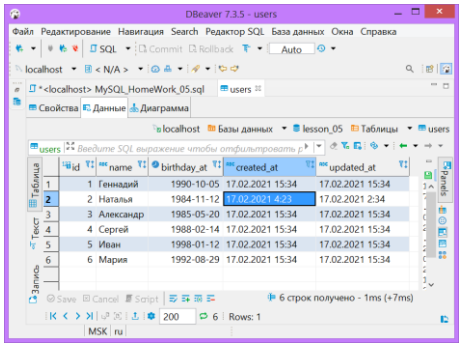




## 2. *Таблица users была неудачно спроектирована. Записи created_at и updated_at были заданы типом VARCHAR и в них долгое время помещались значения в формате 20.10.2017 8:10. Необходимо преобразовать поля к типу DATETIME, сохранив введённые ранее значения.*

Действительно, кто-то недальновидно вместо DATETIME или, на «худой конец», TIMESTAMP использовал VARCHAR.

Произведем переформатирование в «привычный формат» (dd.mm.yyyy hh:mm). Для начала, проверим корректность «*SELECT name, DATE_FORMAT(created_at, '%d.%m.%Y %k:%i'), DATE_FORMAT(updated_at, '%d.%m.%Y %k:%i') FROM users;*» - все Ок, даже ведущий ноль у часов убрали (%k).

Выполним «*UPDATE users SET created_at = DATE_FORMAT(created_at, '%d.%m.%Y %k:%i'), updated_at = DATE_FORMAT(updated_at, '%d.%m.%Y %k:%i');*».

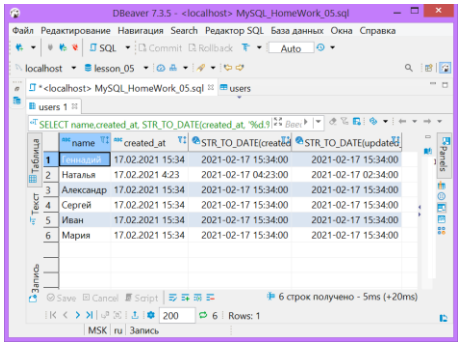
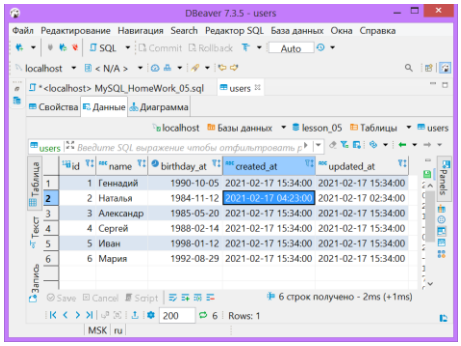
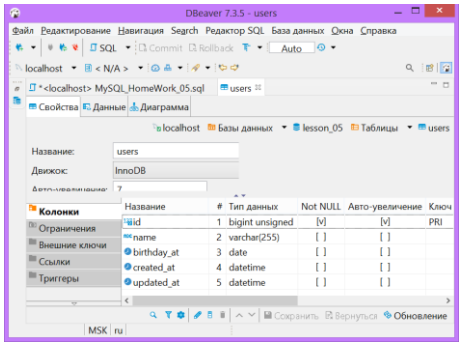
Для преобразования поля к формату DATETIME можно, либо:

- создать временные столбцы в таблице типа DATETIME, перенести туда значения с преобразованием, и снова изменить таблицу, удалив «старые» столбцы и переименовать «новые»;
- создать временную таблицу, скопировать туда значения столбцов, изменить тип столбцов в users и залить туда значения;
- преобразовать значение в столбцах, согласно принятому в mySQL и изменить тип столбцов;

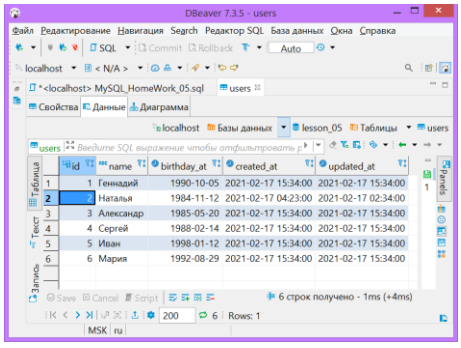
Выбираем последний вариант, как наименее затратный. Для начала, проверим корректность « *SELECT name,created_at, STR_TO_DATE(created_at, '%d.%m.%Y %k:%i'), STR_TO_DATE(updated_at, '%d.%m.%Y %k:%i') FROM users;*» - все Ok.

Выполним « *UPDATE users SET created_at = STR_TO_DATE(created_at, '%d.%m.%Y %k:%i'), updated_at = STR_TO_DATE(updated_at, '%d.%m.%Y %k:%i');*».

Произведем изменение в описании столбцов:
«*ALTER TABLE users MODIFY COLUMN created_at DATETIME DEFAULT CURRENT_TIMESTAMP;*»
«*ALTER TABLE users MODIFY COLUMN updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP;*»

Данные сохранились.



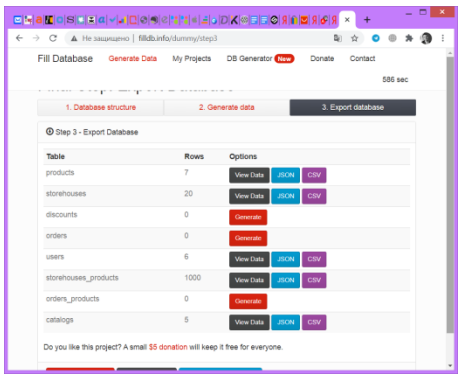
## *3. В таблице складских запасов storehouses_products в поле value могут встречаться самые разные цифры: 0, если товар закончился и выше нуля, если на складе имеются запасы. Необходимо отсортировать записи таким образом, чтобы они выводились в порядке увеличения значения value. Однако нулевые запасы должны выводиться в конце, после всех.*

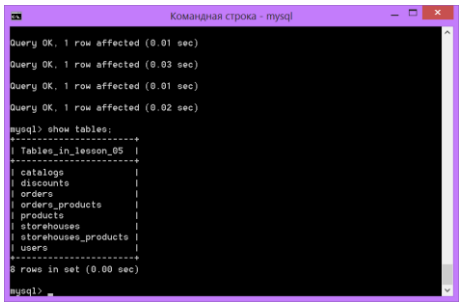
Используя сервис http://filldb.info, заполним предлагаемую БД. Не будем заморачиваться и уделим внимание только:

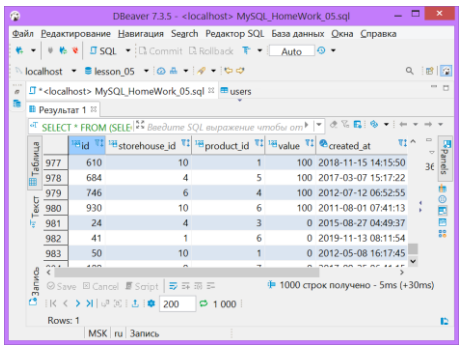
- Складам «storehouses», связанным с основной таблицей запроса – 20шт
- основной таблице «storehouses_products» - 1000шт, ForeignKey с «products» «storehouses». Для поля «value» выберем рандомный интервал от -1 до 100 (нижняя граница не включается, поэтому выборка начинается от 0).

Загрузим дамп в базу данных «*SOURCE fulldb17-02-2021 13-58.sql*» (зашел в mysql, сделал текущей «*USE lesson_05*» и запустил скрипт). Проверим «*SHOW TABLES;*» - таблицы созданы.

Самый простой вариант выполнить заданный запрос «*SELECT * FROM (SELECT * FROM storehouses_products WHERE value <> 0 ORDER BY value LIMIT 1000) AS a UNION SELECT * FROM storehouses_products WHERE value = 0;*». Обратите внимание на «*LIMIT 1000*». Без указания этого параметра, записи в итоговой выборке получаются не отсортированными по значению в поле «value».



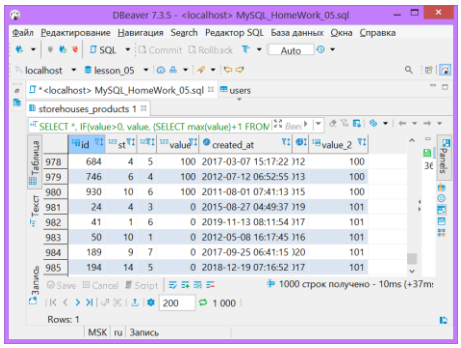



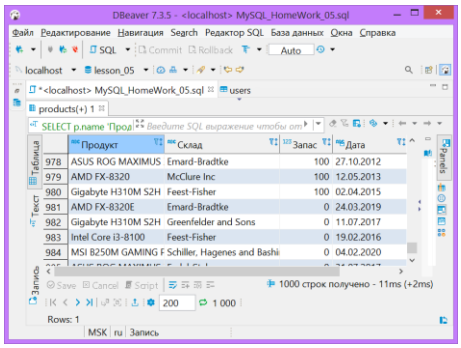




С моей т.з. более элегантно выглядит конструкция с вычисляемым полем и сортировкой по нему «*SELECT *, IF(value>0, value, 2147483647) AS value_2 FROM storehouses_products ORDER BY value_2;*» (2147483647 - максимальное значение INT, вместо него можно было указать «*(SELECT max(value)+1 FROM storehouses_products)*»)

Естественно, «по-нормальному» запрос должен выглядеть со значениями из связанных таблиц: «*SELECT p.name 'Продукт', s.name 'Склад', sp.value 'Запас', DATE_FORMAT(sp.updated_at, GET_FORMAT(DATE, 'EUR')) 'Дата', IF(value>0, value, 2147483647) AS value_2 FROM storehouses_products sp, storehouses s, products p WHERE sp.storehouse_id = s.id AND sp.product_id = p.id ORDER BY value_2;*»

Кстати, дата приведена к Европейскому формату…

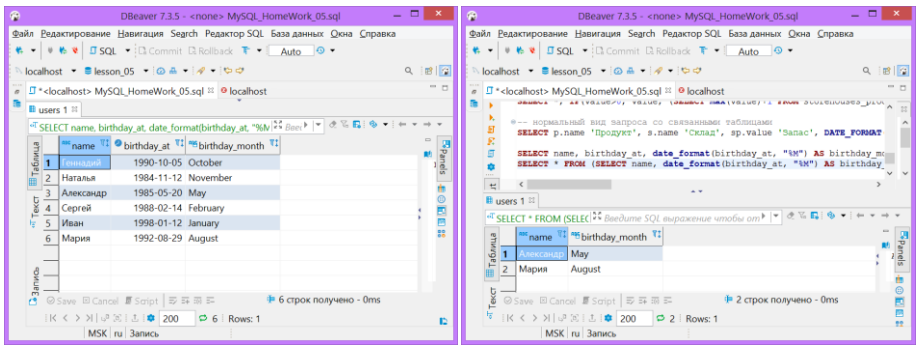


## 4. Из таблицы users необходимо извлечь пользователей, родившихся в августе и мае. Месяцы заданы в виде списка английских названий ('may', 'august')

Для начала, сделаем простой запрос, дающий нам значение месяца рождения: «*SELECT name, birthday_at, date_format(birthday_at, "%M") AS birthday_month FROM users;*»

Теперь, просто добавим фильтр «*SELECT * FROM (SELECT name, date_format(birthday_at, "%M") AS birthday_month FROM users) AS u WHERE birthday_month IN ('may', 'august');*»



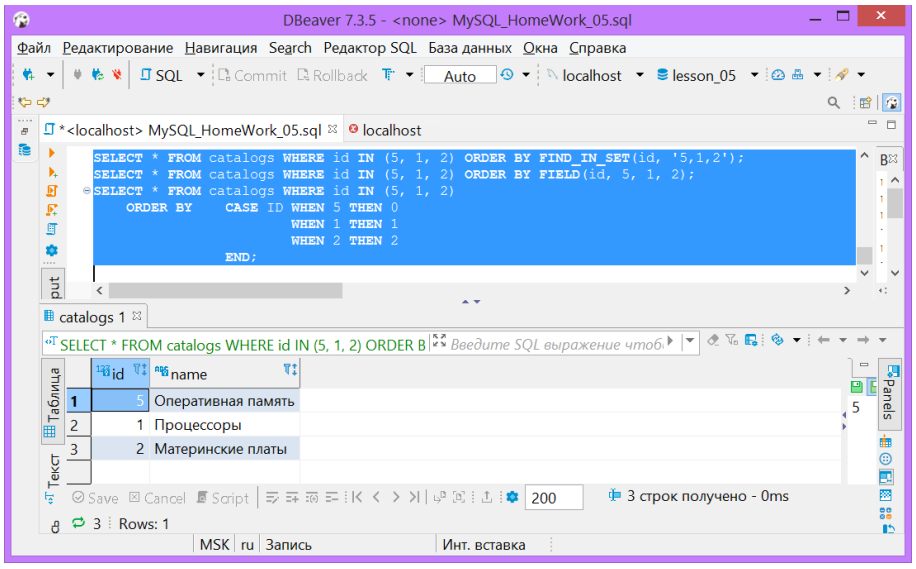
## 5. Из таблицы catalogs извлекаются записи при помощи запроса. SELECT * FROM catalogs WHERE id IN (5, 1, 2); Отсортируйте записи в порядке, заданном в списке IN

Все достаточно просто, воспользуемся функцией «*FIND_IN_SET()*» которая возвращает положение строки в списке строк «*SELECT * FROM catalogs WHERE id IN (5, 1, 2) ORDER BY FIND_IN_SET(id, '5,1,2');*».

Ничуть не хуже работает функция «*FIELD(value, val1, val2, val3, ...)*», которая возвращает индексную позицию в строковом списке «*SELECT * FROM catalogs WHERE id IN (5, 1, 2) ORDER BY field(id, 5, 1, 2);*»

Есть и самый «дебильный» способ – при помощи конструкции «*CASE*»:

```
SELECT * FROM catalogs WHERE id IN (5, 1, 2) ORDER BY CASE ID
                                                        WHEN 5 THEN 0
                                                        WHEN 1 THEN 1
                                                        WHEN 2 THEN 2
                                                    END;
```

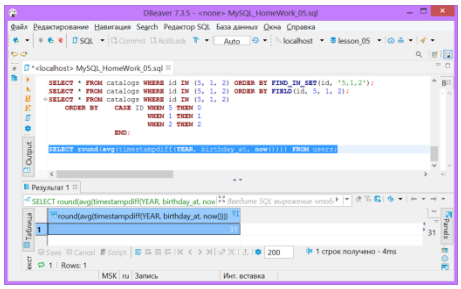
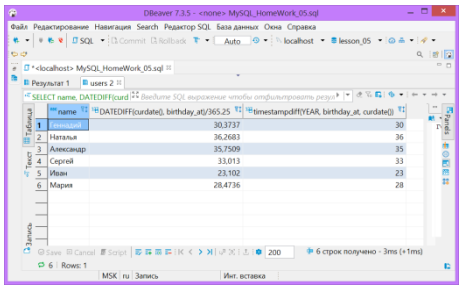
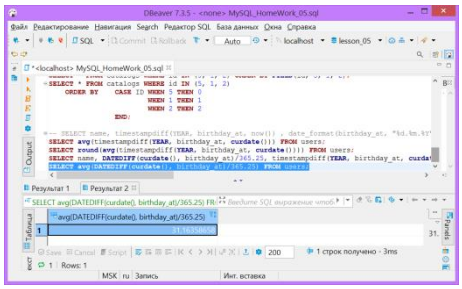


## 6. Подсчитайте средний возраст пользователей в таблице users.

Если не занудствовать, то запрос «*SELECT avg(timestampdiff(YEAR, birthday_at, curdate())) FROM users;*» выдаст нам что-то типа «30,8333». Возникает вопрос, что в разговорной речи мы обычно называем целые числа, т.е требуется округление: «*SELECT round(avg(timestampdiff(YEAR, birthday_at, curdate()))) FROM users;*» - даст «31».

Но, осталась небольшая доля вероятности, что мы посчитали не совсем точно. При подсчете возраста каждого человека, мы использовали «*timestampdiff*» с округлением и ошибка могла накопиться. Как вариант, мы можем посчитать общую сумму возраста сотрудников в днях и поделить на ср. количество дней в году с количеством сотрудников. Запрос «*SELECT name, DATEDIFF(curdate(), birthday_at)/365.25, timestampdiff(YEAR, birthday_at, curdate()) FROM users;*» демонстрирует разницу в этих подсчетах по каждой строке.

Теперь произведем агрегацию «*SELECT avg(DATEDIFF(curdate(), birthday_at)/365.25) FROM users;*» - получили «31.16358658» , т.е. погрешность составила 0,33 – весьма немаленькая (могла с большой долей вероятности при округлении дать др. ответ по ср. возрасту.

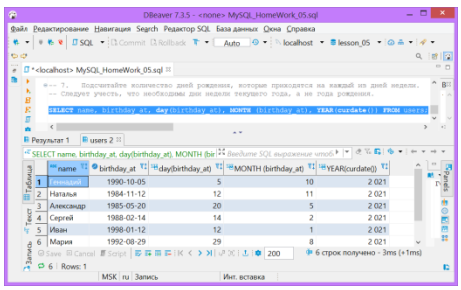
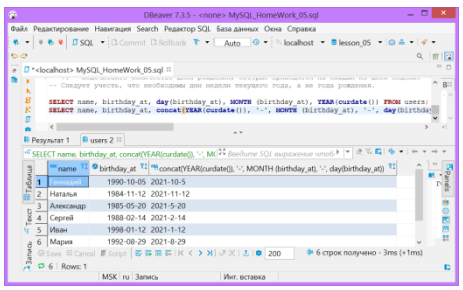
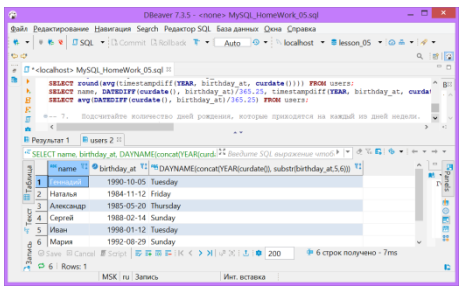




## 7. Подсчитайте количество дней рождения, которые приходятся на каждый из дней недели. Следует учесть, что необходимы дни недели текущего года, а не года рождения.

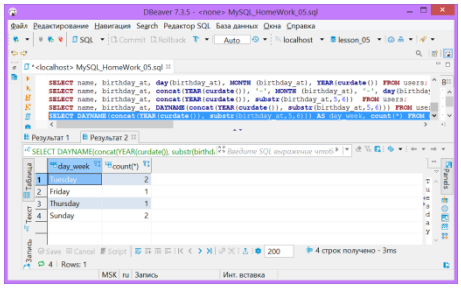
Для получения даты дня рождения в текущем году, воспользуемся функциями преобразования. Можно было воспользоваться функцией EXTRACT(type FROM date), но мы выбрали специализированные… Проверим «*SELECT name, birthday_at, DAY(birthday_at), MONTH (birthday_at), YEAR(curdate()) FROM users;*».

Произведем перевод полученных данных в дату
«*SELECT name, birthday_at, concat(YEAR(curdate()), '-', MONTH (birthday_at), '-', day(birthday_at)) FROM users;*»
хотя мне больше импонирует запрос:
«*SELECT name, birthday_at, concat(YEAR(curdate()), substr(birthday_at,5,6)) FROM users;*»

Для получения дня недели по дате, используются функции DAYNAME(date), DAYOFWEEK(date) и WEEKDAY(date). В зависимости от задания, мы выводим название дня недели или его порядковый номер. Для получения даты дня рождения в текущем году, воспользуемся преобразованием. Выполним запрос «*SELECT name, birthday_at, DAYNAME(concat(YEAR(curdate()), substr(birthday_at,5,6))) FROM users;*»

Осталось произвести агрегацию по столбцу
«*SELECT DAYNAME(concat(YEAR(curdate()), substr(birthday_at,5,6))) AS day_week, count(*) FROM users GROUP BY day_week;*»

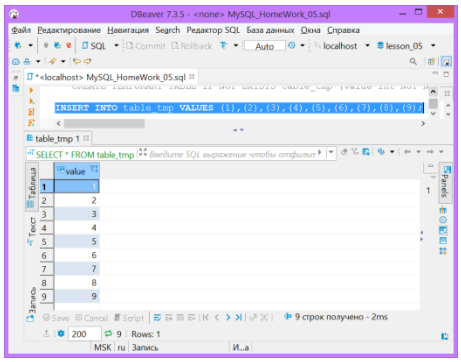
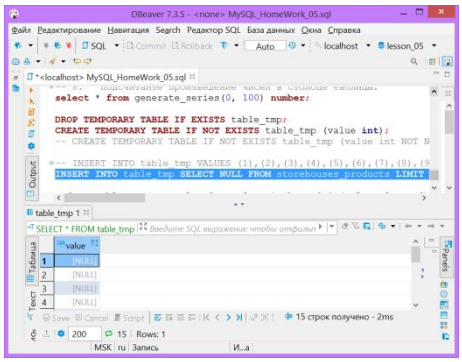
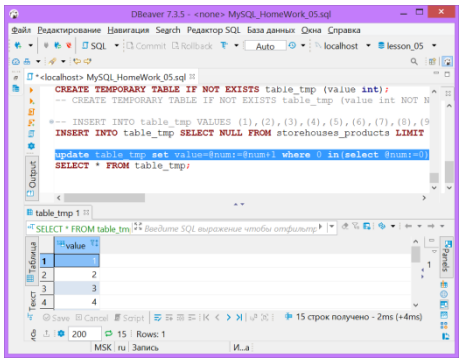


## 8. *Подсчитайте произведение чисел в столбце таблицы.*

Для начала, заполним таблицу значениями. Можно просто взять готовую таблицу и скопировать/ переименовать/ использовать так…создать новую и заполнить используя сервис http://filldb.info,… В общем вариантов масса. Мы создадим временную таблицу «*CREATE TEMPORARY TABLE IF NOT EXISTS table_tmp (value int);*». Заполняем ее произвольным количеством записей. Можно было пойти по экстенсивному пути «*INSERT INTO table_tmp VALUES (1),(2),(3),(4),(5),(6),(7),(8),(9);*», но мы простых путей не ищем.

Заполним таблицу пустыми значениями, используя ссылку на «большую» таблицу запросом «*INSERT INTO table_tmp SELECT NULL FROM storehouses_products LIMIT 15;*»

И присвоим последовательные значения от 1,2,3,…n, где n-число строк в таблице.
«*UPDATE table_tmp SET value=@num:=@num+1 WHERE 0 IN (SELECT @num:=0);*»
В принципе, достаточно было объявить поле «value» как «*…AUTO_INCREMENT, PRIMARY KEY…*» и система сама заполнит последовательные значения…

Теперь само задание. Есть «старинный» способ «*SELECT round(exp(sum(log(value)))) FROM table_tmp;*». Это из матанализа: логарифм произведения равен сумме логарифмов. Будем использовать натуральный логарифм LN() в паре с функцией возведения экспоненты в степень EXP(). Проверил в интернете, факториал «15» действительно равен «1 307 674 368 000».

Тут стоит заметить, что решение не идеальное. К примеру, если в какой-то строке будет значение «0», то решение выдаст неверный ответ. Мы поменяли в строке с «1» на значение «0» «*UPDATE table_tmp SET value=0 WHERE value = 1;*» и снова выполнили команду «*SELECT round(exp(sum(log(value)))) FROM table_tmp;*» - ответ остался прежним «1 307 674 368 000», что НЕВЕРНО!!!