КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Пинягина О.В.

Практикум по курсу «Хранилища данных»

Казань – 2022

УДК 004.6
ББК 32.973.26 – 018.2

Печатается по решению учебно-методической комиссии
Института вычислительной математики и информационных технологий
протокол № 7 от 31.03.2022 г.

Рецензенты:

Доцент кафедры «Информатика и информационно-управляющие системы» Института цифровых технологий и экономики Казанского государственного энергетического университета, к.ф.-м.н. **Шустова К.П.**

Старший преподаватель кафедры системного анализа и информационных технологий ИВМиИТ КФУ **Рубцова Р.Г.**

Пинягина О.В.

Практикум по курсу «Хранилища данных» / О.В. Пинягина – Казань: Казанский университет, 2022. – 126 с.

Данное учебное пособие разработано для поддержки компьютерных лабораторных занятий и самостоятельной работы по курсу «Хранилища данных» для студентов, обучающихся по специальностям «Бизнес-информатика», «Прикладная математика и информатика».

В качестве среды программирования используется **Microsoft SQL Server** и **Loginom Community** или **Deductor Studio Academic**.

Оглавление

Оглавление 3

Работа в программе SQL Server 5

Этап 1: Описание предметной области, разработка модели, создание таблиц. 5

Задание 1. 9

Этап 2. Заполнение базы данными. 10

Задание 2. 19

Этап 3. Запросы, представления, а также хранимые процедуры. 20

Задание 3. 29

Этап 4. Очистка данных 30

Задание 4. 33

Этап 5. Обогащение данных 34

Задание 5. 37

Работа в программе Loginom 38

Этап 6. Подключение к SQL server, обработка данных и визуализация 38

Задание 6. 62

Этап 7. Анализ данных в Loginom 63

Задание 7. 74

Работа в программе Deductor Studio 79

Этап 6а. Выгрузка данных в текстовые файлы 79

Задание 6а. 82

Этап 6б: Загрузка данных в Deductor Studio из текстовых файлов, создание хранилища данных в СУБД FireBird. 83

Задание 6б. 96

Этап 6в: Преобразования и визуализаторы. 97

Задание 6в. 117

Этап 7. Анализ данных в Deductor Studio 118

Задание 7. 125

Литература 126

Интернет-источники 126

Работа в программе SQL Server

Этап 1: Описание предметной области, разработка модели, создание таблиц.

Постановка задачи:

В базе данных «Аптеки» накапливается оперативная информация по реализации лекарств и других товаров через аптечную сеть, состоящую из нескольких аптек, расположенных в нескольких городах нескольких регионов. Каждый товар имеет фирму-производителя, тип, вид фасовки, текущую цену. В один чек может быть внесено несколько товаров. В зависимости от общей суммы покупки делается скидка по следующему правилу: от 1000 до 5000 р. – 2%, более 5000 р. – 5%.

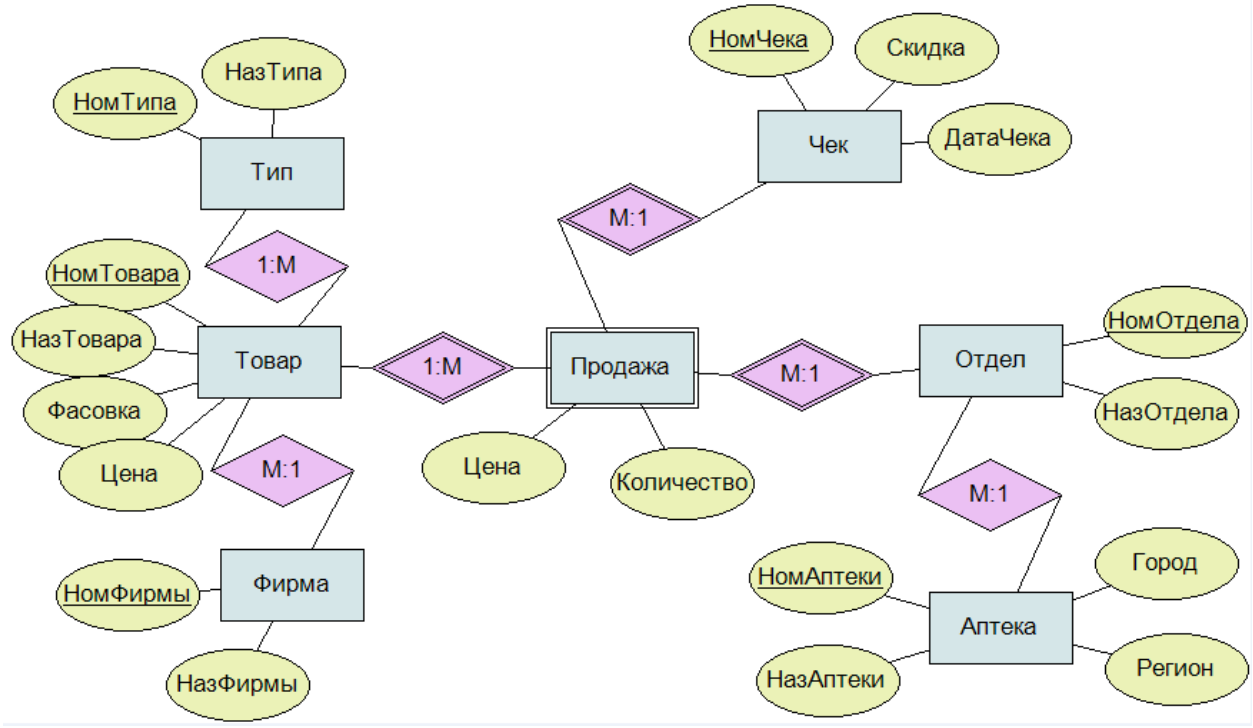
По мере накопления оперативной информации возникает необходимость её анализа. У пользователей появляются вопросы:

- Сколько единиц каждого товара и на какую сумму продано в каждой аптеке?
- Сколько единиц каждого **типа** товаров и на какую сумму продано
- в каждой аптеке?
- На какую сумму продано лекарств «от гриппа» в каждой аптеке в порядке убывания итоговой суммы?
- А можно получить суммы продаж «от гриппа» по месяцам в порядке убывания итогов?
- А если по каждой аптеке отдельно?
- А сколько реализовано товара с названием «Товар 117» в «Аптеке 10» за август текущего года?
- А можно получить такую таблицу, чтобы столбцами были, например, месяцы, а в строках находились типы товаров и итоги продаж по месяцам?
- А можно построить графики или диаграммы для продаж за текущий год? А по отдельному товару? А сразу по нескольким товарам, для сравения?
- А можно построить прогноз продаж на следующий месяц?

И т.п.

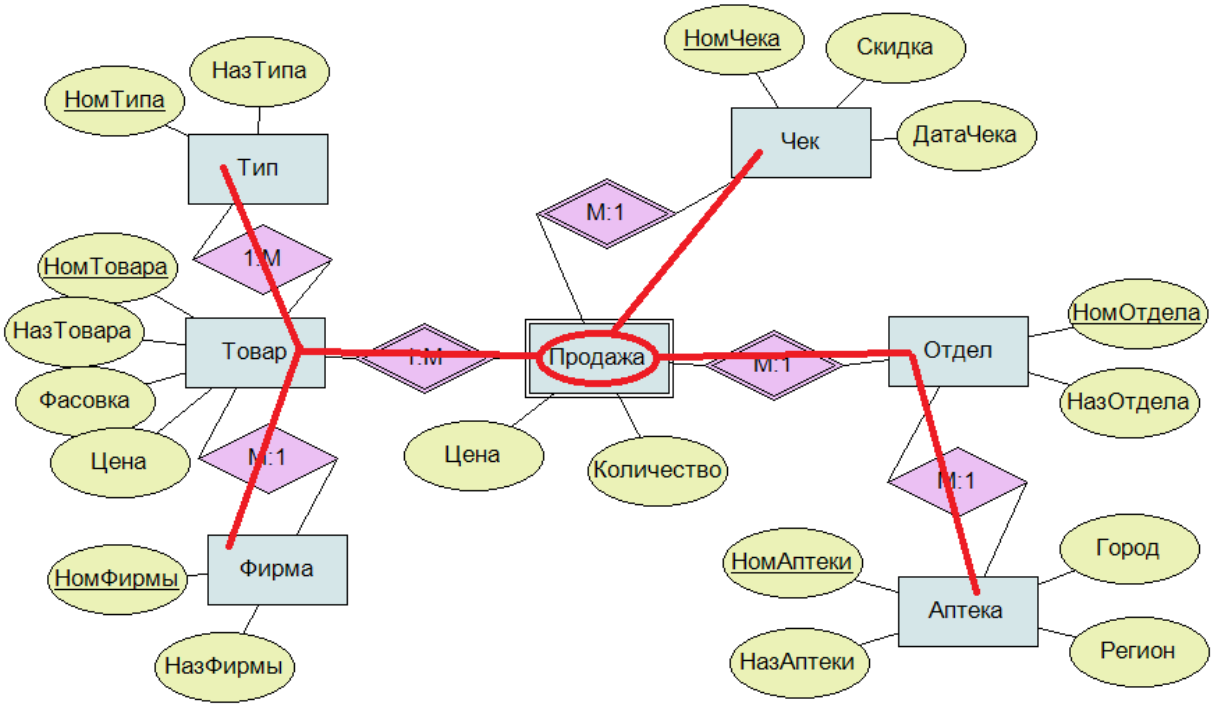
Как видим, здесь мы рассматриваем **основной бизнес-процесс** аптечной сети – продажу товаров покупателям. Разумеется, у организации имеются и **другие** бизнес-процессы: оптовая покупка товаров у поставщиков, учет сотрудников, покупка оборудования и т.п.

Начнем реализацию задачи с построения ER-модели:

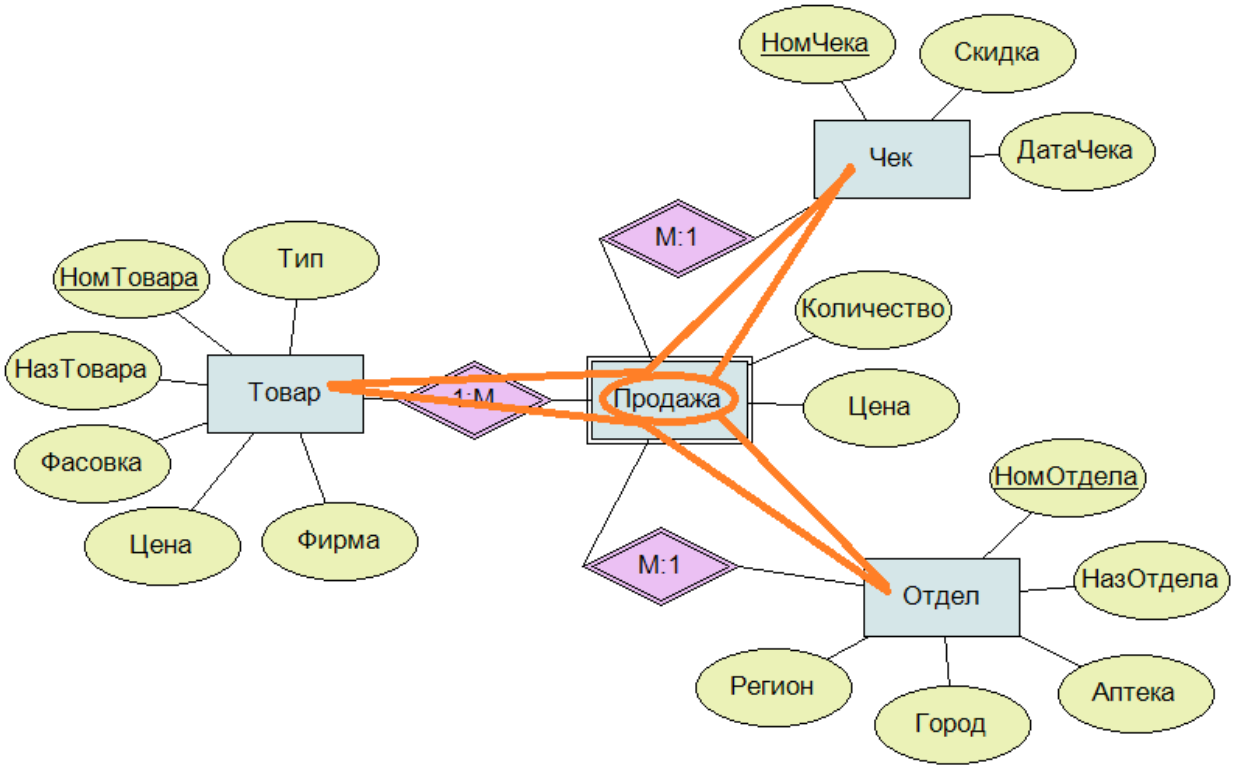


Для этой базы данных используется модель «**Снежинка**». Центральным набором объектов (набором **событий**, или **фактов**) является «Продажа». Она представляет собой **слабую сущность**, которая зависит от сущностей (**измерений**) «Чек», «Товар» и «Отдел». В свою очередь, сущность «Товар» ссылается на измерения «Фирма-производитель» и «Тип товара», а сущность «Отдел» ссылается на сущность «Аптека».

В модели «Снежинка» всегда имеется некоторая сущность (набор фактов), которая является **центром**, от нее исходят лучи к сущностям следующего уровня (измерениям), от которых также могут исходить лучи к сущностям третьего уровня (измерениям), и т.п. – получается ветвящаяся структура, отдаленно напоминающая снежинку:



Структура «Звезда» проще: здесь есть только центр (факты) и лучи, указывающие на один уровень измерений:

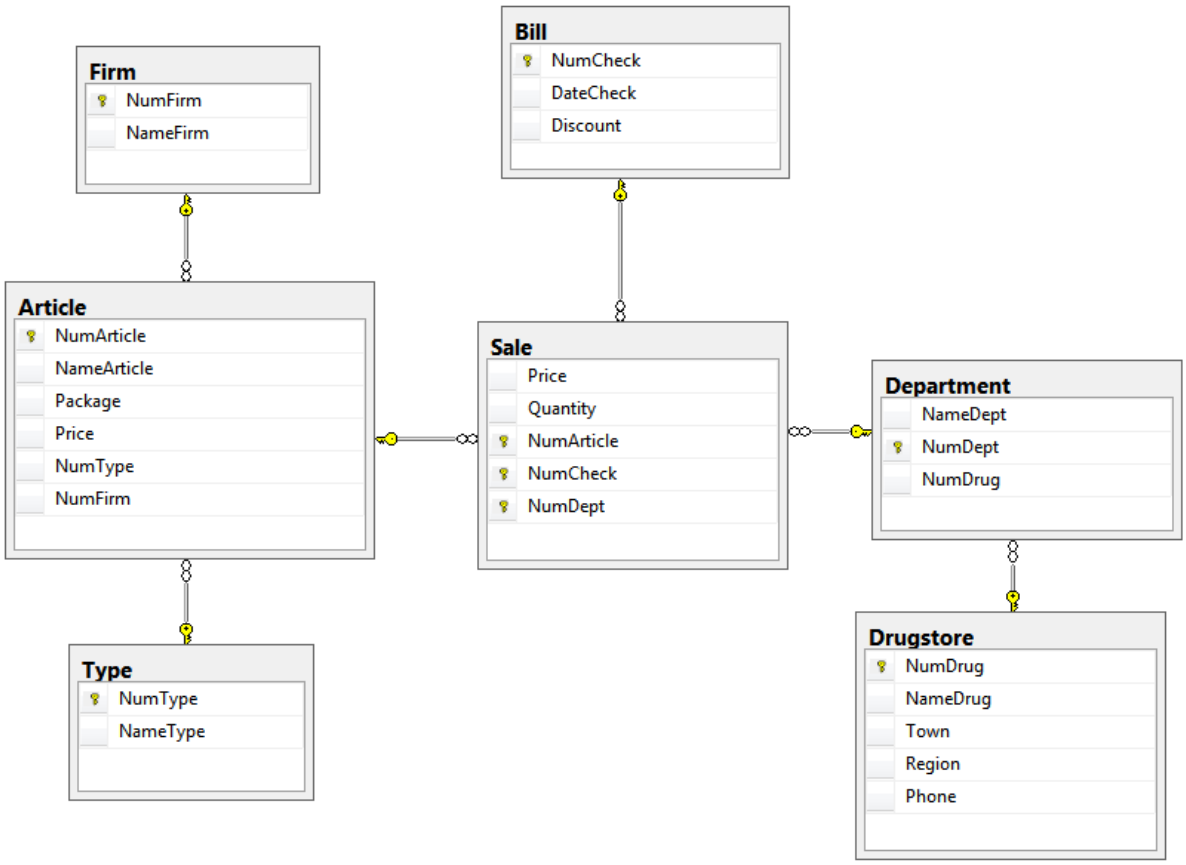


Итак, независимо от вида модели («снежинка» или «звездочка») **центр** представляет собой **слабую сущность** или связь «**многие ко многим**»

(которая, как вы знаете, есть скрытая слабая сущность). А все остальные связи имеют тип «**многие-к-одному**» (в редких случаях «один-к-одному») и направлены из центра наружу. Если в вашей модели имеется **несколько** связей «многие-ко-многим», либо какие-то связи «один-ко-многим» направлены из центра наружу, это означает, что вы рассматриваете не **один**, а **несколько** бизнес-процессов, т.е., ваша схема представляет собой **несколько** «снежинок» и/или «звездочек», обычно «склеившихся» лучами.

Далее следует написать сценарий создания таблиц базы данных на языке SQL и выполнить его в среде **SQL server management studio**. Рекомендуется для названия баз данных, таблиц, столбцов и т.п. **не использовать русские буквы.**

После создания базы данных проведем «обратное проектирование» - создадим в SQL server диаграмму базы данных:



Убедимся, что полученная диаграмма по структуре соответствует нашей ER-модели.

Задание 1. Выберите предметную область, подходящую для разработки хранилища данных. Выберите в этой предметной области один бизнес-процесс. *(Можете выбрать несколько бизнес-процессов, но при этом вырастет трудоемкость всех заданий!)* Обратите внимание, что в описании бизнес-процесса вашей модели обязательно должны присутствовать атрибуты типа «дата» и/или «время».

Примерная схема может быть такой: оперативная информация хранится в слабой сущности, плюс 3 или более сильных сущности (модель «звезда»), либо ещё плюс дополнительные справочники (модель «снежинка»). Опишите предметную область (5 баллов).

Создайте таблицы в SQL Server (5 баллов). После создания БД проведите «reverse engineering» (обратное проектирование) - создайте диаграмму базы данных с помощью автоматизированных средств SQL Server. ER-модель сдавать не обязательно, достаточно диаграммы из SQL Server.

Итого 10 баллов.

Этап 2. Заполнение базы данными.

Реальные данные в масштабах тысяч записей нам взять негде. Поэтому будем генерировать их искусственно. Следует сгенерировать не менее 10 000 записей для оперативных данных и, по крайней мере, по 10-100 записей для остальных таблиц. Для генерации данных можно использовать хранимые процедуры, а также загружать данные из внешних источников. Рассмотрим все эти возможности.

Пример реализации:

Таблицу **«Тип»** заполним вручную:

```
SELECT * FROM Type
```

Результаты | Сообщения

	NumTy...	NameType
1	1	антибиотик
2	2	витамины
3	3	жаропонижающее
4	4	обезболивающее
5	5	противодиарейное
6	6	от кашля
7	7	против гриппа

Для заполнения таблицы **«Фирма»** разработаем хранимую процедуру, которая создает 50 строк примерно такого вида:

Результаты | Сообщения

	NumFirm	NameFirm
1	1	Фирма 1
2	2	Фирма 2
3	3	Фирма 3
4	4	Фирма 4
5	5	Фирма 5
6	6	Фирма 6
7	7	Фирма 7
8	8	Фирма 8
9	9	Фирма 9
10	10	Фирма 10

```
CREATE PROC insert_firms AS
DECLARE @nom INT
SET @nom=1
WHILE @nom<=50
```

```
BEGIN
    INSERT INTO Firm (NumFirm, NameFirm)
        VALUES (@nom, 'Фирма '+LTRIM(STR(@nom)))
    SET @nom=@nom+1
END
```

Пусть в нашей базе будет 2 региона, 10 городов, 50 аптек и 200 отделов.

Для заполнения таблицы **«Аптека»** разработаем хранимую процедуру, которая создает 50 строк примерно такого вида:

Результаты | Сообщения

	NumDr...	NameDrug	Town	Region	Phone
1	1	Аптека 1	Город 1	Регион 1	525017
2	2	Аптека 2	Город 1	Регион 1	949597
3	3	Аптека 3	Город 1	Регион 1	107967
4	4	Аптека 4	Город 1	Регион 1	409099
5	5	Аптека 5	Город 1	Регион 1	289814
6	6	Аптека 6	Город 2	Регион 1	450039
7	7	Аптека 7	Город 2	Регион 1	787675
8	8	Аптека 8	Город 2	Регион 1	811716
9	9	Аптека 9	Город 2	Регион 1	553691
10	10	Аптека 10	Город 2	Регион 1	534565
11	11	Аптека 11	Город 3	Регион 1	767481
12	12	Аптека 12	Город 3	Регион 1	247438
13	13	Аптека 13	Город 3	Регион 1	292790

Пусть в каждом городе у нас по 5 аптек, города с номерами 1 – 3 относятся к первому региону, города с номерами 4 – 10 – ко второму региону.

Здесь поле «Телефон» заполнено псевдослучайными равномерно распределенными числами в диапазоне от 100000 до 999999. Для этого используется функция **RAND()**, которая возвращает значение с плавающей точкой, равномерно распределенное от 0 до 1. Для того чтобы эта функция в рамках процедуры всегда генерировала **один и тот же** набор случайных чисел, следует ее предварительно вызвать с параметром-константой, например: **SET @x=RAND(1)**

Итак, функция **RAND** возвращает случайное значение из отрезка **[0, 1]**. Для того чтобы получить, например, значение из отрезка [100, 150], следует умножить полученную величину на 50 (длина отрезка) и прибавить 100 (левый конец отрезка).

Для преобразования чисел с плавающей точкой к целому типу удобно использовать функцию **CEILING** (это «потолок» по-русски), которая преобразует свой аргумент к ближайшему **большему** целому числу. Есть также симметричная ей функция **FLOOR,** которая преобразует аргумент к ближайшему **меньшему** целому числу.

Далее заполним таблицу **«Отдел»** (в каждой аптеке по 4 отдела):

Результаты | Сообщения

	NumDept	NameDept	NumDrug
1	1	Отдел 1	1
2	2	Отдел 2	1
3	3	Отдел 3	1
4	4	Отдел 4	1
5	5	Отдел 5	2
6	6	Отдел 6	2

Для заполнения таблицы **«Товар»** разработаем хранимую процедуру, которая создает 350 строк примерно такого вида:

Результаты | Сообщения

	NumArticle	NameArticle	Price	NumType	NumFirm
1	1	Товар 1	189.00	1	1
2	2	Товар 2	42.00	1	20
3	3	Товар 3	2414.00	1	26
4	4	Товар 4	148.00	1	9
5	5	Товар 5	6.00	1	22
6	6	Товар 6	385.00	1	29
7	7	Товар 7	588.00	1	5
8	8	Товар 8	2532.00	1	46
9	9	Товар 9	3.00	1	6
10	10	Товар 10	35.00	1	31
11	11	Товар 11	131.00	1	12
12	12	Товар 12	232.00	1	38

Здесь поле «НомерФирмы» заполнено псевдослучайными равномерно распределенными числами в диапазоне первичного ключа таблицы «Фирма». Поле «Номер типа» заполнено такими значениями: товар 1-товар 50 относится к типу 1, товар 51-100 относится к типу 2 и т.п., . . . товар 301-350 относится к типу 7.

Для заполнения поля «Цена» мы используем такое правило: 50 % лекарств имеет цену от 0 до 200 р. и 50 % - от 201 до 3000 р.

```
DECLARE @price NUMERIC(4),
```

```
        @x FLOAT

-- предварительно инициализируем функцию RAND(), чтобы она
-- генерировала одну и ту же последовательность чисел
SET @x = RAND(1)
. . .
-- получаем случайную величину от 0 до 1
SET @x=RAND()

IF @x<=0.5      -- с вероятностью 0.5 цена от 0 до 200
   SET @price=(RAND()*200.0)
ELSE            -- с вероятностью 0.5 цена от 200 до 3000
   SET @price=(RAND()*2800.0+200.0)
```

Далее заполним таблицу **«Чек».** Для этого разработаем хранимую процедуру, с помощью которой создадим 30 000 строк примерно такого вида:

Результаты | Сообщения

	NumCheck	DateCheck	Discount
1	1	2016-11-01 00:00:00.000	NULL
2	2	2016-04-12 00:00:00.000	NULL
3	3	2016-10-16 00:00:00.000	NULL
4	4	2016-09-05 00:00:00.000	NULL
5	5	2016-01-14 00:00:00.000	NULL
6	6	2016-03-18 00:00:00.000	NULL
7	7	2016-04-03 00:00:00.000	NULL
8	8	2016-10-29 00:00:00.000	NULL
9	9	2016-01-04 00:00:00.000	NULL
10	10	2016-01-19 00:00:00.000	NULL
11	11	2016-08-08 00:00:00.000	NULL
12	12	2016-03-24 00:00:00.000	NULL

Здесь дата генерируется следующим образом. Для того чтобы проиллюстрировать увеличение объема продаж, создадим

- 8 000 чеков за 2016 год,
- 10 000 чеков за 2017 год и
- 12 000 чеков за 2018 год.

Будем генерировать месяц по следующему правилу: 30% продаж приходится на зиму, а остальное – поровну на весну, лето и осень.

Номер дня также выбирается случайно, с учетом количества дней в полученном месяце. Затем из этих частей составляется дата в виде строковой переменной в формате ‘ГГГГ-ММ-ДД’. Время – часы, минуты, секунды – тоже можно при необходимости генерировать с помощью случайных величин.

Поле «Скидка» будет заполнено позже, когда будут данные о продажах.

Наконец, заполним таблицу **«Продажа»**.

Результаты | Сообщения

	Price	Quantity	NumArticle	NumCheck	NumDept
1	2251.00	1	50	1	190
2	1636.00	1	268	1	189
3	1617.00	1	238	1	190
4	904.00	1	79	2	105
5	12.00	1	51	2	108
6	137.00	1	93	2	105
7	2436.00	1	283	3	122
8	28.00	2	286	4	151
9	182.00	1	314	5	200

Для заполнения этой таблицы создадим хранимую процедуру, которая перебирает все строки из таблицы «Чек». (Вспомните о возможности использования курсоров!) Предполагаем, что в одном чеке могут присутствовать продажи из разных отделов, но только из одной аптеки.

В каждом чеке может быть от 1 до 5 наименований лекарств и прочих товаров, каждое в количестве от 1 до 3 штук:

- 1 штука с вероятностью 0.9,
- 2 штуки с вероятностью 0.09,
- 3 штуки с вероятностью 0.01.

Для того чтобы отразить сезонность продаж разных типов лекарств, используем следующие правила:

- **зимой** 50% проданных лекарств относятся к типу **"от гриппа"**, остальные типы равновероятны,
- **весной** 50% проданных лекарств относятся к типу **"витамины"**, остальные типы равновероятны,
- **летом** 50% проданных лекарств относятся к типу **"против диареи"**, остальные типы равновероятны,
- **осенью** 50% проданных лекарств относятся к типу **"от кашля"**, остальные типы равновероятны.

Цена лекарства копируется из таблицы «Товар», позже по сумме чека будем вычислять скидки.

В процессе выполнения процедуры было создано более 89 тыс. строк. Процедура выполнялась более 5 минут! Для такого количества данных это что-то слишком медленно; похоже, следует проанализировать узкие места процедуры.

Узкими местами обычно бывают условия в запросах. Выполнение запросов можно существенно ускорить, если применять индексы. В нашей процедуре используется запрос к таблице **sale** с условиями на столбцы **numCheck** и **numDept**, по которым в данной таблице нет индексов. Создадим их:

```
CREATE INDEX saleCheck ON sale (numCheck)

CREATE INDEX saleDept ON sale (numDept)
```

Снова запустим ту же самую процедуру. Теперь она выполнилась за 46 секунд!

Теперь напишем процедуру для вычисления скидок (это необязательный этап, просто для данной предметной области он позволяет придать нашим искусственным данным больше правдоподобия). В зависимости от общей суммы покупки скидка вычисляется по следующему правилу: от 1000 до 5000 р. – 2%, более 5000 р. – 5%. Обратите внимание на использование конструкции CASE!

```
CREATE PROC Discounts AS

DECLARE cur1 CURSOR FOR SELECT NumCheck FROM Bill
DECLARE @numCheck NUMERIC(6),
        @sumCheck NUMERIC(6),
        @coeff INT

OPEN cur1
FETCH cur1 INTO @numCheck
WHILE @@FETCH_STATUS=0
   BEGIN
     -- считаем сумму чека и процент скидки
     SELECT @sumCheck=SUM(price*quantity), @coeff=
        (CASE
          WHEN @sumCheck>5000 THEN 5
          WHEN @sumCheck>1000 AND @sumCheck<=5000 THEN 2
          ELSE 0
        END)
        FROM sale WHERE @numCheck=NumCheck
     -- записываем скидку в чек
     UPDATE Bill SET Discount=@coeff WHERE @numCheck=NumCheck
     -- уменьшаем цену в продажах на процент скидки
```

```
      UPDATE Sale SET Price=Price*(100.0-@coeff)/100.0
           WHERE @numCheck=NumCheck
      FETCH cur1 INTO @numCheck
      END
DEALLOCATE cur1
```

Таблицы «Чек» и «Продажа» после выполнения процедуры подсчета скидок:

Результаты | Сообщения

	NumCheck	DateCheck	Discount
1	1	2016-11-01 00:00:00.000	5
2	2	2016-04-12 00:00:00.000	2
3	3	2016-10-16 00:00:00.000	2
4	4	2016-09-05 00:00:00.000	0
5	5	2016-01-14 00:00:00.000	0
6	6	2016-03-18 00:00:00.000	2
7	7	2016-04-03 00:00:00.000	2
8	8	2016-10-29 00:00:00.000	5
9	9	2016-01-04 00:00:00.000	0
10	10	2016-01-19 00:00:00.000	2

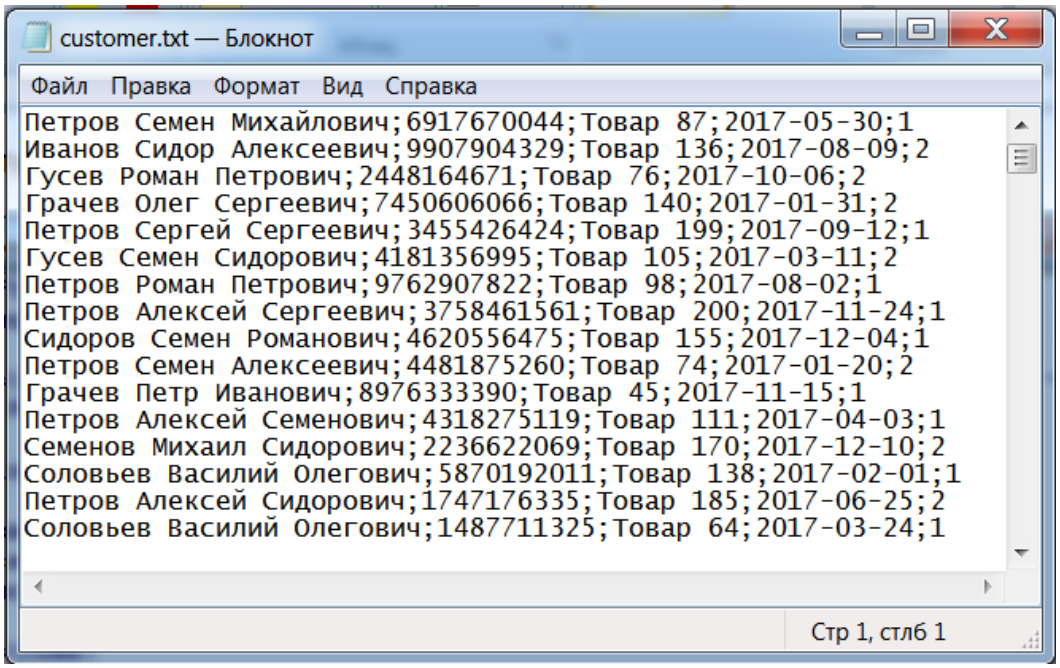
Результаты | Сообщения

	Price	Quantity	NumArticle	NumCheck	NumDept
1	2138.45	1	50	1	190
2	1536.15	1	238	1	190
3	1554.20	1	268	1	189
4	134.26	1	93	2	105
5	11.76	1	51	2	108
6	885.92	1	79	2	105
7	2387.28	1	283	3	122
8	28.00	2	286	4	151
9	182.00	1	314	5	200
10	2864.54	1	229	6	171

Наконец, проиллюстрируем загрузку данных из внешнего текстового файла. Пусть для аналитических целей, кроме рассмотренных выше данных о продажах, нам понадобится информация о лекарствах, которые выдавались пациентам-льготникам бесплатно. Эту информацию мы сгенерируем, например, с помощью программы, написанной на C#.

В созданном файле данные хранятся в формате:

```
Номер; ФИО; паспорт; лекарство; дата; количество
```



Обратите внимание, что в формате даты день и месяц обязательно должен состоять из двух цифр!

Создаем таблицу с такой же структурой. Чтобы проще было копировать данные, порядок полей сделаем в точности таким же, как в файле:

```
CREATE TABLE SocialReceipt
(     fioCust VARCHAR(100) NOT NULL,
      pasportCust VARCHAR(10) NOT NULL,
      nameArticle VARCHAR(20) NOT NULL,
      dateCust DATETIME NOT NULL,
      quantityCust INT NOT NULL)
```

Для загрузки данных из внешнего источника в SQL server существуют разные способы. Мы воспользуемся программой **bcp (bulk copy procedure – процедура массового копирования)**. Эта программа служит как для загрузки данных из текстового файла, так и для выгрузки данных в текстовый файл.

Программа **bcp** запускается из командной строки Windows **(Пуск – Поиск – cmd – Enter)**. В нашем случае формат команды следующий:

```
bcp drugstores.dbo.SocialReceipt in c:\USERS\HOME\customers.txt
-T -S HOME-PC -C 1251 -t; -c
```

Здесь `drugstores.dbo.`**`SocialReceipt`** – полное имя таблицы данных, в которую мы загружаем информацию, в виде: `ИмяБД.Владелец.ИмяТаблицы;`

`in` – направление потока данных: извне в SQL server;

`c:\USERS\HOME\customers.txt` – полное имя файла-источника;

`-T` – флажок означает, что используется тип аутентификации Windows, т.е. операция выполняется от имени пользователя Windows;

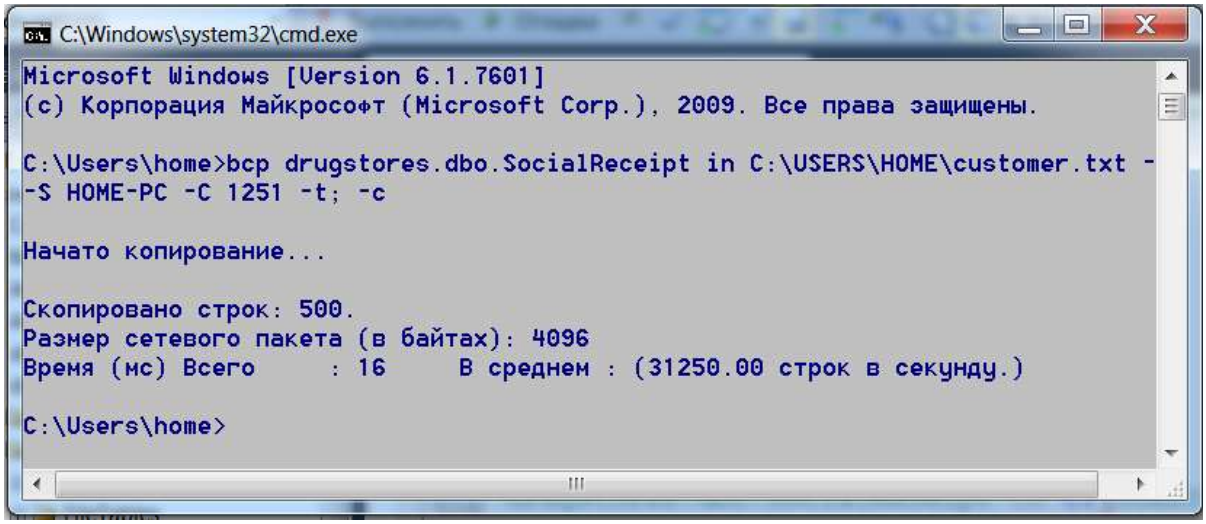
`-S HOME-PC` – имя SQL-сервера;

`-C 1251` – флажок задает кодовую страницу для правильного отображения русских букв (в программе на C# при создании файла нужно задать ту же кодовую страницу, например, через **Encoding.GetEncoding(1251)**);

`-t;` – флажок задает в качестве символа-разделителя полей ";"

`-c` – флажок указывает, что на входе предполагаются данные в символьном (текстовом) представлении. При использовании этого параметра не запрашивается тип данных каждого поля.

Запустим программу на выполнение:



Теперь таблица льготников успешно заполнена:

Результаты | Сообщения

	fioCust	pasportCust	nameArticle	dateCust	quantityCust
1	Петров Семен Михайлович	6917670044	Товар 87	2012-05-30 00:00:00.000	1
2	Иванов Сидор Алексеевич	9907904329	Товар 136	2012-08-09 00:00:00.000	2
3	Гусев Роман Петрович	2448164671	Товар 76	2012-10-06 00:00:00.000	2
4	Грачев Олег Сергеевич	7450606066	Товар 140	2012-01-31 00:00:00.000	2
5	Петров Сергей Сергеевич	3455426424	Товар 199	2012-09-12 00:00:00.000	1
6	Гусев Семен Сидорович	4181356995	Товар 105	2012-03-11 00:00:00.000	2
7	Петров Роман Петрович	9762907822	Товар 98	2012-08-02 00:00:00.000	1
8	Петров Алексей Сергеевич	3758461561	Товар 200	2012-11-24 00:00:00.000	1
9	Сидоров Семен Романович	4620556475	Товар 155	2012-12-04 00:00:00.000	1
10	Петров Семен Алексеевич	4481875260	Товар 74	2012-01-20 00:00:00.000	2
11	Грачев Петр Иванович	8976333390	Товар 45	2012-11-15 00:00:00.000	1
12	Петров Алексей Семенович	4318275119	Товар 111	2012-04-03 00:00:00.000	1

Задание 2. Следует сгенерировать не менее 10 000 записей для таблицы фактов и, по крайней мере, по 10-100 записей для остальных таблиц. Для генерации данных удобно использовать хранимые процедуры SQL.

Обязательно создавайте данные за несколько лет!

При генерации данных постарайтесь задавать какие-нибудь закономерности, например: **«летом мороженого продается в 5 раз больше, чем зимой»** или **«часы пик в продуктовом магазине с 17 до 19»**.

Если таблицы заполнены с помощью хранимых процедур, за это начисляется 5 баллов; если при этом данные генерируются с учетом закономерностей, то еще плюс 5 баллов.

Можно сгенерировать данные с помощью внешней программы (написанной, например, на C#) и записать их в текстовый файл, а затем загрузить в базу. Если данные в каких-либо таблицах загружены из внешних файлов, то ещё плюс 5 баллов.

В самом крайнем случае, можно ограничиться максимум сотней строк для каждой таблицы и все данные занести вручную (1 балл).

Итого 15 баллов.

Этап 3. Запросы, представления, а также хранимые процедуры.

Пример реализации:

Для начала напишем сложный запрос, который содержит детальную информацию о продажах: какие товары, в каком количестве, каких типов, каких фирм, какие отделы каких аптек и когда продали:

```
SELECT NameArticle, s.Price, NameType, NameFirm, NameDept,
     NameDrug, Town, Region, DateCheck, b.NumCheck, Quantity
FROM Bill b, Department dp, Drugstore d, Firm f, Article g,
     Sale s, Type t
WHERE
     b.NumCheck=s.NumCheck AND
     dp.NumDept=s.NumDept AND
     g.NumArticle=s.NumArticle AND
     d.NumDrug=dp.NumDrug AND
     t.NumType=g.NumType AND
     f.NumFirm=g.NumFirm
```

Обратите внимание, что в данном запросе связано 7 таблиц, поэтому потребовалось 6 условий связи.

Результаты | Сообщения

	NameArticle	Price	NameType	NameFirm	NameDept	NameDrug	Town	Region	DateCheck	NumCheck	Quantity
1	Товар 1	185.22	антибиотик	Фирма 1	Отдел 1	Аптека 1	Город 1	Регион 1	2016-02-14 00:00:00.000	248	1
2	Товар 1	185.22	антибиотик	Фирма 1	Отдел 140	Аптека 35	Город 7	Регион 2	2016-04-13 00:00:00.000	567	1
3	Товар 1	179.55	антибиотик	Фирма 1	Отдел 29	Аптека 8	Город 2	Регион 1	2016-11-11 00:00:00.000	791	2
4	Товар 1	185.22	антибиотик	Фирма 1	Отдел 61	Аптека 16	Город 4	Регион 2	2016-12-29 00:00:00.000	946	1
5	Товар 1	189.00	антибиотик	Фирма 1	Отдел 59	Аптека 15	Город 3	Регион 1	2016-01-06 00:00:00.000	998	1
6	Товар 1	179.55	антибиотик	Фирма 1	Отдел 173	Аптека 44	Город 9	Регион 2	2016-07-23 00:00:00.000	1252	1
7	Товар 1	185.22	антибиотик	Фирма 1	Отдел 67	Аптека 17	Город 4	Регион 2	2016-09-23 00:00:00.000	2596	1
8	Товар 1	189.00	антибиотик	Фирма 1	Отдел 28	Аптека 7	Город 2	Регион 1	2016-10-20 00:00:00.000	3252	1
9	Товар 1	189.00	антибиотик	Фирма 1	Отдел 199	Аптека 50	Город 10	Регион 2	2016-10-21 00:00:00.000	3298	2

А ещё лучше запрос сформулировать вот так (запрос имеет более чёткую структуру, никакое условие связи не потеряется):

```
SELECT NameArticle, s.Price, NameType, NameFirm, NameDept,
       NameDrug, Town, Region, DateCheck, b.NumCheck, Quantity
FROM  Sale s JOIN Bill b ON b.NumCheck=s.NumCheck
             JOIN Department dp ON dp.NumDept=s.NumDept
             JOIN Article g ON g.NumArticle=s.NumArticle
             JOIN Drugstore d ON d.NumDrug=dp.NumDrug
              JOIN Firm f ON f.NumFirm=g.NumFirm
             JOIN Type t ON t.NumType=g.NumType
```

Если мы хотим сохранить этот запрос в базе данных для дальнейшего использования, можно создать представление:

```
CREATE VIEW fullSalesView AS
SELECT NameArticle, s.Price, NameType, NameFirm, NameDept,
     NameDrug, Town, Region, DateCheck, b.NumCheck, Quantity
FROM  Sale s   JOIN Bill b ON b.NumCheck=s.NumCheck
               JOIN Department dp ON dp.NumDept=s.NumDept
               JOIN Article g ON g.NumArticle=s.NumArticle
               JOIN Drugstore d ON d.NumDrug=dp.NumDrug
               JOIN Firm f ON f.NumFirm=g.NumFirm
               JOIN Type t ON t.NumType=g.NumType
```

А также можно создать вспомогательную таблицу и переписать данные в неё:

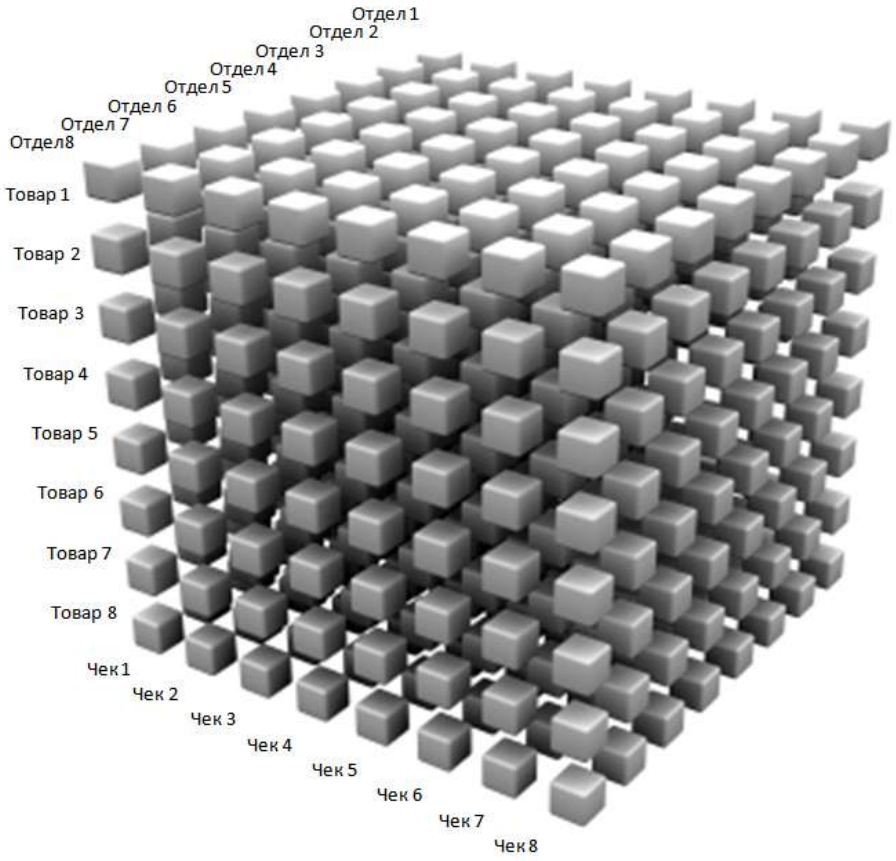
```
CREATE TABLE fullSalesTable
(
     NameArticle VARCHAR(50),
     Price Numeric(6,2),
     NameType VARCHAR(50),
     NameFirm VARCHAR(50),
     NameDept VARCHAR(50),
     NameDrug VARCHAR(50),
     Town VARCHAR(50),
     Region VARCHAR(50),
     DateCheck DATETIME,
     NumCheck INT,
     Quantity INT,
CONSTRAINT pk_sales PRIMARY KEY(NameArticle, NameDept, NumCheck)
)

INSERT INTO fullSalesTable SELECT * FROM fullSalesView
```

Обратите внимание, что данная таблица денормализована. Первичным ключом здесь является комбинация из названия товара, названия отдела и номера чека. Из названия товара (части ключа) следует название фирмы и тип лекарства (неключевые поля), из номера чека следует дата чека, то есть нарушена **вторая** нормальная форма.

Таблица или представление, которые мы только что получили, на самом деле хранят многомерные данные. Многомерные данные удобно представлять в виде так называемых **OLAP-кубов**.

На следующем рисунке представлен OLAP-куб для нашей таблицы. Каждая ячейка такого куба хранит количество и цену некоторого товара, проданного в некотором отделе в составе некоторого чека. «Количество» и «Цена» представляют собой **факты** куба, а «Товар», «Отдел» и «Чек» - **измерения** куба. Заметим, что измерение «Товар» может быть

сгруппировано (или агрегировано) по типу товара или по фирме-изготовителю, измерение «Отдел» может быть сгруппировано по аптеке, городу или региону, а измерение «Чек» может быть агрегировано по дате, по месяцу или по году.



Из этого куба можно получать отдельные **ячейки**, **срезы**, а также агрегировать данные по разным критериям.

Рассмотрим разные запросы с агрегированием данных, т.е., с использованием группировок и итоговых функций.

- Сколько единиц каждого товара и на какую сумму продано в каждом отделе?

```
SELECT NameArticle, NameDept, SUM(Quantity), SUM(Quantity*Price)
     FROM fullSalesView
GROUP BY NameArticle, NameDept
```

Результаты | Сообщения

	NameArticle	NameDept	(Отсутствует имя столбца)	(Отсутствует имя столбца)
1	Товар 140	Отдел 177	3	288.12
2	Товар 155	Отдел 103	1	61.74
3	Товар 99	Отдел 192	1	111.00
4	Товар 51	Отдел 87	1	11.40
5	Товар 102	Отдел 110	1	75.46

В этом примере мы «схлопнули» (просуммировали) измерение «Чек», а по остальным измерениям оставили текущий уровень детализации.

- Сколько единиц каждого **типа** товаров и на какую сумму продано в каждом отделе?

```
SELECT NameType, NameDept,
       SUM(Quantity) AS Количество,
       SUM(Quantity*Price) AS Стоимость
FROM fullSalesView GROUP BY NameType, NameDept
```

Результаты | Сообщения

	NameType	NameDept	Количество	Стоимость
1	витамины	Отдел 167	79	57326.39
2	антибиотик	Отдел 33	46	28714.04
3	жаропонижающее	Отдел 81	48	23823.97
4	обезболивающее	Отдел 136	60	46756.63
5	противодиарейное	Отдел 126	102	102835.58
6	противодиарейное	Отдел 114	86	67612.95
7	противодиарейное	Отдел 140	112	122138.09
8	от кашля	Отдел 165	83	86981.71
9	от кашля	Отдел 88	92	70870.75
10	против гриппа	Отдел 88	88	75644.78

В этом примере мы также «схлопнули» измерение «Чек», а для измерения «Товар» произвели агрегирование по типу товара.

- Сколько единиц каждого **типа** товаров и на какую сумму продано в каждом городе?

```
SELECT NameType, Town,
       SUM(Quantity) AS Количество,
       SUM(Quantity*Price) AS Стоимость
FROM fullSalesView GROUP BY NameType, Town
```

Результаты | Сообщения

	NameType	Town	Количество	Стоимость
1	жаропонижающее	Город 10	834	489298.75
2	противодиарейное	Город 9	1896	1836292.57
3	против гриппа	Город 6	2099	1701688.90
4	жаропонижающее	Город 2	756	444518.31
5	против гриппа	Город 5	2122	1675023.67
6	противодиарейное	Город 4	1782	1800554.96
7	от кашля	Город 3	1937	1848147.65
8	витамины	Город 6	1893	1409332.67
9	от кашля	Город 1	1788	1741673.21

В этом примере мы дополнительно для измерения «Отдел» произвели агрегирование по городу.

- На какую сумму продано лекарств «от гриппа» в каждой аптеке в порядке убывания итоговой суммы?

Результаты | Сообщения

	NameDrug	Стоимость
1	Аптека 44	396344.08
2	Аптека 49	394977.10
3	Аптека 36	383321.60
4	Аптека 9	376965.35
5	Аптека 19	373060.34
6	Аптека 35	369683.80
7	Аптека 20	369055.19
8	Аптека 27	367243.79
9	Аптека 31	364866.05
10	Аптека 29	364405.48

```
SELECT NameDrug, SUM(Quantity*Price)
AS Стоимость
     FROM fullSalesView
     WHERE NameType='против гриппа'
GROUP BY NameDrug
ORDER BY 2 DESC
```

В этом случае мы получили **срез** куба – в измерении «Товар» мы зафиксировали значение для типа товара. По измерению «Отдел» произвели группировку данных по уровню аптек, по измерению «Чек» все значения просуммированы.

- А можно получить суммы продаж «от гриппа» по месяцам в порядке убывания итогов?

Результаты | Сообщения

	Месяц	Стоимость
1	1	4118855.05
2	2	4051490.61
3	12	3951111.94
4	9	617008.10
5	11	565429.19
6	7	543173.36
7	8	533461.52
8	3	523443.13
9	10	508090.13
10	6	495894.49
11	4	473969.44
12	5	448156.73

```
SELECT MONTH(DateCheck) AS Месяц,
       SUM(Quantity*Price) AS Стоимость
FROM fullSalesView
WHERE NameType='против гриппа'
GROUP BY MONTH(DateCheck)
ORDER BY 2 DESC
```

В этом случае мы также получили срез куба – в измерении «Товар» мы зафиксировали значение для типа товара. По измерению «Отдел» все значения просуммированы, а по измерению «Чек» данные агрегированы в масштабе месяцев.

- А если по каждой аптеке отдельно?

```
SELECT NameDrug, MONTH(DateCheck) AS Месяц,
       SUM(Quantity*Price) AS Стоимость
FROM fullSalesView WHERE NameType='против гриппа'
GROUP BY NameDrug, MONTH(DateCheck)
ORDER BY 3 DESC
```

Результаты | Сообщения

	NameDrug	Месяц	Стоимость
1	Аптека 44	2	146858.20
2	Аптека 27	12	133108.02
3	Аптека 31	1	124340.77
4	Аптека 19	12	117002.97
5	Аптека 45	2	111968.51
6	Аптека 9	1	111359.95
7	Аптека 6	1	111149.46
8	Аптека 35	12	109306.94
9	Аптека 24	2	108532.97
10	Аптека 39	12	105253.53
11	Аптека 36	1	104446.16
12	Аптека 43	1	104104.05
13	Аптека 17	2	103762.18

Это тоже срез куба – в измерении «Товар» зафиксировано значение для типа товара. По измерению «Отдел» данные агрегированы в масштабе аптек, а по измерению «Чек» данные агрегированы в масштабе месяцев.

- А сколько реализовано товара с названием «Товар 1» в «Аптеке 35» за апрель 2016-го года?

В этом запросе зафиксированы значения по измереням «Товар» и «Аптека». Для измерения «Чек» данные агрегированы по месяцам, и зафиксированы значения месяца и года.

Для таких запросов самым удобным подходом будет создание хранимой процедуры, и чем больше у неё будет параметров, тем лучше:

```
CREATE PROC detailSales
@nameDrug VARCHAR(50),
@nameArticle VARCHAR(50),
@month INT,
@year INT
AS
SELECT SUM(Quantity*Price) FROM fullSalesView
WHERE nameDrug=@nameDrug AND NameArticle=@NameArticle AND
YEAR(DateCheck)=@year AND MONTH(DateCheck)=@month
GO

EXEC detailSales
'Аптека 35', 'Товар 1', 4, 2016
```

Результаты | Сообщения

	(Отсутствует имя столб...
1	185.22

Можно сформулировать огромное количество подобных запросов и процедур. Заранее очень сложно предугадать, какие именно данные и в каком разрезе понадобятся пользователям. Было бы гораздо удобнее предоставить самим пользователям возможность в визуальном режиме, без написания сложных SQL-команд выбирать данные по нужным критериям. Именно для этого и предназначены специализированные приложения для работы с хранилищами данных. В таких приложениях, кстати, очень легко выполнить следующий запрос (а написать его на обычном SQL довольно сложно):

- А можно получить такую таблицу, чтобы столбцами были, например, месяцы, а в строках находились типы товаров и итоги продаж по месяцам?

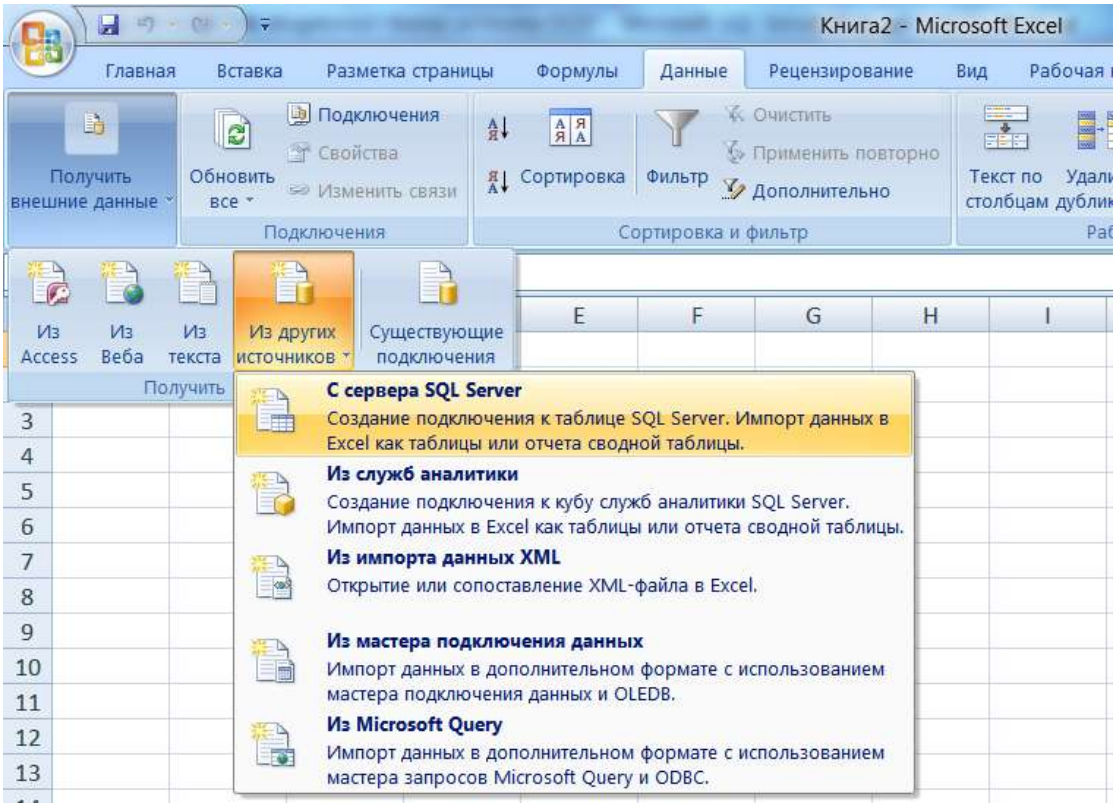
Результаты | Сообщения

	Тип Лекарства	январь	февраль	март	апрель	май	июнь
1	антибиотик	803863.10	692765.21	552923.41	531550.03	628280.61	595443.34
2	витамины	660876.93	590185.59	2843772.52	2785739.31	2658140.87	516710.80
3	жаропонижающее	485038.20	491324.73	334351.04	325994.22	409741.59	325679.00
4	обезболивающее	696770.77	604141.64	504498.25	486420.85	536933.85	481030.28
5	от кашля	844923.48	757341.28	618424.53	635216.26	546624.87	631314.14
6	против гриппа	4118855.05	4051490.61	523443.13	473969.44	448156.73	495894.49
7	противодиарейное	791780.54	893128.99	634901.10	606591.92	625088.84	4040356.21

Такие таблицы называются перекрёстными, или сводными, или кросс-таблицами. Получить такую таблицу стандартными средствами языка SQL можно только для заранее заданных количества и названий столбцов. Для этого используется команда **SELECT** с ключевым словом **PIVOT** (развернуть). Попробуйте разработать такой запрос самостоятельно.

Удобный интерфейс для работы со сводными таблицами представляет офисный пакет MS EXCEL. Попробуем с его помощью создать сводную таблицу, подобную вышеприведенной, а заодно и построить на ее основе диаграмму.

Запустите пакет EXCEL и создайте новую книгу. Перейдите на вкладку «Данные». Нажмите на кнопку «Получить внешние данные». Выберите вариант «Из других источников» - «С сервера SQL server».

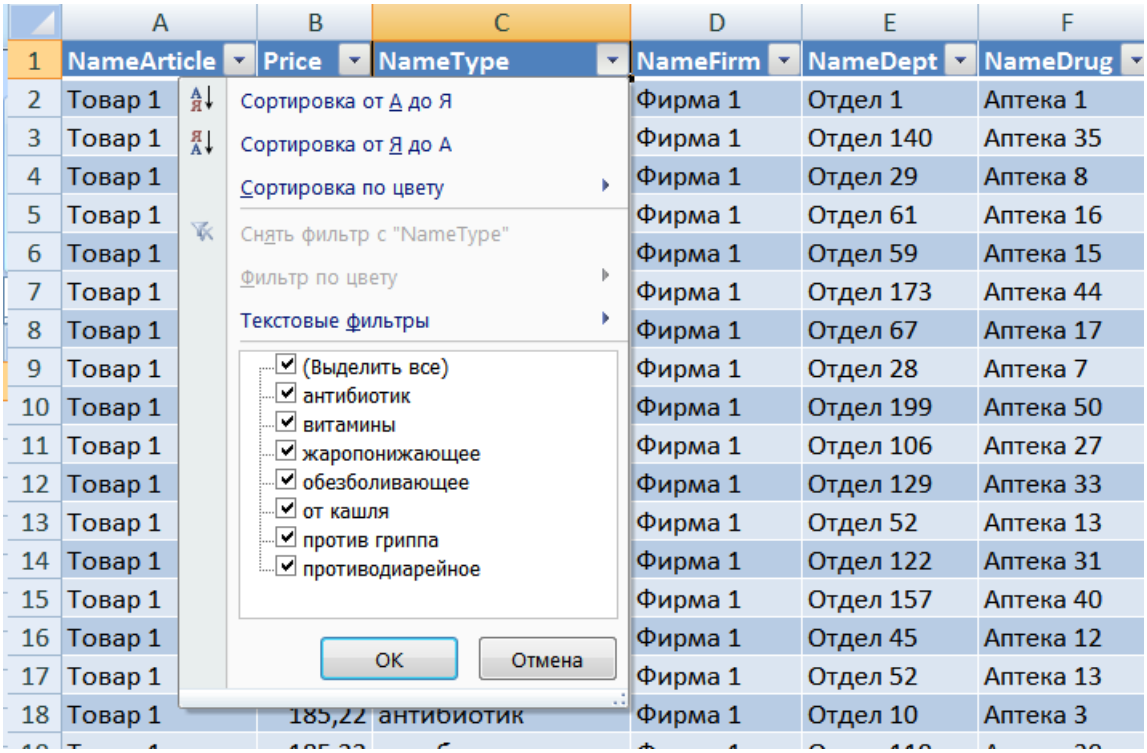


Будет запущен мастер подключений, в котором нужно:

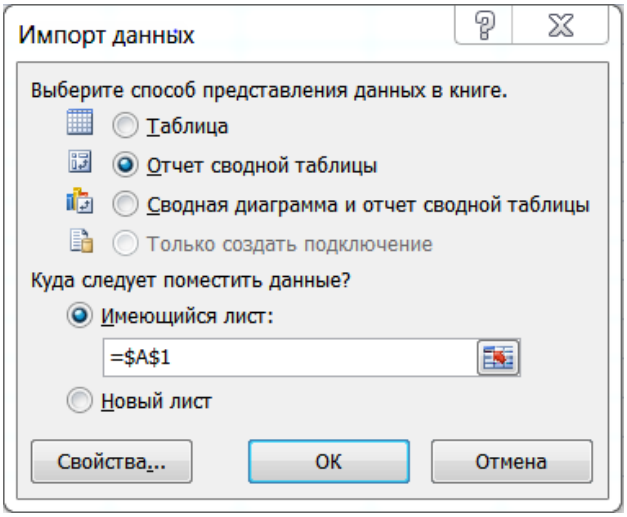
- задать имя сервера,
- выбрать базу данных,
- выбрать таблицу или представление.

Полученное подключение сохраняется в файле с типом ODC для дальнейшего применения.

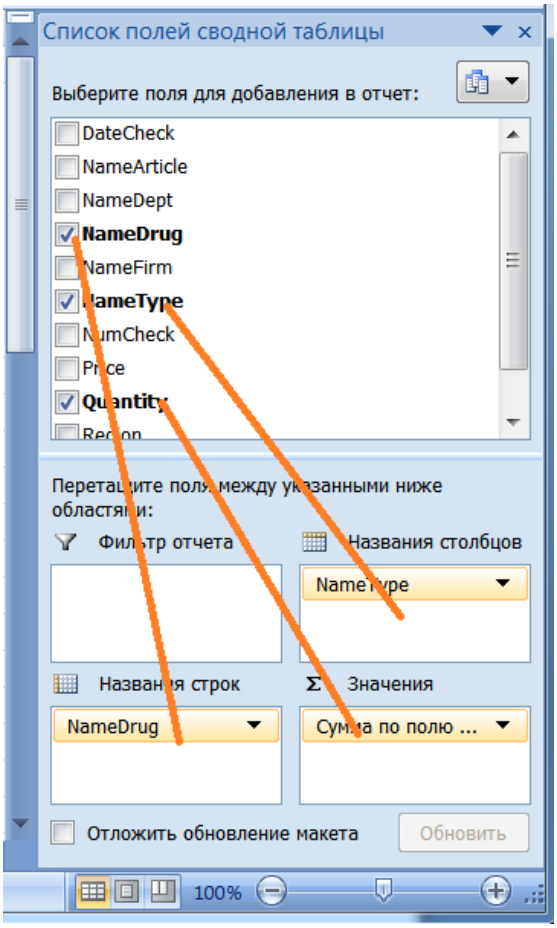
Далее будет предложено выбрать режим представления данных. Если выбрать режим «Таблица», то будут просто импортированы данные из источника без всяких преобразований. Полученную таблицу можно сортировать и фильтровать:



Нас интересует способ представления данных «Отчет сводной таблицы»:



При выборе этого режима справа в окне появляется конструктор сводной таблицы, в котором можно (перетаскиванием) выбирать измерения для столбцов, для строк и факты, а также назначить агрегирующую функцию для фактов (по умолчанию – сумма):



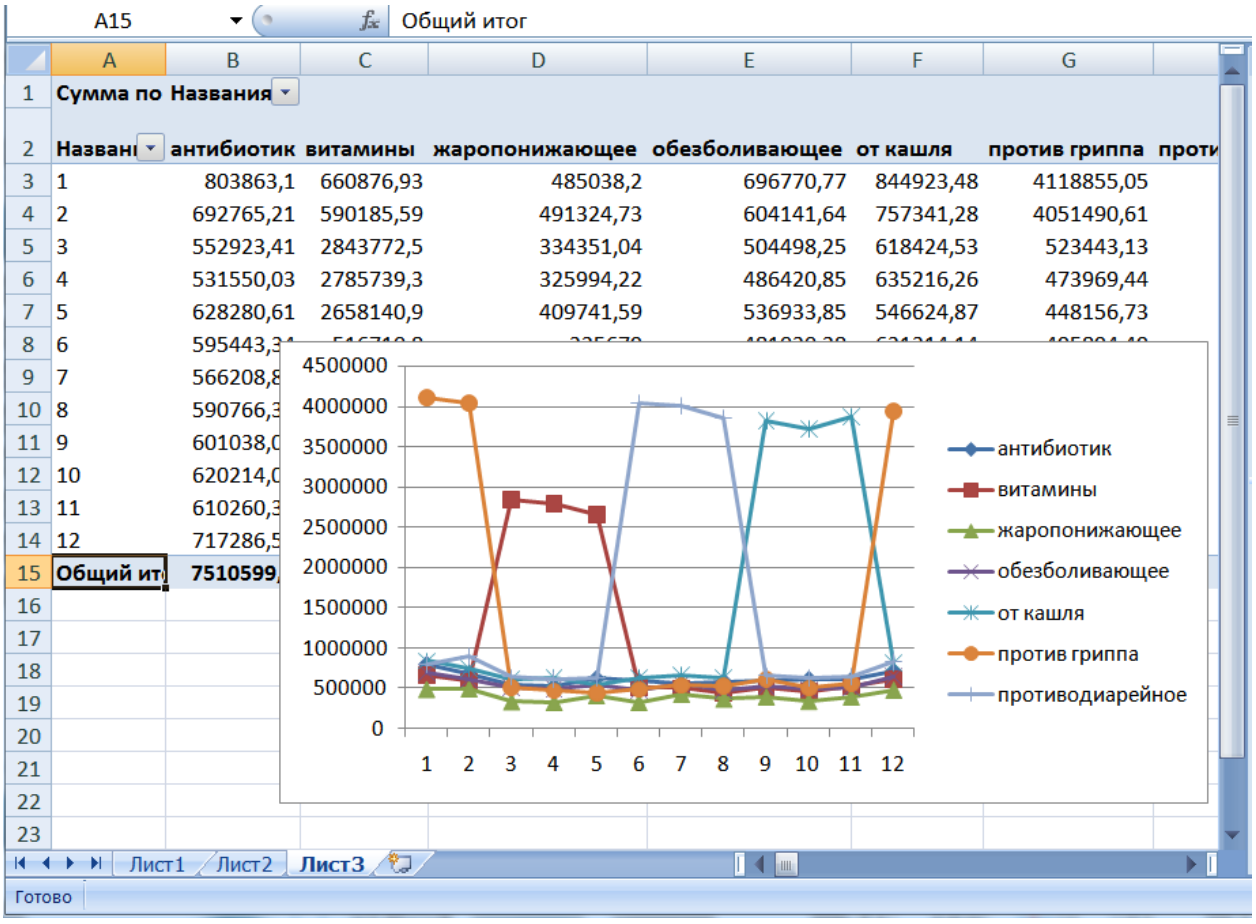
Сводная таблица будет показана в левой части окна:

A2 fx Названия строк

	A	B	C	D	E	F	G	H	I	J
1	Сумма по полю Названия столбцов									
2	Названия строк	антибиотик	витамины	жаропонижающее	обезболивающее	от кашля	против гриппа	противодиарейное	Общий итог	
3	Аптека 1	178	366	181	195	366	447	391	2124	
4	Аптека 10	138	387	133	174	362	400	337	1931	
5	Аптека 11	163	365	171	191	352	407	395	2044	
6	Аптека 12	169	389	186	172	399	436	344	2095	
7	Аптека 13	166	368	190	180	391	366	405	2066	
8	Аптека 14	160	379	162	148	401	383	374	2007	
9	Аптека 15	144	364	169	163	394	357	344	1935	
10	Аптека 16	163	379	174	158	361	374	346	1955	
11	Аптека 17	190	323	148	146	348	427	329	1911	
12	Аптека 18	150	322	150	160	346	385	339	1852	
13	Аптека 19	143	378	189	173	385	403	346	2017	
14	Аптека 2	171	325	140	154	324	393	327	1834	
15	Аптека 20	170	336	173	187	365	391	422	2044	

Лист1 Лист2 Лист3

Если заранее в SQL server создать представление и дополнительно вычислить из даты чека номер месяца, а из цены и количества общую стоимость, то в EXCEL очень легко создать сводную таблицу такого вида:



А с помощью режима «Сводная диаграмма» можно добавить к нашему отчету дополнительную визуализацию.

Задание 3. Напишите запрос, соединяющий все таблицы вашего хранилища – с созданием представления и с записью полученных данных в новую таблицу (2 балла).

Напишите не менее 5 разнообразных запросов для получения срезов куба, с применением группировок и агрегирующих функций (4 баллов).

Напишите хранимую процедуру для получения отдельной ячейки куба (2 балла).

Создайте кросс-таблицу с помощью команды SELECT … PIVOT (3 балла), а также сводную таблицу в MS EXCEL (4 балла). Заодно создайте на основе сводной таблицы диаграмму в EXCEL.

Итого 15 баллов.

Этап 4. Очистка данных

Процесс ETL – extract, transform, load (извлечь, преобразовать, загрузить) – представляет собой промежуточное звено между OLTP-системами и хранилищами данных. Очистка и преобразование данных – это очень важные процедуры перед загрузкой информации в хранилище данных. Здесь возникает вопрос: с какими данными производить эти операции – с данными из исходных таблиц или же с промежуточными результатами? Для начала займемся поиском некорректных и неполных данных и их очисткой в общей таблице, которую создали на предыдущем этапе. Например, с точки зрения полноты данных имеет смысл проверить важные поля на пустоту:

```
SELECT * FROM fullSalesTable WHERE nameArticle IS NULL
SELECT * FROM fullSalesTable WHERE nameType IS NULL
SELECT * FROM fullSalesTable WHERE nameFirm IS NULL
```

И т.п. Если такие поля найдены, то следует либо их заполнить реальными данными, либо просто избавиться от таких строк, поскольку они имеют мало смысла для дальнейшей обработки в хранилище. В приведенных выше примерах отсутствие названия лекарства (nameArticle) означает явную ошибку ввода, поэтому такие строки можно удалить. (Напишите соответствующую команду!) Если название лекарства есть, но отсутствует тип лекарства (nameType) или название фирмы (nameFirm), их можно попробовать найти в соответствующих таблицах и откорректировать пустую ячейку. (Напишите для этой операции запрос или хранимую процедуру!) *(На самом деле, в нашем примере эти 3 поля заведомо не могут быть пустыми, поскольку данные генерировались автоматически.)*

Далее, например, мы хотим в дальнейшем использовать только данные за 2016 - 2018 годы. Всё остальное из таблицы следует удалить.

```
SELECT * FROM fullSalesTable WHERE YEAR(dateCheck) NOT IN
(2016,2017,2018)
```

Результаты | Сообщения

	NameArticle	Price	NameType	NameFirm	NameDept	NameDrug	Town	Region	DateCheck	NumCheck	Quantity
1	Товар 1	189.00	антибиотик	Фирма 1	Отдел 4	Аптека 1	Город 1	Регион 1	2015-01-30 00:00:00.000	300002	2
2	Товар 10	34.30	антибиотик	Фирма 31	Отдел 139	Аптека 35	Город 7	Регион 2	2015-01-31 00:00:00.000	300001	1

```
DELETE FROM fullSalesTable WHERE YEAR(dateCheck) NOT IN
(2016,2017,2018)
```

Также полезной в нашем случае будет следующая проверка: посчитаем, сколько было продаж в таблице **Sale** и сколько строк попало в таблицу `fullSalesTable`.

```
SELECT COUNT(*) FROM fullSalesTable
SELECT COUNT(*) FROM Sale
```

Если не все строки из таблицы **Sales** попали в таблицу **fullSalesTable**, причина может быть в следующем. Таблица **fullSalesTable** собиралась из 7 таблиц и, возможно, где-то было пропущено значение, которое участвовало в условии связи (мы использовали INNER JOIN –внутреннее соединение). Пусть, например, у нас есть товар, который не привязан к типу лекарства:

```
SELECT * FROM Article WHERE numType IS NULL
```

Результаты | Сообщения

	NumArti...	NameArticle	Package	Price	NumTy...	NumFirm
1	351	Товар без типа	Коробка	100.00	NULL	40

и этот товар участвовал в продажах:

```
SELECT * FROM Sale WHERE numArticle IN
   (SELECT numArticle FROM Article WHERE numType IS NULL)
```

Результаты | Сообщения

	Price	Quantity	NumArticle	NumCheck	NumDept
1	200.00	2	351	30000	1

Для такого товара следует выяснить, к какому типу лекарств он относится, заполнить поле **numType**, а затем добавить информацию о продажах этого товара в таблицу **fullSalesTable**. Здесь можно воспользоваться такой уловкой: после исправления типа лекарства информация о продажах этого лекарства сразу же попадет в представление **fullSalesView**, из которого мы эти продажи и скопируем:

```
INSERT INTO fullSalesTable
        SELECT * FROM fullSalesView WHERE nameArticle NOT IN
             (SELECT nameArticle FROM fullSalesTable)
```

Проверки корректности данных можно также провести и для таблицы льготников, которую мы загрузили из внешнего файла. Например, проверим корректность названий лекарств (т. е., соответствие названий нашему справочнику):

```
SELECT * FROM SocialReceipt WHERE nameArticle NOT IN
```

```
        (SELECT nameArticle FROM Article)
```

Результаты | Сообщения

	fioCust	pasportCust	nameArticle	dateCust	quantityCust
1	Петров Семен Михайлович	6917670044	Товар 0	2017-05-30 00:00:00.000	1
2	Петров Олег Сидорович	2695962910	Товар 0	2017-06-19 00:00:00.000	2

Получили строку с названием, которое **отсутствует** в нашем справочнике. Что делать с такими данными? Либо выяснить, какое название имелось в виду, либо просто удалить эту строку.

Преобразование данных не ограничивается только исправлением ошибок. Иногда приходится, например, объединять в рамках одной таблицы данные из нескольких источников. Например, сделаем таблицу, содержащую всю информацию о реализации лекарств – и о продажах, и о льготной реализации.

Создадим общую таблицу со структурой, как у **fullSalesTable**, но с простым первичным ключом:

```
CREATE TABLE fullRealization
(
     NameArticle VARCHAR(50),
     Price Numeric(6,2),
     NameType VARCHAR(50),
     NameFirm VARCHAR(50),
     NameDept VARCHAR(50),
     NameDrug VARCHAR(50),
     Town VARCHAR(50),
     Region VARCHAR(50),
     DateCheck DATETIME,
     NumCheck INT,
     Quantity INT,
     ID INT PRIMARY KEY IDENTITY
)
```

Добавим в нее продажи:

```
INSERT INTO fullRealization SELECT * FROM fullSalesTable
```

А теперь нужно добавить информацию о льготной реализации из таблицы **SocialReceipt**. Обратите внимание, как заполняются недостающие поля.

```
INSERT INTO fullRealization
  SELECT sr.NameArticle, 0 AS Price, t.NameType, f.NameFirm,
    '' AS NameDept,  '' AS NameDrug,  '' AS Town,  '' AS Region,
    sr.DateCust, 0 AS NumCheck, sr.quantityCust
   FROM  SocialReceipt sr
        JOIN Article g ON sr.NameArticle=g.NameArticle
        JOIN Firm f ON f.numFirm=g.numFirm
        JOIN Type t ON t.numType=g.numType
```

Результаты | Сообщения

	NameArti...	Price	NameType	NameFirm	NameD...	NameDr...	Town	Region	DateCheck
89878	Товар 99	108.78	витамины	Фирма 14	Отдел 98	Аптека ...	Город 5	Регион 2	2016-12-11 00:00
89879	Товар 99	111.00	витамины	Фирма 14	Отдел 98	Аптека ...	Город 5	Регион 2	2017-12-13 00:00
89880	Товар 99	108.78	витамины	Фирма 14	Отдел 98	Аптека ...	Город 5	Регион 2	2018-03-30 00:00
89881	Товар 99	111.00	витамины	Фирма 14	Отдел 99	Аптека ...	Город 5	Регион 2	2017-05-10 00:00
89882	Товар бе...	200.00	антибиотик	Фирма 40	Отдел 1	Аптека 1	Город 1	Регион 1	2018-06-20 00:00
89883	Товар 136	0.00	жаропонижающее	Фирма 41					2017-08-09 00:00
89884	Товар 76	0.00	витамины	Фирма 15					2017-10-06 00:00
89885	Товар 140	0.00	жаропонижающее	Фирма 40					2017-01-31 00:00
89886	Товар 199	0.00	обезболивающее	Фирма 13					2017-09-12 00:00
89887	Товар 105	0.00	жаропонижающее	Фирма 29					2017-03-11 00:00
89888	Товар 98	0.00	витамины	Фирма 30					2017-08-02 00:00
89889	Товар 200	0.00	обезболивающее	Фирма 46					2017-11-24 00:00
89890	Товар 155	0.00	обезболивающее	Фирма 26					2017-12-04 00:00

Задание 4. Произведите очистку данных: напишите запросы, которые находят пустые значения, неверные или неполные данные и, при необходимости, корректируют их. *Не ограничивайтесь примерами, приведенными в данном параграфе, придумывайте свои собственные преобразования!*

Итого 5 баллов.

Этап 5. Обогащение данных

Один из интересных видов трансформации данных – это обогащение данных. Например, к обогащению данных можно отнести вычисление разнообразных рейтингов – рейтингов товаров, рейтингов сотрудников, рейтингов поставщиков и т.п. Рассмотрим несколько примеров.

1) Сначала, например, создадим рейтинг товаров за 2016 г. по объемам продаж в стоимостном выражении. Для этого в таблицу товары добавим новый столбец числового типа:

```
ALTER TABLE ARTICLE ADD rating2016 INT;
```

Затем создадим процедуру, в которой используется курсор со следующими данными (из таблиц Товары и Чеки):

- Номер товара,
- Сумма продаж товара за 2016 г.

Данные в курсоре нужно отсортировать по второму столбцу в порядке убывания. Затем в цикле перебираем строки курсора и обновляем значение рейтинга в таблице Товары. Получим примерно такой результат:

```
SELECT * FROM Article ORDER BY rating2016;
```

Результаты | Сообщения

	NumArticle	NameArticle	Price	NumType	NumFirm	rating2016
1	317	Товар 317	2894.00	7	6	1
2	328	Товар 328	2961.00	7	36	2
3	330	Товар 330	2377.00	7	6	3
4	59	Товар 59	2622.00	2	48	4
5	300	Товар 300	2815.00	6	47	5
6	204	Товар 204	2869.00	5	8	6
7	217	Товар 217	2915.00	5	18	7
8	303	Товар 303	2626.00	7	32	8
9	220	Товар 220	2787.00	5	46	9

2) Рассмотрим более сложный пример. Будем создавать рейтинги аптек и городов. Рейтинг аптек будем строить так.

Все аптеки разобьем на 4 группы (A,B,C,D) в зависимости от объемов продаж за весь период.

Интервал *[МинимальныйОбъемПродаж, МаксимальныйОбъемПродаж]* делится на 4 равные части. Если объем продаж аптеки

принадлежит первому отрезку, то аптека имеет рейтинг D, если второму – C, третьему – B и четвертому – A. Алгоритм удобно оформить в виде хранимой процедуры с курсором. Получим такие результаты:

```
SELECT * FROM Drugstore ORDER BY rating;
```

Результаты | Сообщения

	NumDrug	NameDrug	Town	Region	Phone	rating
1	7	Аптека 7	Город 2	Регион 1	787675	A
2	11	Аптека 11	Город 3	Регион 1	767481	A
3	20	Аптека 20	Город 4	Регион 2	224626	A
4	21	Аптека 21	Город 5	Регион 2	181980	A
5	1	Аптека 1	Город 1	Регион 1	525017	A
6	31	Аптека 31	Город 7	Регион 2	460914	A
7	33	Аптека 33	Город 7	Регион 2	306788	A
8	39	Аптека 39	Город 8	Регион 2	936817	A
9	35	Аптека 35	Город 7	Регион 2	110367	A
10	36	Аптека 36	Город 8	Регион 2	778449	B
11	37	Аптека 37	Город 8	Регион 2	159746	B
12	48	Аптека 48	Город...	Регион 2	873256	B
13	49	Аптека 49	Город...	Регион 2	903271	B
14	50	Аптека 50	Город...	Регион 2	465794	B
15	19	Аптека 19	Город 4	Регион 2	840584	B

```
SELECT * FROM Drugstore
```

Результаты | Сообщения

	NumDrug	NameDrug	Town	Region	Phone	rating
1	1	Аптека 1	Город 1	Регион 1	525017	A
2	2	Аптека 2	Город 1	Регион 1	949597	D
3	3	Аптека 3	Город 1	Регион 1	107967	B
4	4	Аптека 4	Город 1	Регион 1	409099	B
5	5	Аптека 5	Город 1	Регион 1	289814	B
6	6	Аптека 6	Город 2	Регион 1	450039	B
7	7	Аптека 7	Город 2	Регион 1	787675	A
8	8	Аптека 8	Город 2	Регион 1	811716	B
9	9	Аптека 9	Город 2	Регион 1	553691	B
10	10	Аптека 10	Город 2	Регион 1	534565	B
11	11	Аптека 11	Город 3	Регион 1	767481	A
12	12	Аптека 12	Город 3	Регион 1	247438	B
13	13	Аптека 13	Город 3	Регион 1	292790	B
14	14	Аптека 14	Город 3	Регион 1	128906	B
15	15	Аптека 15	Город 3	Регион 1	489673	C

Посмотрим, сколько в каждом городе аптек каждой категории. Для удобства создадим вспомогательное представление:

```
CREATE VIEW townRating as
```

```
SELECT Town, COUNT(NumDrug) as Count, rating FROM
Drugstore
GROUP BY Town, rating;

SELECT * FROM townRating;
```

Результаты | Сообщения

	Town	Count	rating
1	Город 1	1	A
2	Город 2	1	A
3	Город 3	1	A
4	Город 4	1	A
5	Город 5	1	A
6	Город 7	3	A
7	Город 8	1	A
8	Город 1	3	B
9	Город 10	3	B
10	Город 2	4	B
11	Город 3	3	B
12	Город 4	1	B
13	Город 5	1	B
14	Город 6	3	B
15	Город 8	3	B

Можно также посмотреть результат в виде кросс-таблицы, это будет более компактно:

Результаты | Сообщения

	Город	A	B	C	D
1	Город 1	1	3	0	1
2	Город 10	0	3	2	0
3	Город 2	1	4	0	0
4	Город 3	1	3	1	0
5	Город 4	1	1	2	1
6	Город 5	1	1	2	1
7	Город 6	0	3	2	0
8	Город 7	3	0	0	2
9	Город 8	1	3	1	0
10	Город 9	0	4	1	0

Теперь подсчитаем рейтинг городов. Считаем, что категория аптеки

A – это 4 балла,
B – это 3 балла,
C – это 2 балла,
D – это 1 балл.

```
SELECT Town, SUM(Count * IIF(rating='A', 4,
                             IIF(rating='B', 3,
                                IIF(rating='C', 2, 1))))
         as townRating
FROM townRating
GROUP BY Town
```

Результаты | Сообщения

	Town	townRating
1	Город 1	14
2	Город 10	13
3	Город 2	16
4	Город 3	15
5	Город 4	12
6	Город 5	12
7	Город 6	13
8	Город 7	14
9	Город 8	15
10	Город 9	14

Здесь используется функция IIF, которая представляет собой «свернутый» оператор условия:

```
IIF( условие, результат_если_истина, результат_если_ложь)
```

Если первый параметр (логическое выражение) истинный, то функция возвращает значение второго параметра. В противном случае возвращается значение третьего параметра.

Задание 5. Проведите обогащение данных для своего хранилища. Постарайтесь не ограничиваться примерами из данного параграфа, придумывайте свои примеры обогащения данных.

Итого 5 баллов.

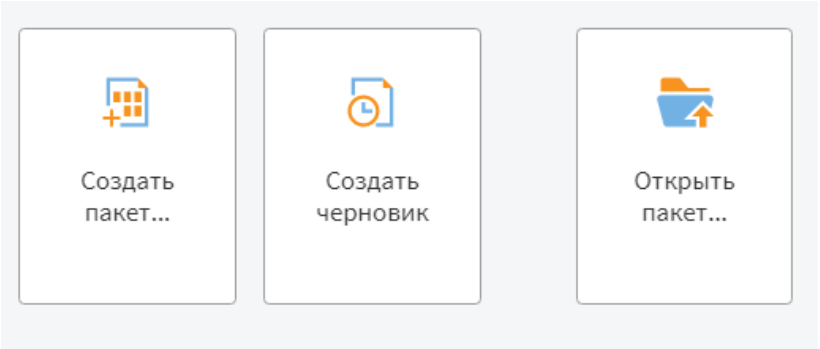
Работа в программе Loginom

Loginom является аналитической платформой, которая может подключаться к различным источникам данных**,** в том числе **к SQL Server. Loginom** поставляется в нескольких вариантах. Для целей обучения имеется бесплатная версия **Loginom Community,** которую можно использовать без каких-либо ограничений.

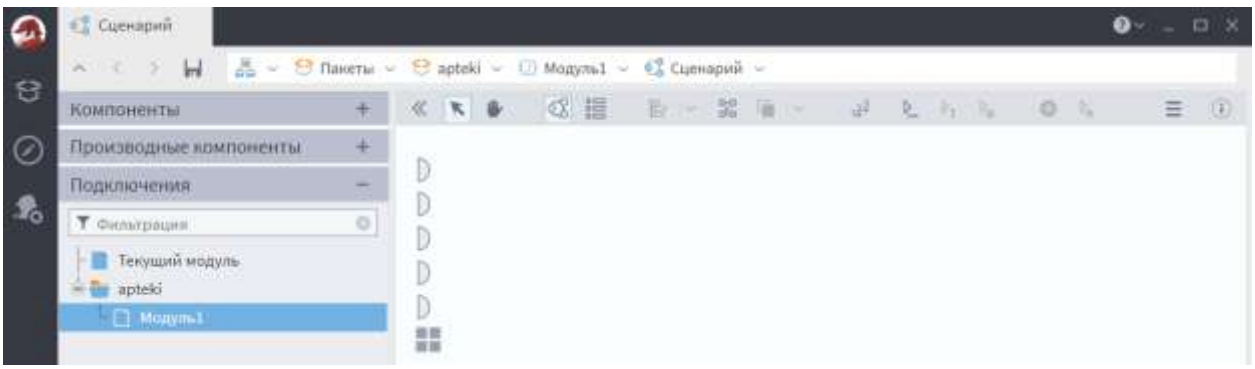
Перед тем как начинать работу в **Loginom**, рекомендуется ознакомиться с руководством пользователя по адресу https://help.loginom.ru/userguide/

Этап 6. Подключение к SQL server, обработка данных и визуализация

Установите и запустите **Loginom Community.** Работу начнём с создания нового пакета (это то же самое, что проект во многих других программах):

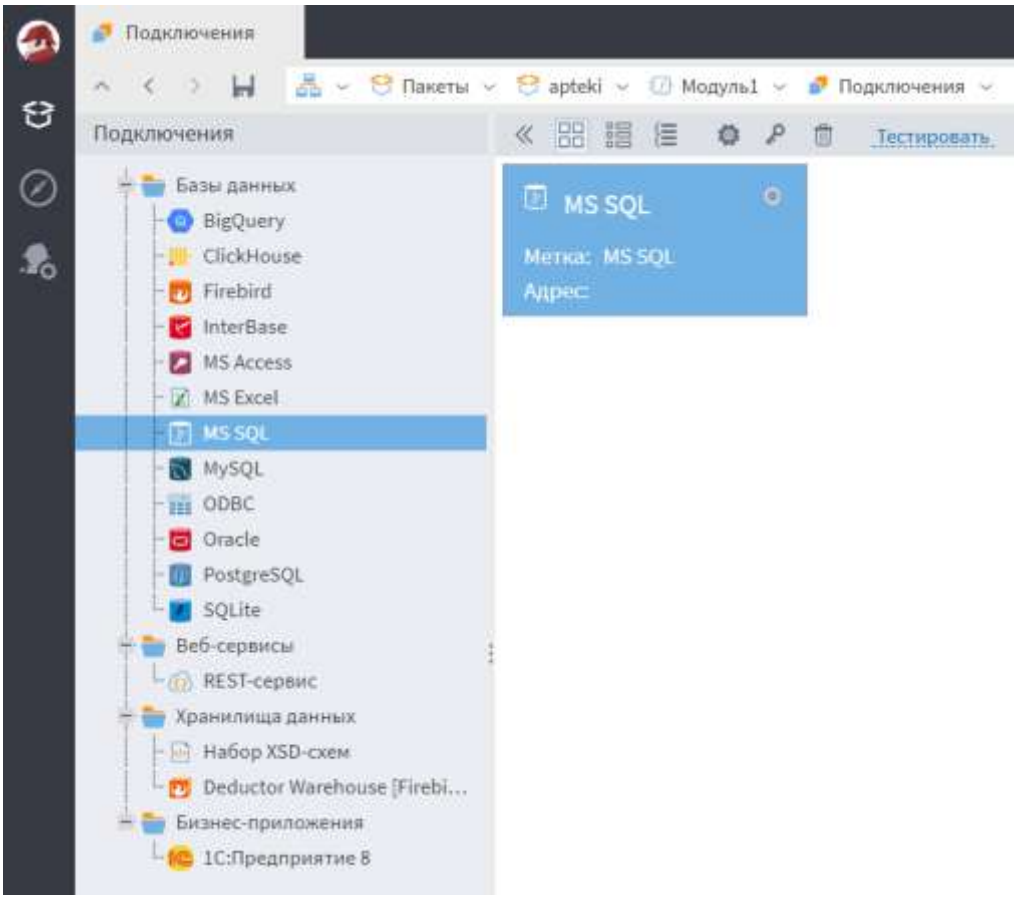


В открывшемся окне в левой панели перейдем на вкладку **Подключения** (она находится в самом низу)**.** Мы назвали пакет **apteki,** по умолчанию создан один модуль с названием **Модуль1** (при необходимости его можно переименовать).

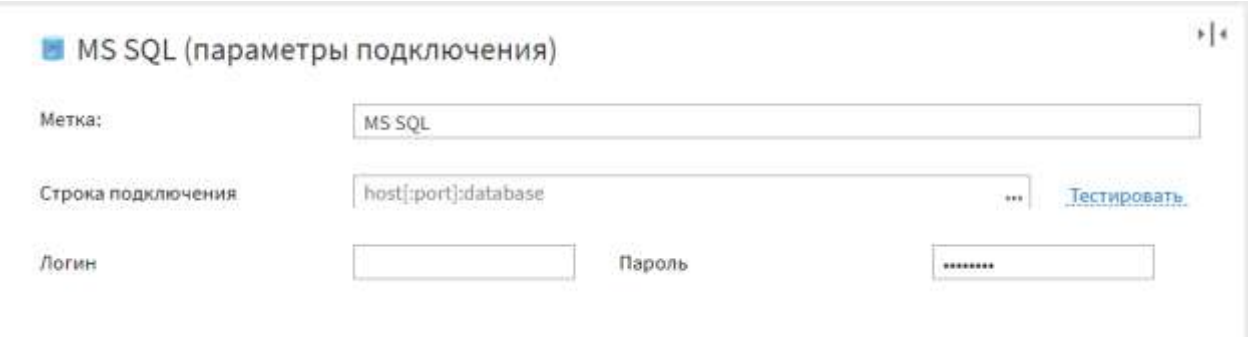


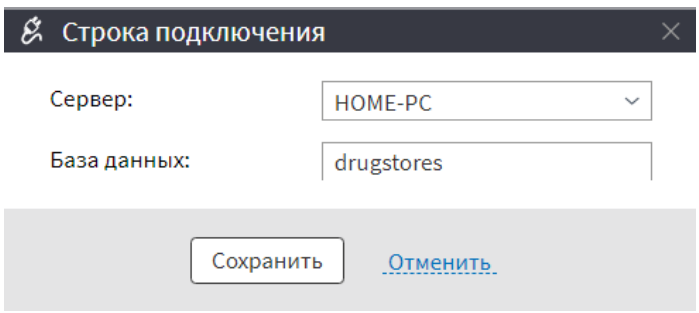
Нужно щелкнуть правой кнопкой мыши по Модулю1 и перейти к подключениям.

В левой панели есть много разных вариантов для подключения. Выберем MS SQL. Щелкнем правой кнопкой и добавим подключение. Новое подключение появится в рабочей области.



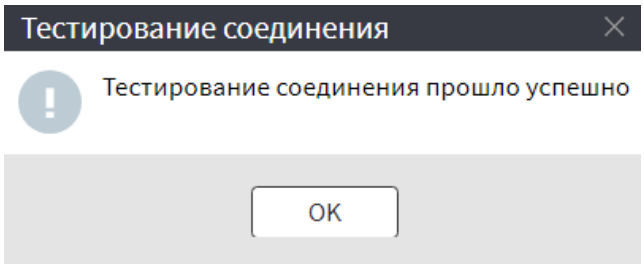
Щелкнем по нему два раза мышкой и в появившемся окне будем настраивать параметры:





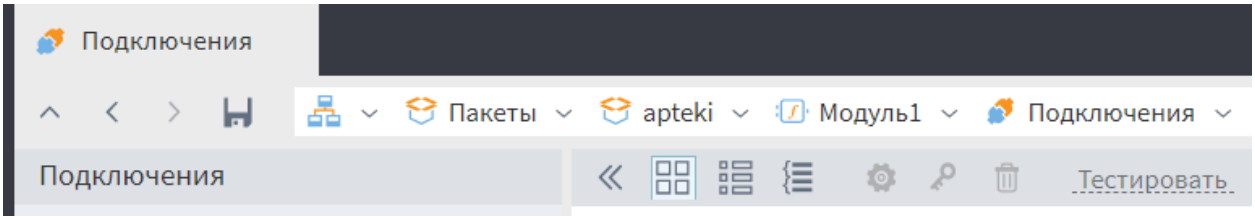
В строке подключения нужно задать имя сервера и имя базы данных.

Можно сразу протестировать подключение.

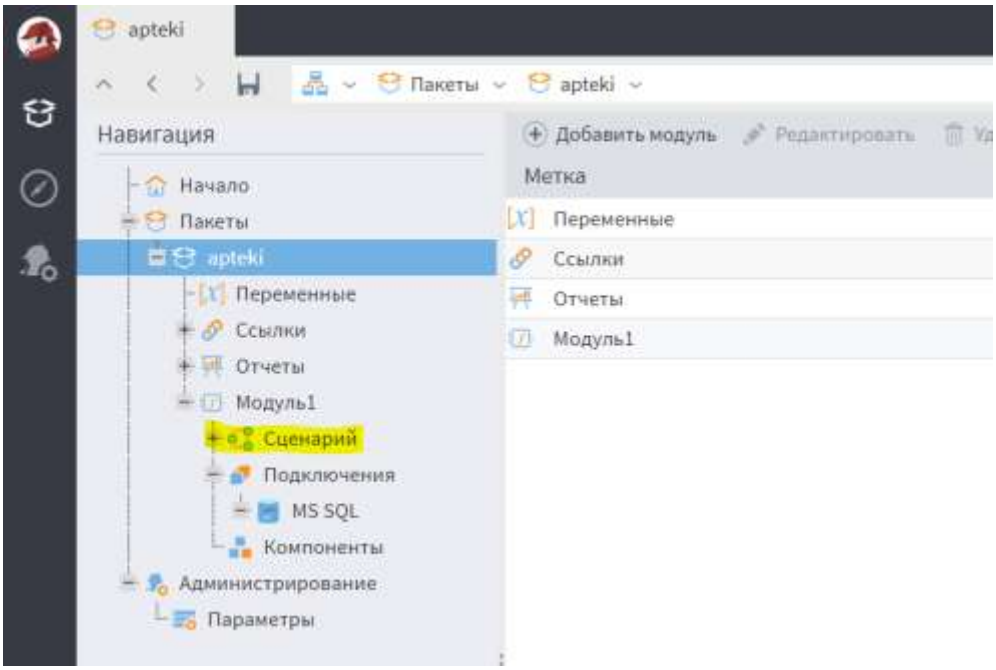


Затем нужно нажать на кнопку  . Соединение активно и готово к работе.

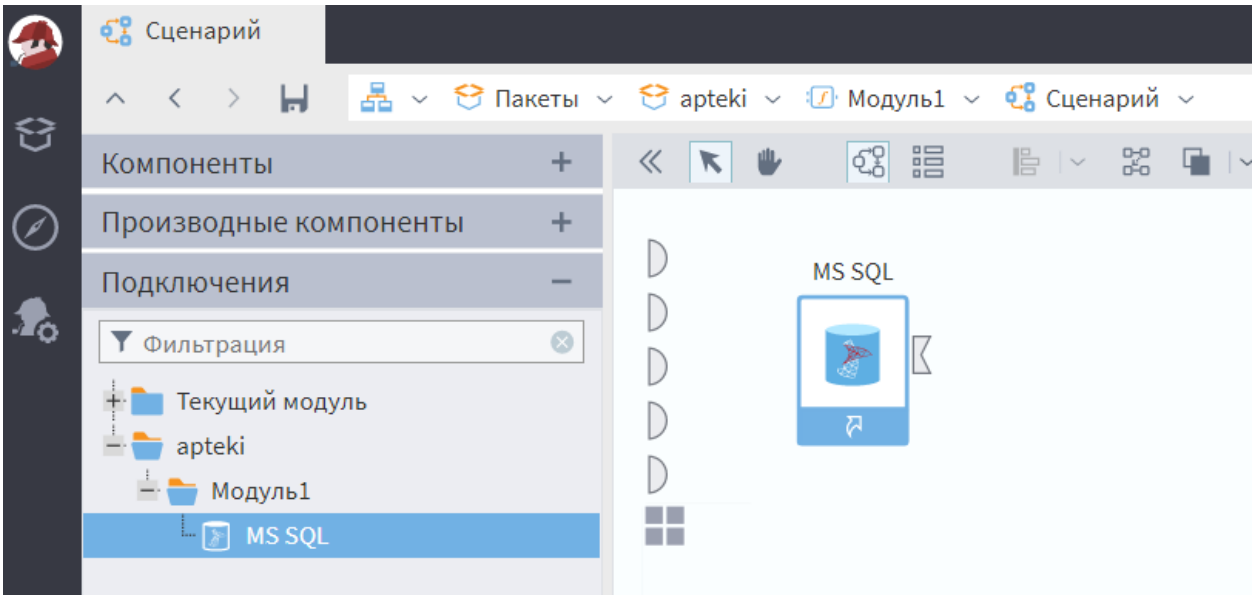
Верхнее меню показывает, в каком режиме вы сейчас находитесь и последовательный путь к этому режиму.



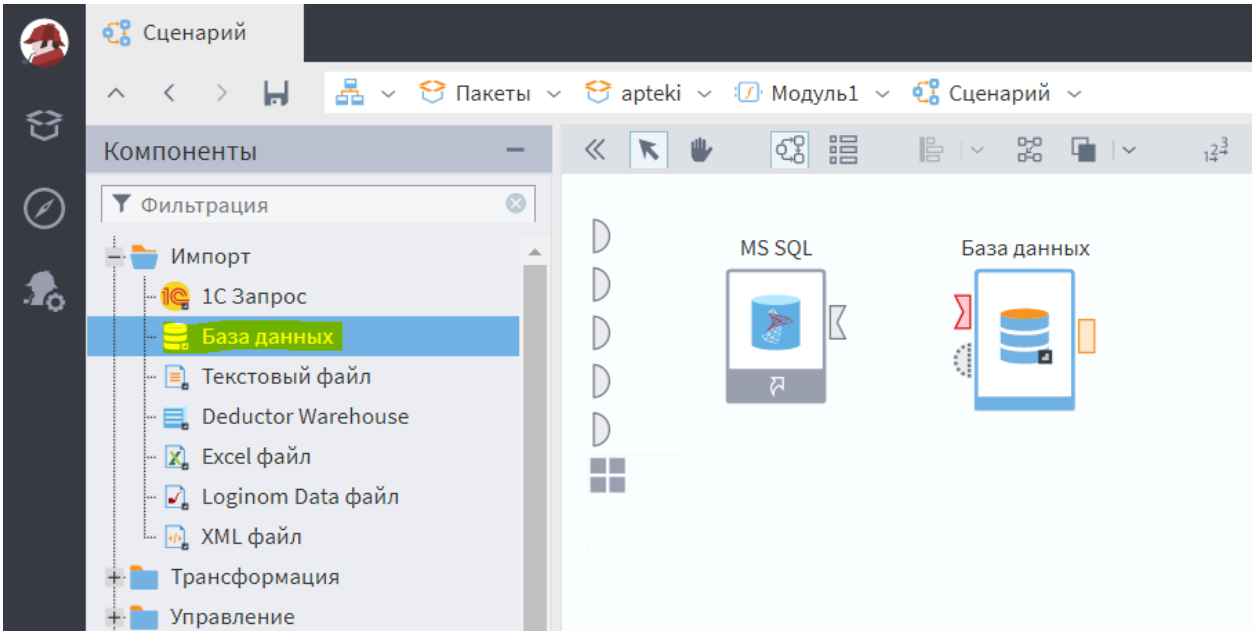
Центральная рабочая область в Loginom – это область Сценария, который есть в каждом Модуле. Именно здесь происходит основная работа.

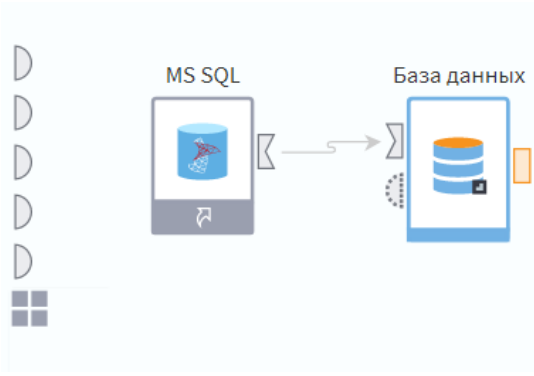


Перейдем в окно **Сценария** и из вкладки **Подключения** добавим первый узел сценария – только что созданное подключение:

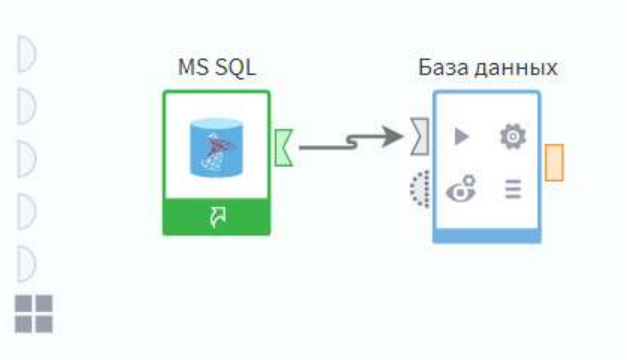


Затем из секции **Компоненты** добавим в сценарий узел **База данных**. В этом узле настраиваются связи с данными из исходных источников.



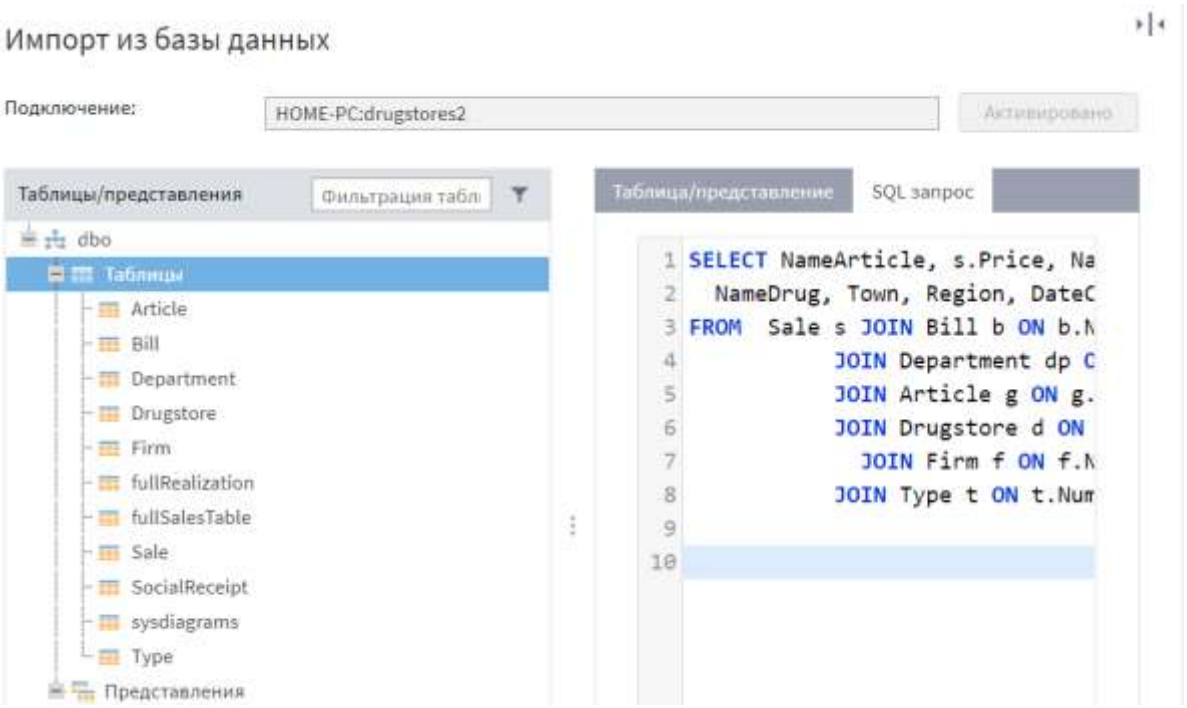


Обратите внимание, что у узлов по бокам есть маленькие цветные фигурки – это так называемые **порты ввода-вывода**, через которые узлы передают данные друг другу. Свяжем **выходной** порт MS SQL и **входной** порт Базы данных (перетащим мышкой). Между ними появится направленная стрелка.



Зеленый цвет узла показывает, что он активен. Активизировать узел можно через контекстное меню, которое вызывается правой кнопкой мыши.

Если щелкнуть мышью по какому-нибудь узлу, то на нем появятся маленькие пиктограммы, в том числе **шестеренка.** Это режим настройки узла. Щелкнем по нему мышью и попадем в следующее окно:

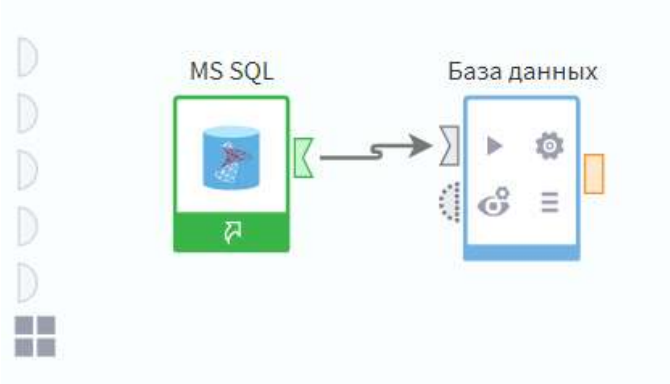


Здесь можно указать, из какой таблицы или запроса мы будем брать данные в SQL Server. Скопируем в окно SQL запроса тот запрос, который мы создали в 3 задании в SQL server для выборки всей информации из всех 7 таблиц базы данных.

Предпросмотр

#	NameArticle	Price	NameTy...	NameFi...	NameD...	NameDrug	Town	Region	DateCheck
1	Товар 1	185,22	антибиотик	Фирма 1	Отдел 1	Аптека 1	Город 1	Регион 1	14.02.2016, 00:
2	Товар 1	185,22	антибиотик	Фирма 1	Отдел 140	Аптека 35	Город 7	Регион 2	13.04.2016, 00:
3	Товар 1	179,55	антибиотик	Фирма 1	Отдел 29	Аптека 8	Город 2	Регион 1	11.11.2016, 00:
4	Товар 1	185,22	антибиотик	Фирма 1	Отдел 61	Аптека 16	Город 4	Регион 2	29.12.2016, 00:
5	Товар 1	189,00	антибиотик	Фирма 1	Отдел 59	Аптека 15	Город 3	Регион 1	06.01.2016, 00:
6	Товар 1	179,55	антибиотик	Фирма 1	Отдел 173	Аптека 44	Город 9	Регион 2	23.07.2016, 00:
7	Товар 1	185,22	антибиотик	Фирма 1	Отдел 67	Аптека 17	Город 4	Регион 2	23.09.2016, 00:
8	Товар 1	189,00	антибиотик	Фирма 1	Отдел 28	Аптека 7	Город 2	Регион 1	20.10.2016, 00:
9	Товар 1	189,00	антибиотик	Фирма 1	Отдел 199	Аптека 50	Город 10	Регион 2	21.10.2016, 00:
10	Товар 1	189,00	антибиотик	Фирма 1	Отдел 106	Аптека 27	Город 6	Регион 2	27.11.2016, 00:
11	Товар 1	189,00	антибиотик	Фирма 1	Отдел 129	Аптека 33	Город 7	Регион 2	19.01.2016, 00:
12	Товар 1	185,22	антибиотик	Фирма 1	Отдел 52	Аптека 13	Город 3	Регион 1	20.12.2016, 00:
13	Товар 1	185,22	антибиотик	Фирма 1	Отдел 122	Аптека 31	Город 7	Регион 2	18.11.2016, 00:
14	Товар 1	179,55	антибиотик	Фирма 1	Отдел 157	Аптека 40	Город 8	Регион 2	20.11.2016, 00:

Сразу же выполнили Предпросмотр, чтобы убедиться, что данные на месте.



Теперь щелкнем на пиктограмме **Базы данных** с «глазом». Такой картинкой обозначается режим **Визуализаторов**. К визуализаторам относятся: **Таблица, Куб, Диаграмма, Статистика**. Начнем с Таблицы.

Нужно перетащить слово Таблица в область с надписью **Добавить визуализатор** или же щелкнуть мышью по значку «плюс» в этой области.

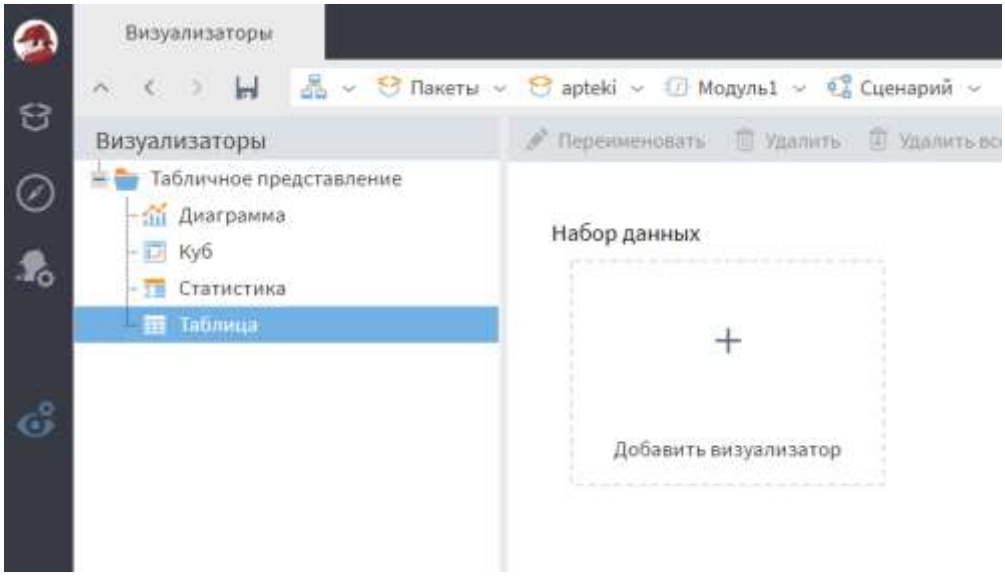
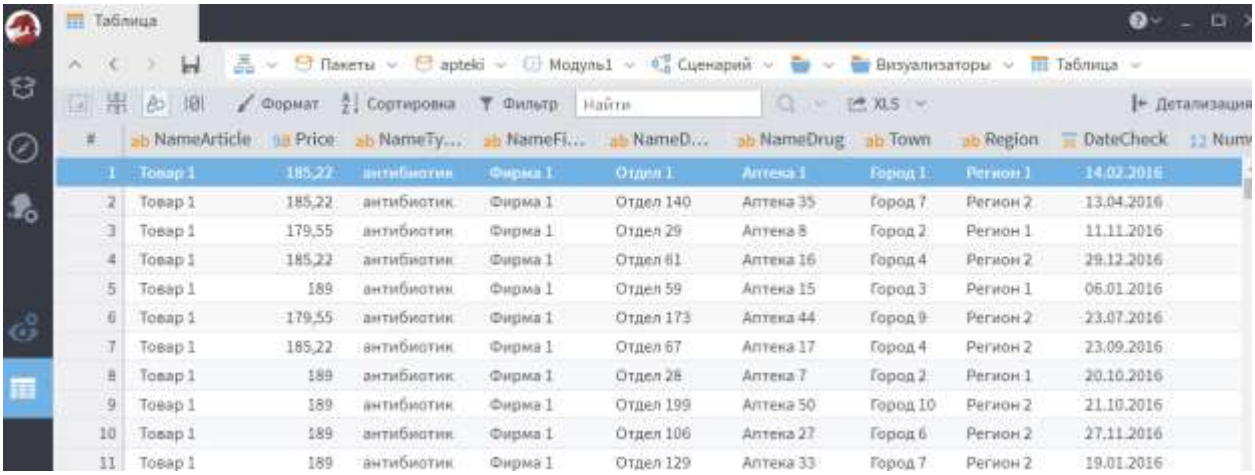
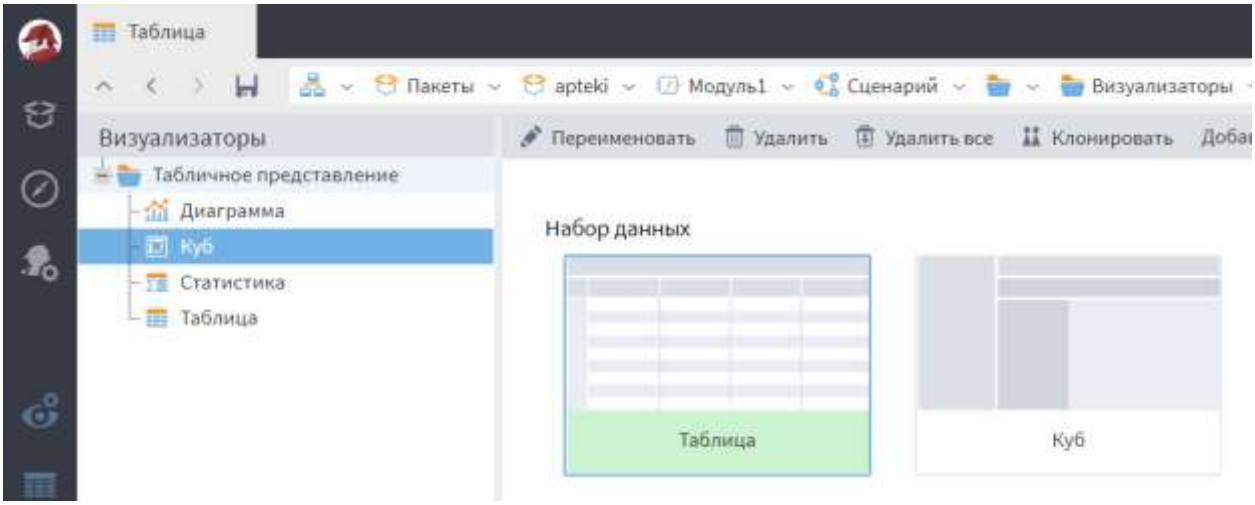


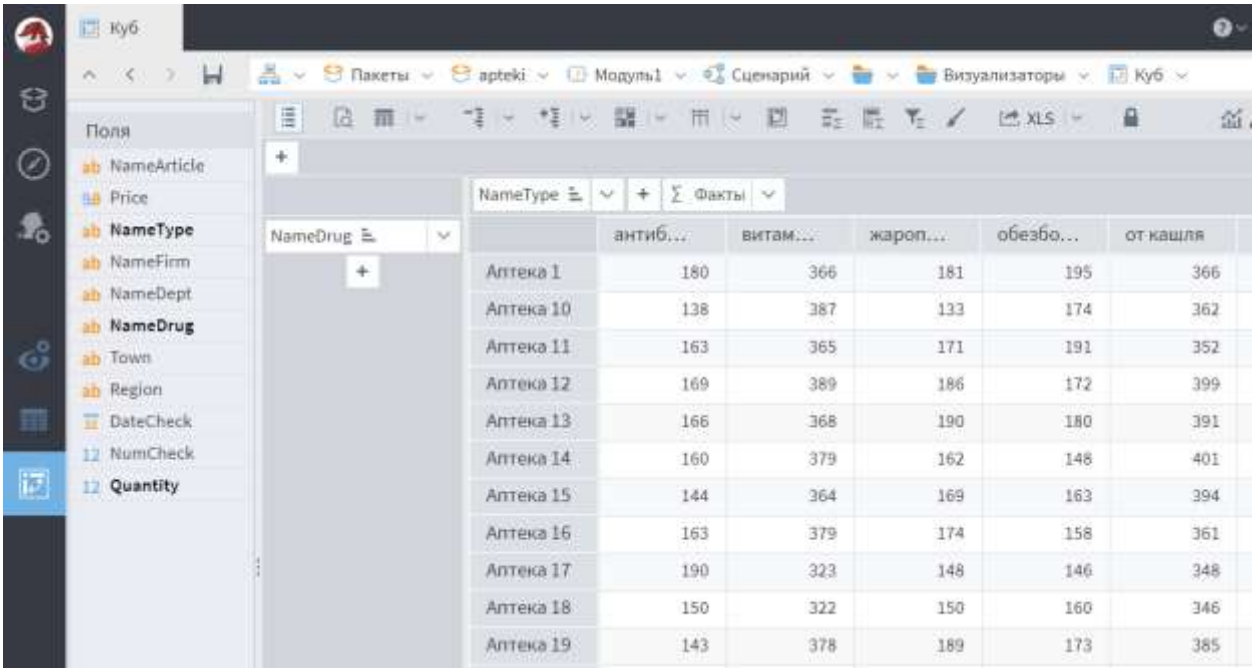
Таблица является самым простым визуализатором. Здесь можно также отсортировать или отфильтровать данные. Сортировка и фильтрация действуют только на текущий просмотр данных и не наследуются другими узлами!



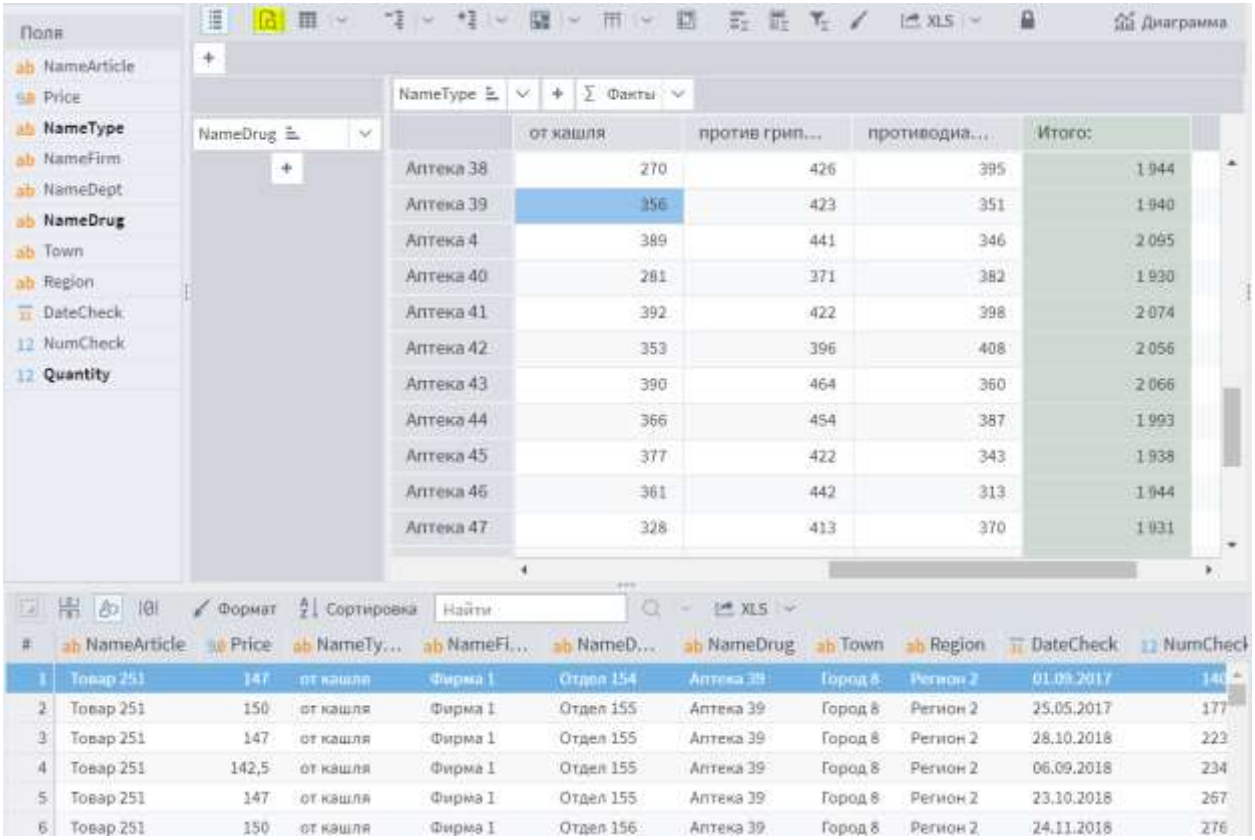
Гораздо более интересным является визуализатор **Куб**. Добавим такой визуализатор и щелкнем по нему.



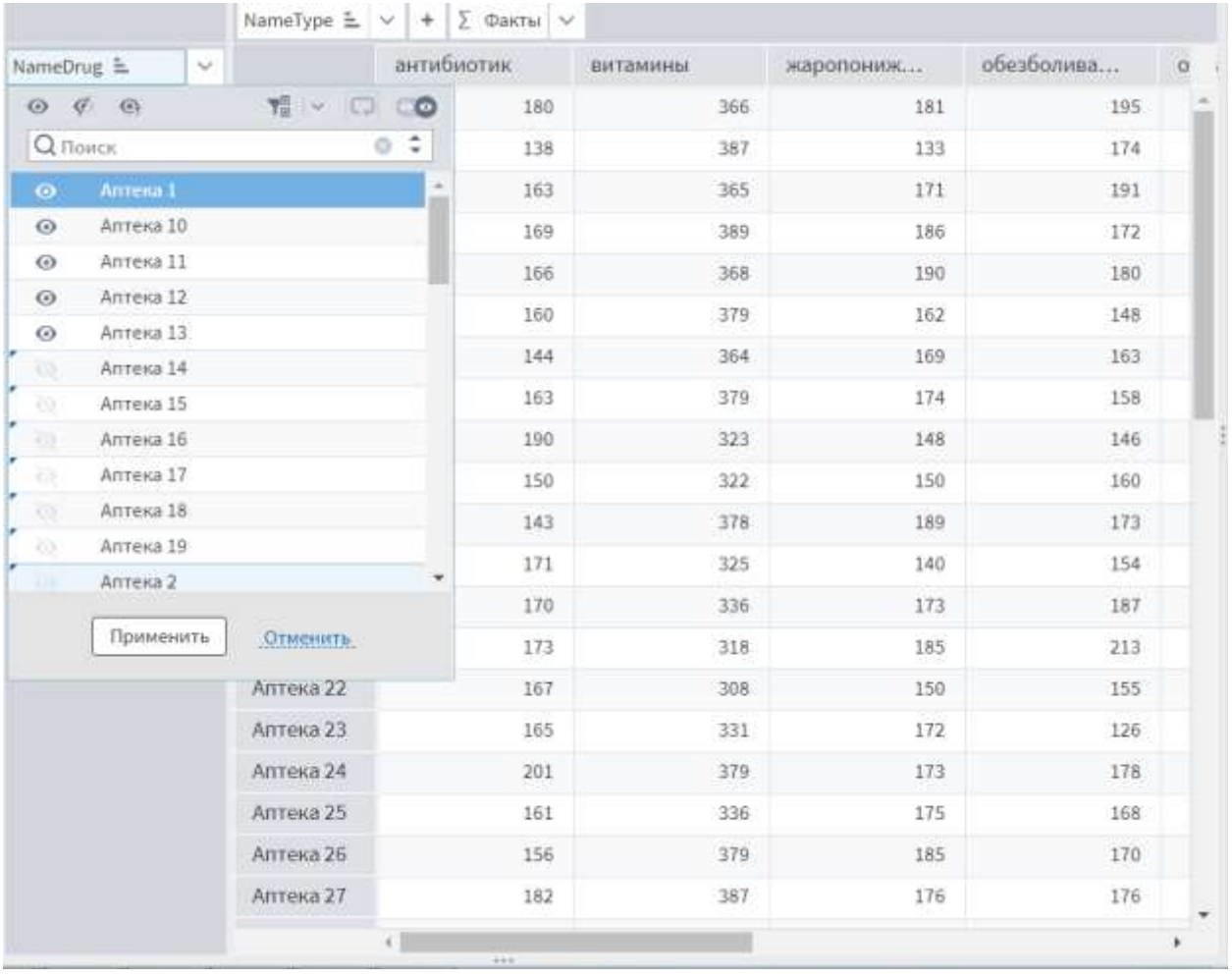
Для двухмерного куба надо настроить *как минимум* одно измерение для строк, одно измерение для столбцов и факты. Просто перетащим соответствующие столбцы слева из списка в области куба. Для столбцов мы выбрали измерение **Тип лекарств** (NameType), для строк – **Название аптеки** (NameDrug), для фактов – **Количество** (Quantity). В левом углу куба есть еще одна кнопка «плюс», с ее помощью в дальнейшем будем добавлять атрибуты куба и скрытые измерения.



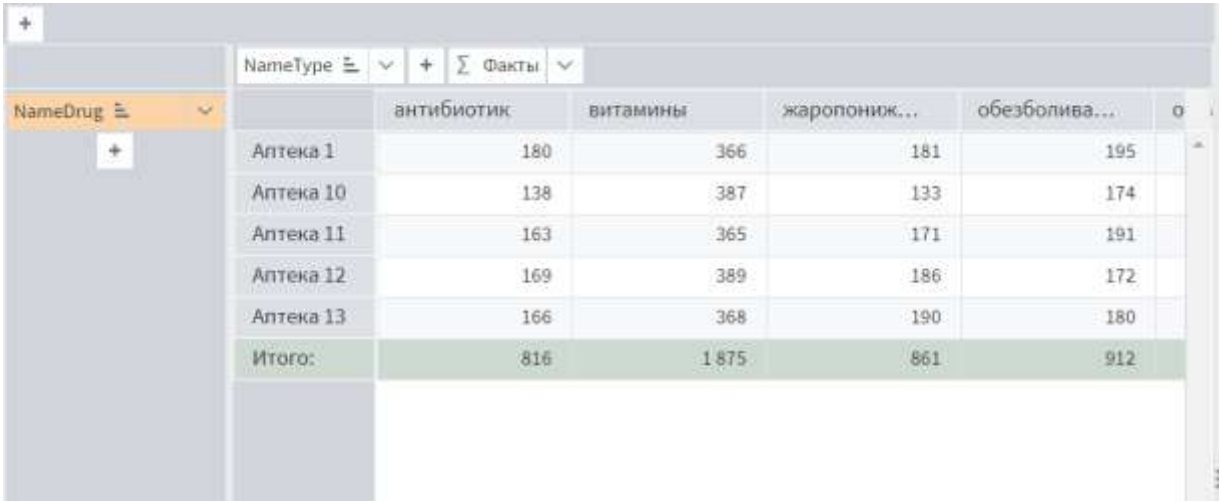
Строка пиктограмм над кубом содержит дополнительные полезные средства. В частности, пиктограмма с лупой позволяет включить окно **Детализации**. Очевидно, что куб обычно содержит агрегированные данные, а в окне Детализации мы можем посмотреть исходную детальную информацию для текущей ячейки куба:



Здесь также можно отфильтровать данные. Щелкнем на названии измерения в строках куба. Появится окно, в котором можно



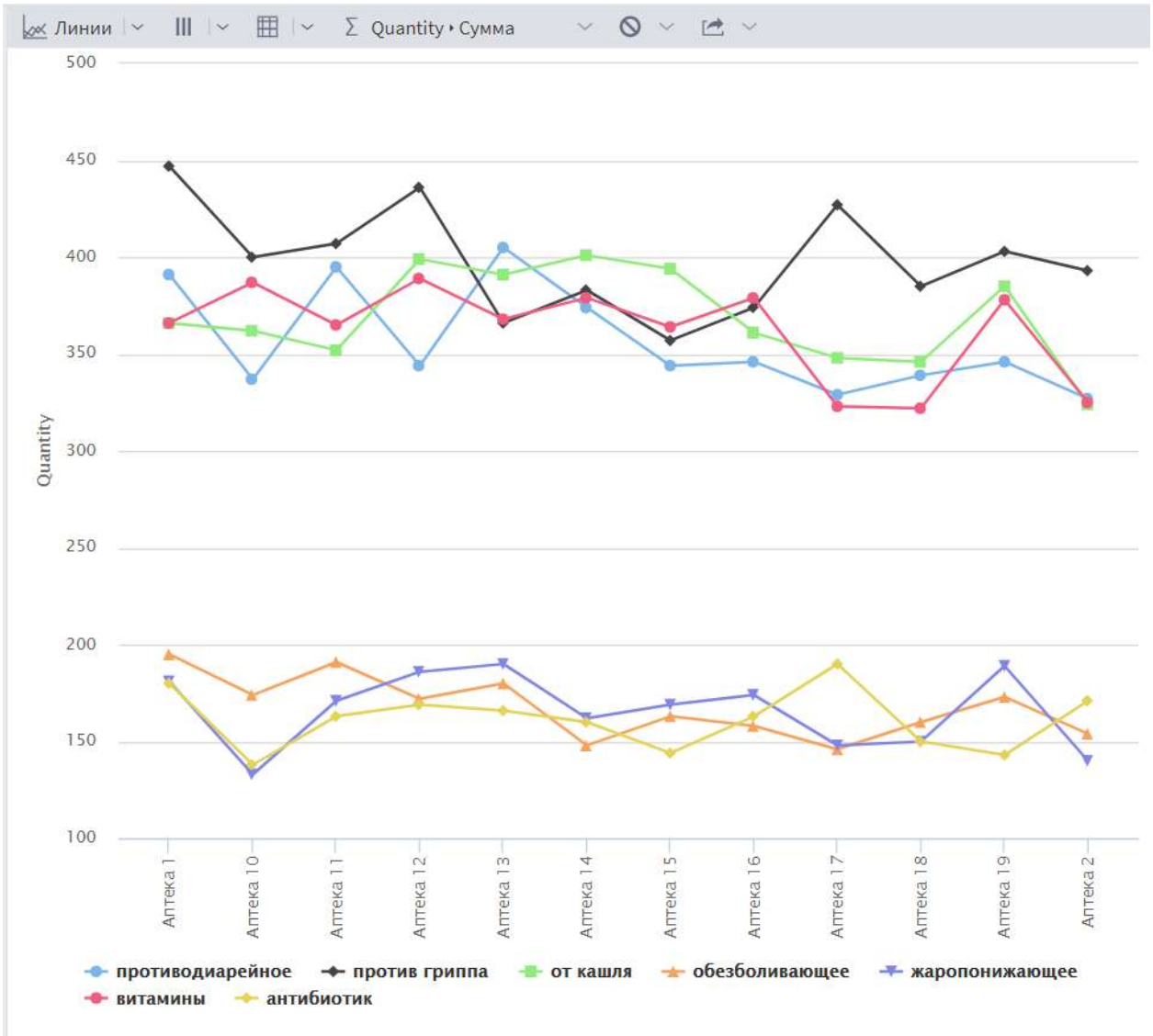
выбрать нужные значения измерения. Например, выберем только несколько аптек из общего списка:



NameDrug	антибиотик	витамины	жаропониж...	обезболива...
Аптека 1	180	366	181	195
Аптека 10	138	387	133	174
Аптека 11	163	365	171	191
Аптека 12	169	389	186	172
Аптека 13	166	368	190	180
Итого:	816	1 875	861	912

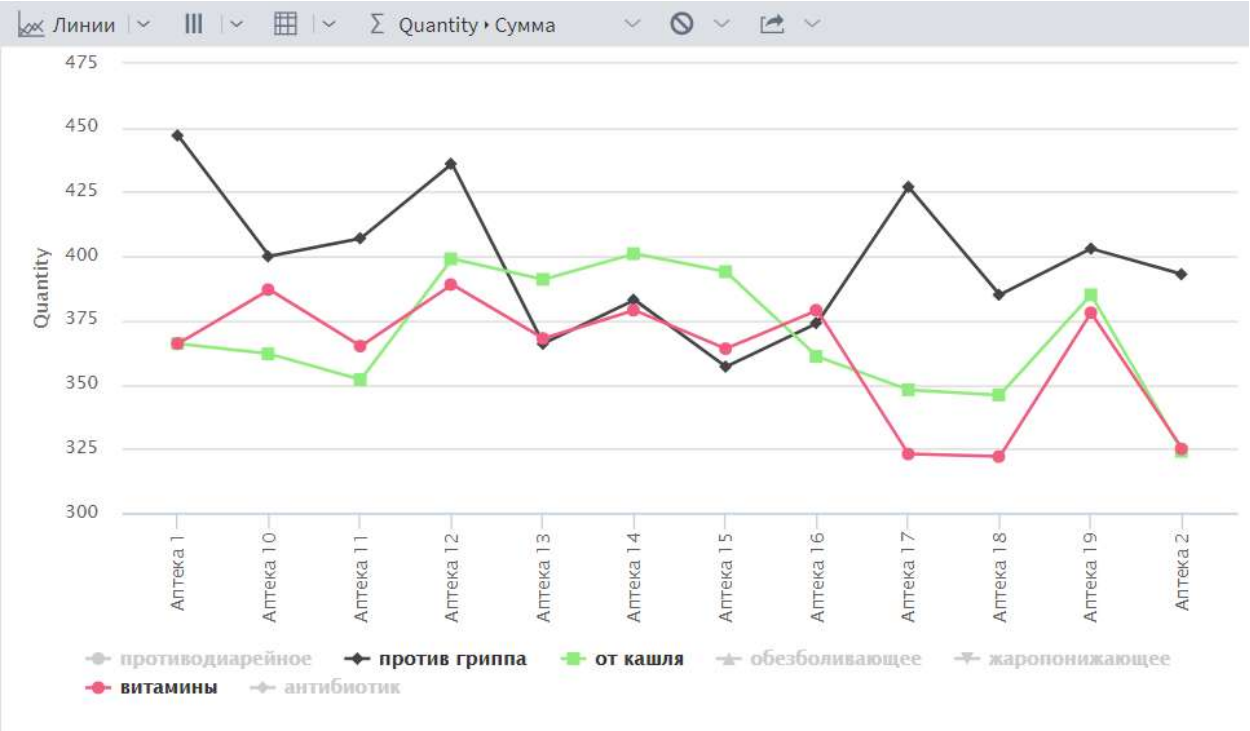
Название измерения изменило цвет. Это означает, что по данному измерению задан фильтр.

Наконец, для куба можно создать диаграмму (здесь мы взяли 12 аптек).



При создании диаграмм обязательно нужно обеспечивать их «читабельность». Не создавайте диаграммы на основании кубов, в которых десятки и сотни строк. Для удобного восприятия данных их нужно агрегировать либо фильтровать.

Фильтровать серии данных можно прямо на диаграмме, если щелкнуть по соответствующим значкам на легенде диаграммы. Значок окрашивается в серый цвет, а данные исчезают с области диаграммы, как в следующем примере:



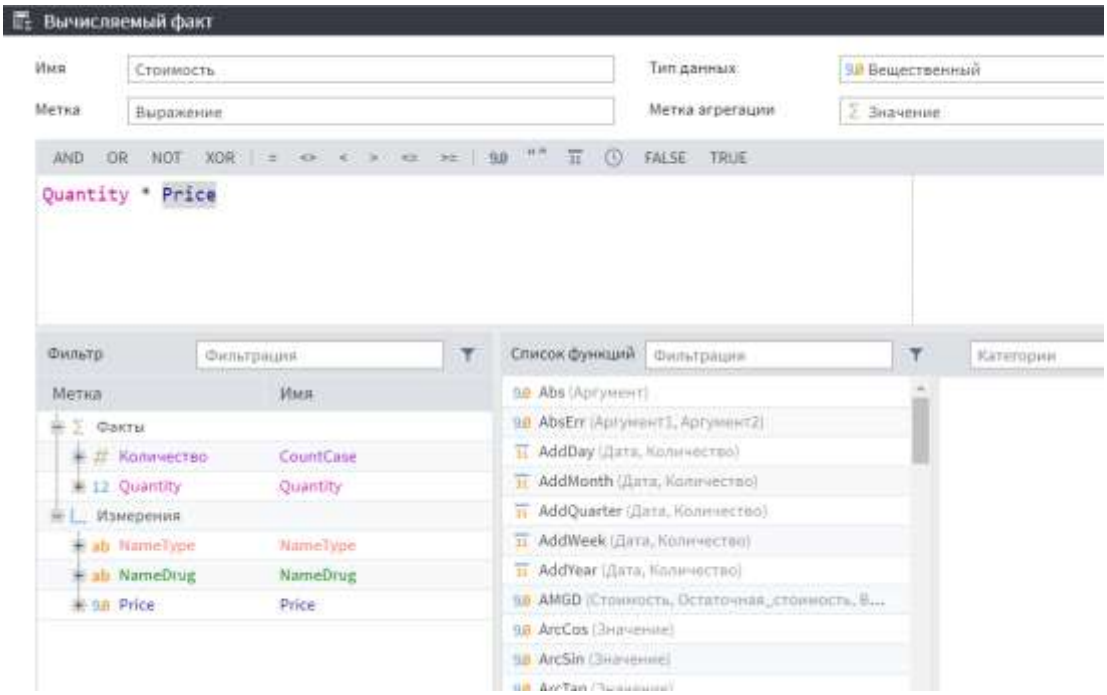
Разумеется, этот фильтр действует только в пределах текущей диаграммы.

Найдите и рассмотрите также в строке пиктограмм разные типы диаграмм и транспонирование данных куба (которое влияет и на вид диаграммы).

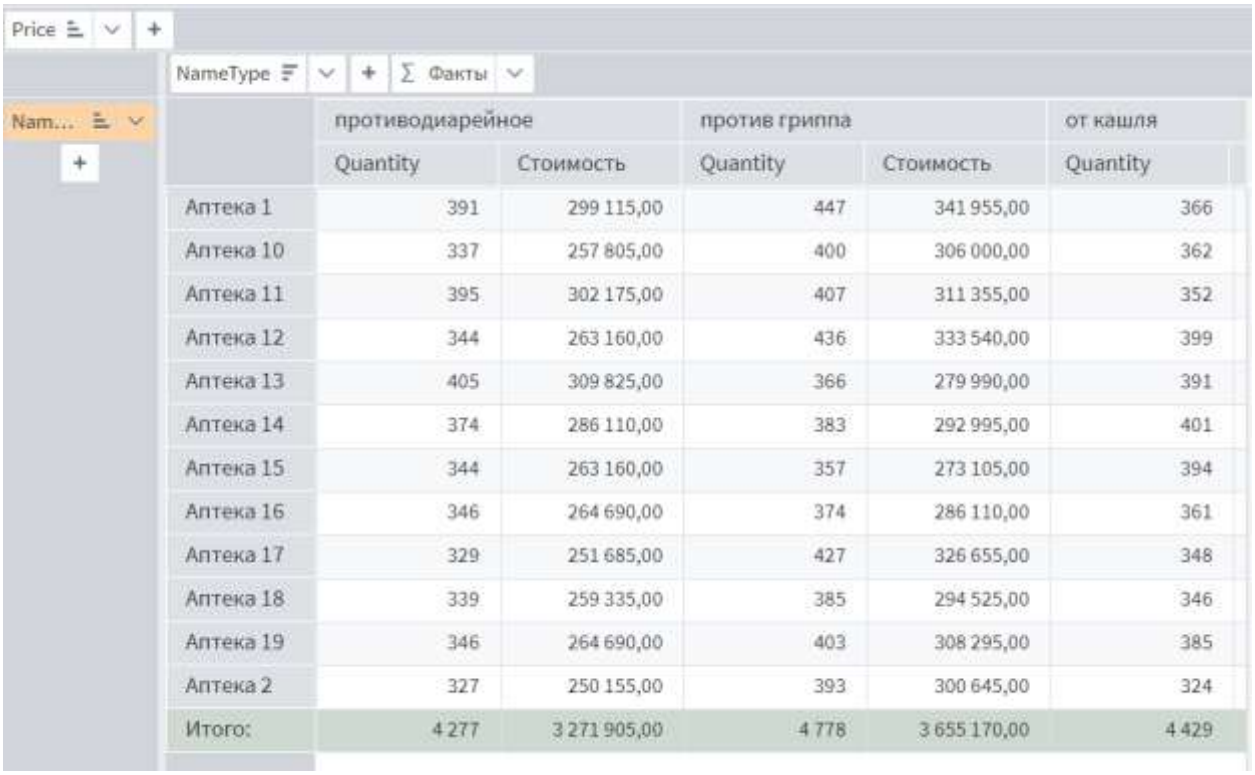
Во всех примерах выше в данном параграфе мы рассматривали количества продаж. А как получить продажи в стоимостном выражении? Есть несколько разных способов.

Можно добавить атрибут **Цена** продажи (Price) в список атрибутов куба (кнопка «плюс» в левом верхнем углу куба), а затем добавить новый факт в список фактов.

Назовем новый факт **Стоимость**, зададим формулу **Количество * Цена** (**Quantity * Price**). Созданный таким образом факт является *локальным*, он не будет наследоваться другими узлами сценария.



Куб примет следующий вид:



Price

NameType | Факты

Nam...

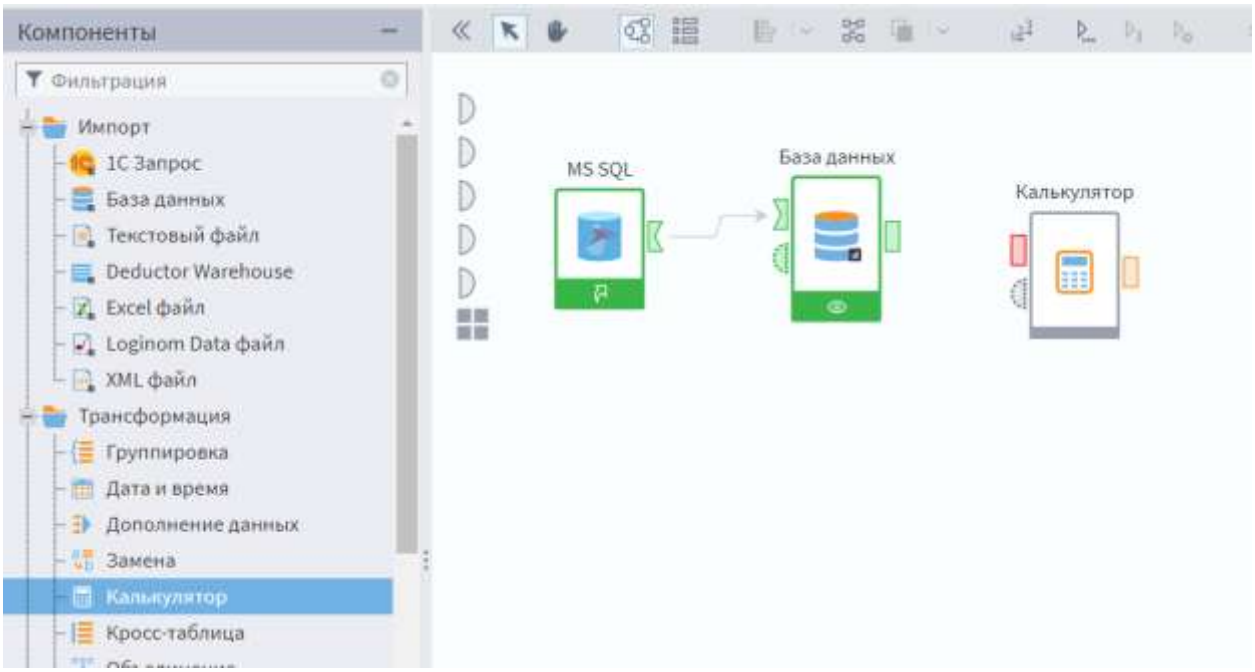
	противодиарейное		против гриппа		от кашля
	Quantity	Стоимость	Quantity	Стоимость	Quantity
Аптека 1	391	299 115,00	447	341 955,00	366
Аптека 10	337	257 805,00	400	306 000,00	362
Аптека 11	395	302 175,00	407	311 355,00	352
Аптека 12	344	263 160,00	436	333 540,00	399
Аптека 13	405	309 825,00	366	279 990,00	391
Аптека 14	374	286 110,00	383	292 995,00	401
Аптека 15	344	263 160,00	357	273 105,00	394
Аптека 16	346	264 690,00	374	286 110,00	361
Аптека 17	329	251 685,00	427	326 655,00	348
Аптека 18	339	259 335,00	385	294 525,00	346
Аптека 19	346	264 690,00	403	308 295,00	385
Аптека 2	327	250 155,00	393	300 645,00	324
Итого:	4 277	3 271 905,00	4 778	3 655 170,00	4 429

Хотя представления куба являются только двухмерными, можно попробовать добавить дополнительные измерения. Например, в строки куба добавим еще одно измерение **Город (Town)** – перетащим его из списка столбцов слева и поместим выше изменения **Аптека.** Получим два уровня измерений в строках куба, с возможностью их «сворачивания». Можно также увидеть частичные итоги по каждой группе:

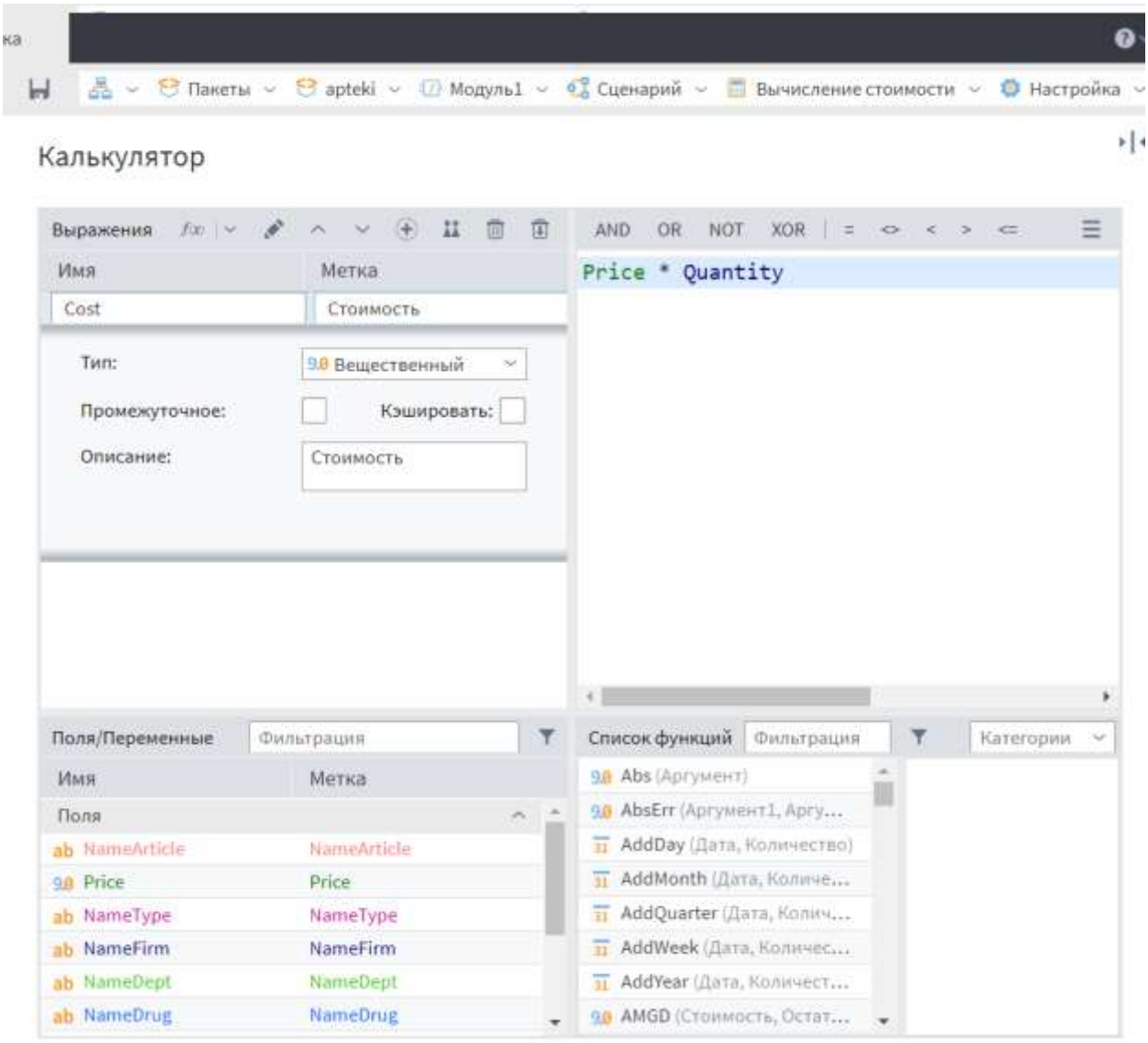
Price | NameType | Факты | Town | Nam...

		противодиарейное		против гриппа	
		Quantity	Стоимость	Quantity	Стоим
Город 1	Аптека 1	391	299 115,00	447	
	Аптека 2	327	250 155,00	393	
	Итого:	718	549 270,00	840	
Город 2	Аптека 10	337	257 805,00	400	
	Итого:	337	257 805,00	400	
Город 3	Аптека 11	395	302 175,00	407	
	Аптека 12	344	263 160,00	436	
	Аптека 13	405	309 825,00	366	
	Аптека 14	374	286 110,00	383	
	Аптека 15	344	263 160,00	357	
	Итого:	1 862	1 424 430,00	1 949	
Город 4		1 360	1 040 400,00	1 589	
Итого:		4 277	3 271 905,00	4 778	

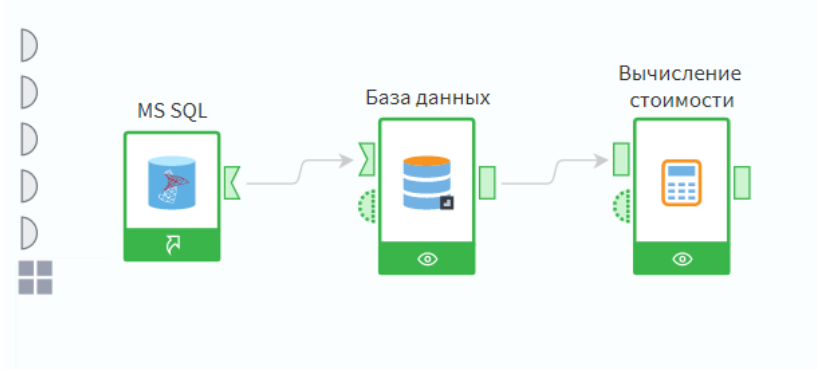
Напомним, что стоимость продаж, вычисленная внутри куба, не будет видна в других узлах. Рассмотрим другой способ получения стоимостей продаж. В рабочую область сценария добавим новый узел **Калькулятор** из группы **Трансформация:**



Выходной порт узла **База данных** соединим с входным (прямоугольным) портом узла **Калькулятор**. Далее зайдем в настройку калькулятора (через контекстное меню правой кнопкой мыши или пиктограмму с шестеренкой):



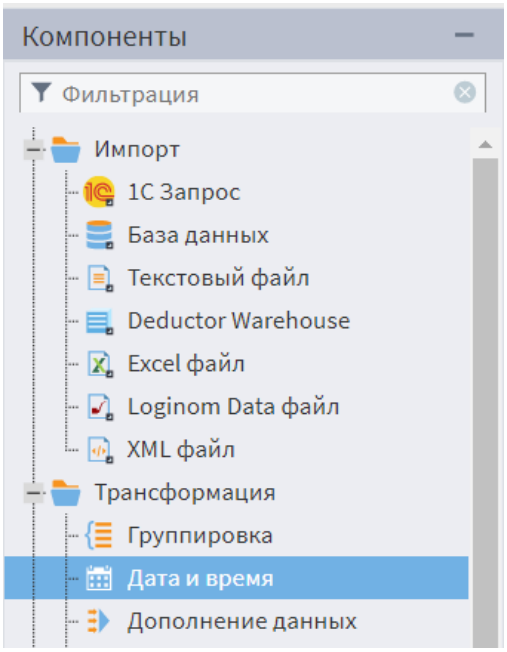
Здесь мы снова задаем ту же формулу **Количество * Цена** (**Quantity * Price**). Для удобства также переименуем калькулятор, назовем его Вычисление стоимости.



Создадим для этого узла визуализатор Куб. Столбец **Стоимость** появился в списке столбцов данных, и он будет также доступен и в последующих узлах сценария.

Используем стоимость в качестве фактов куба:

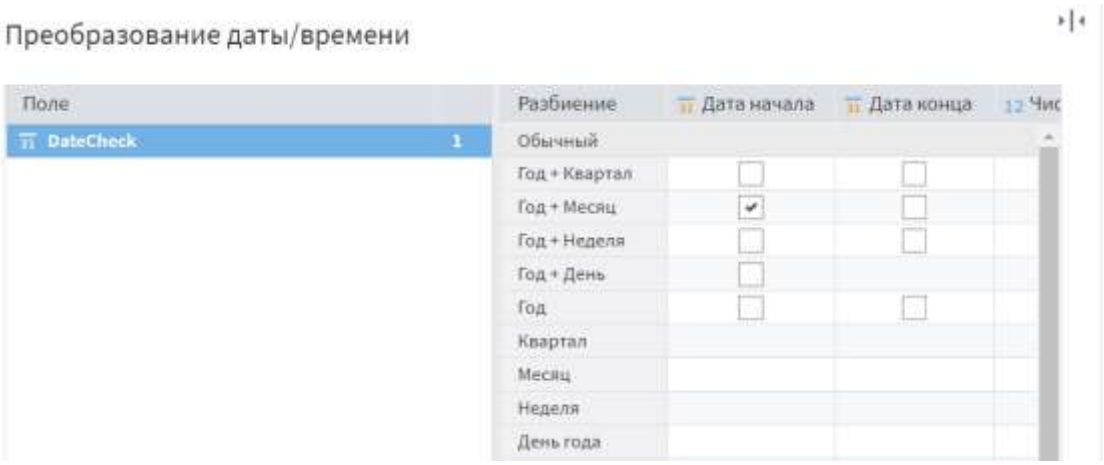
Town	Фирма 1	Фирма 10	Фирма 11	Фирма 12	Фирма 13
Город 1	182 711,63	172 455,82	182 480,77	92 128,63	185 436,34
Город 10	176 582,14	161 222,85	204 757,14	84 893,17	142 921,85
Город 2	194 290,80	157 604,01	184 516,27	108 929,75	141 316,87
Город 3	204 442,22	161 220,15	153 666,84	72 717,13	164 222,51
Город 4	177 298,27	149 013,04	175 137,71	90 061,16	128 465,57
Город 5	187 490,48	178 235,24	195 533,31	110 629,39	152 531,11
Город 6	182 085,06	158 244,55	180 665,67	104 508,59	170 843,65
Город 7	234 932,69	173 613,85	177 004,59	86 114,76	154 821,69
Город 8	211 748,94	150 563,54	204 152,17	100 025,91	178 676,05
Город 9	223 442,99	164 457,12	196 774,40	128 488,47	176 937,19
Итого:	1 975 025,22	1 626 630,17	1 854 688,87	978 496,96	1 596 172,83



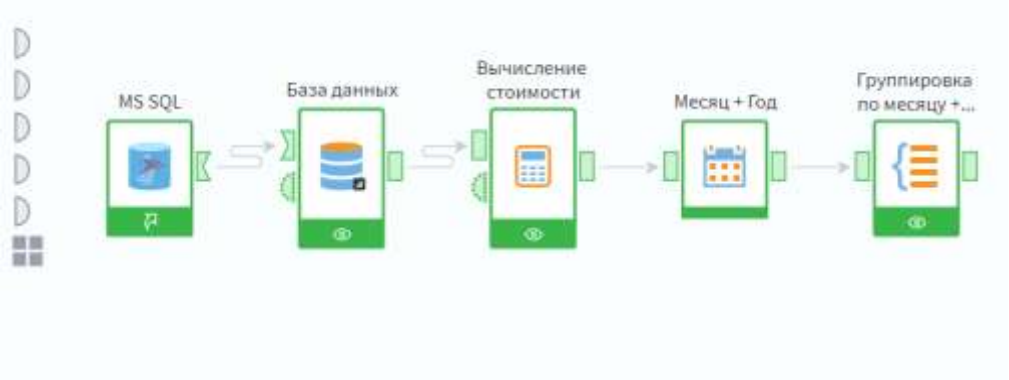
Теперь рассмотрим действия с измерением Дата.

Например, нас интересуют продажи, агрегированные по месяцам и/или годам. Как из столбца **Дата чека (DateCheck)** получить составные части даты?

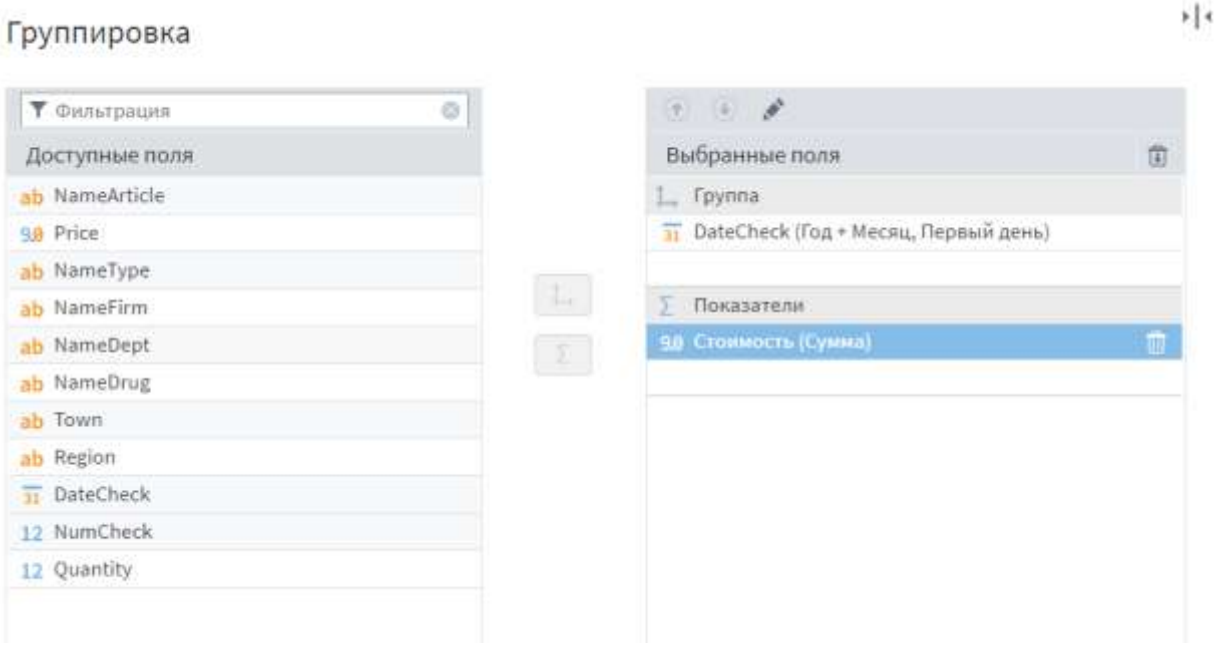
В группе Трансформация в секции Компоненты выберем тип узла **Дата и время** и в качестве вида преобразования выберем «**Год + Месяц**». Будет создан новый столбец типа дата, а его значения – это первые числа каждого месяца.



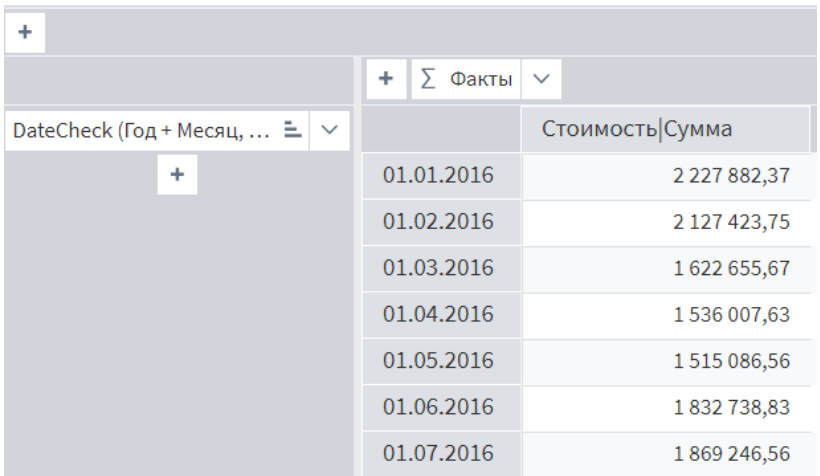
Затем добавим в сценарий узел **Группировка** из группы **Трансформация** секции **Компоненты**. Не забудьте связать новый узел с предыдущим, как на рисунке:



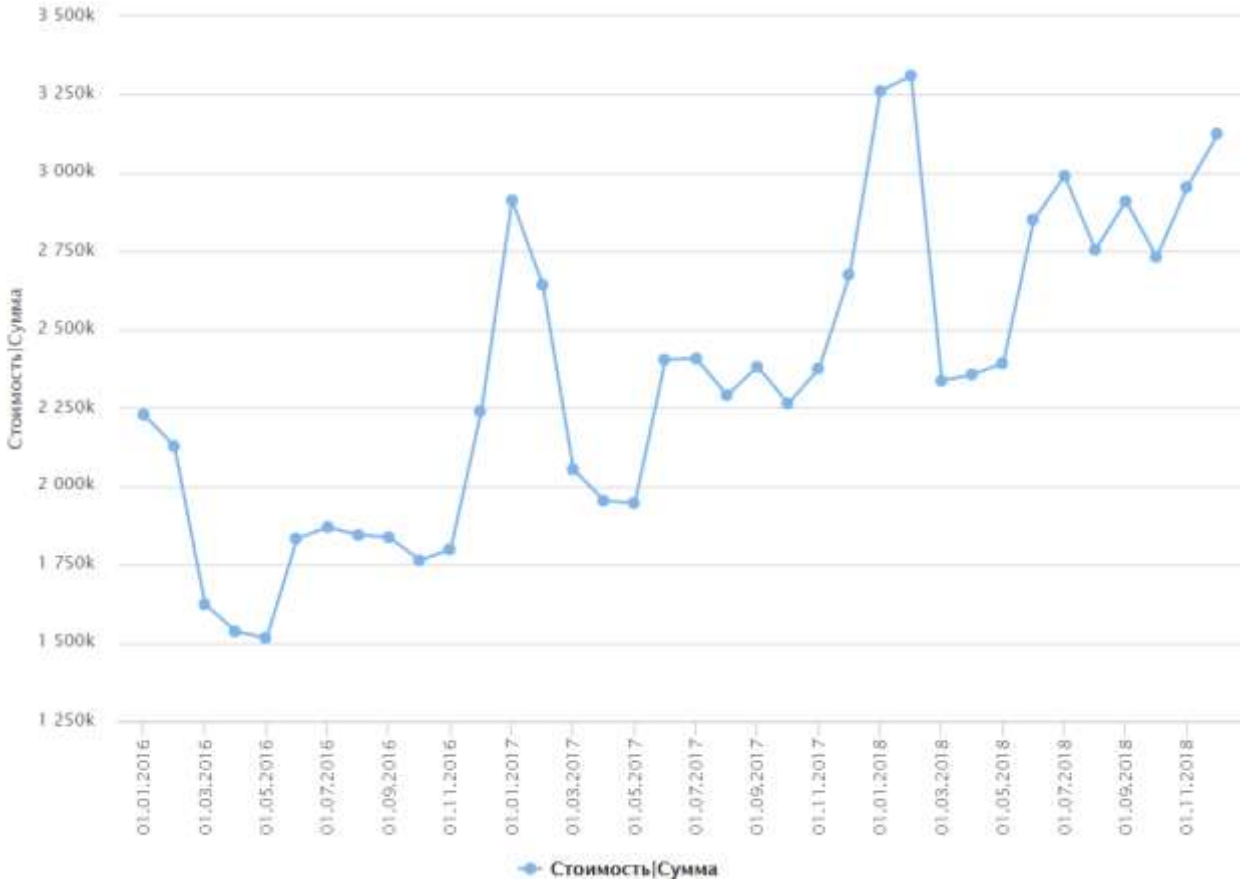
Настройка группировки включает указание столбца группировки (как GROUP BY в SQL) и столбца для подсчета итогов. В качестве группировки мы выбрали **Год+Месяц**, а подсчитывать мы будем суммарную стоимость продаж:



Для текущего узла сценария создадим визуализатор Куб, в котором укажем только одно измерение – Год+Месяц для строк и факты-суммарные стоимости:

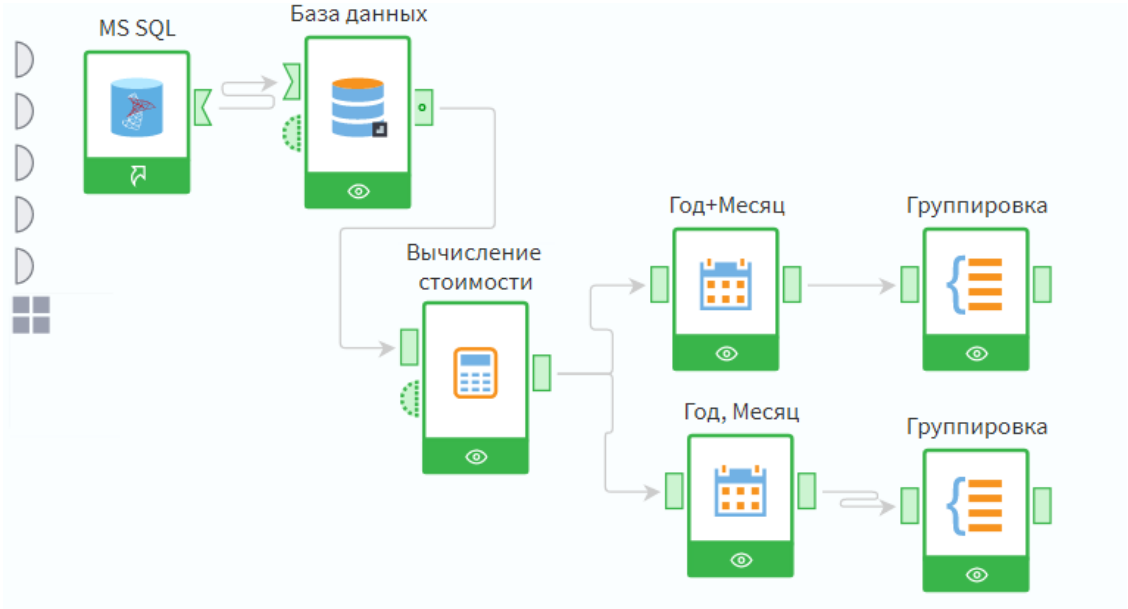


Теперь мы можем построить диаграмму с объемами продаж за 36 месяцев:

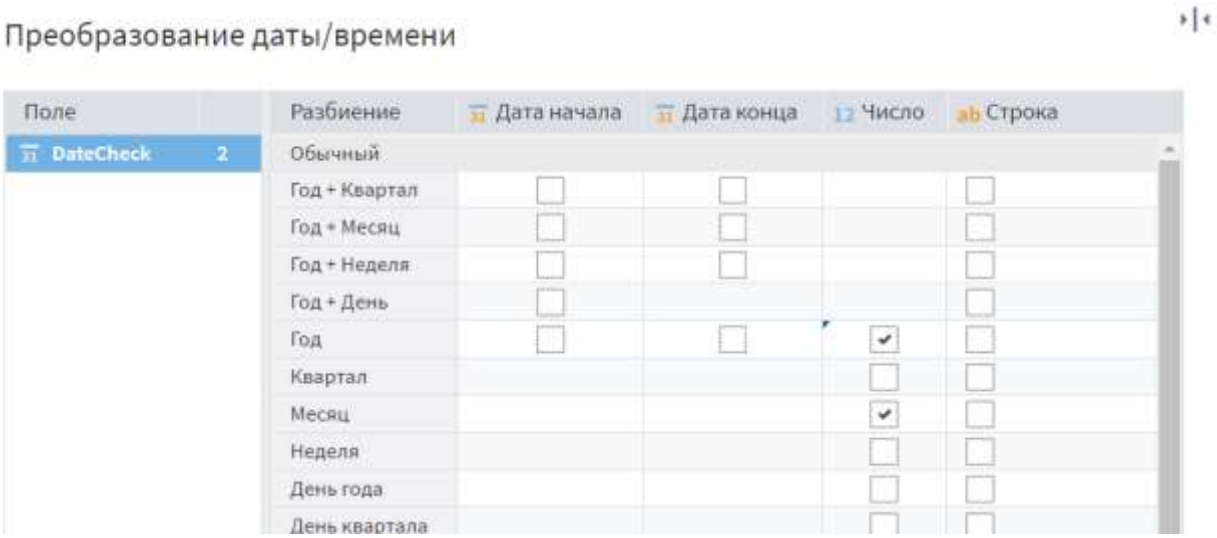


На графике видно, что зимой продажи существенно больше, чем в другие сезоны. Хотелось бы разбить график на 3 года по отдельности и наложить графики друг на друга для сравнения.

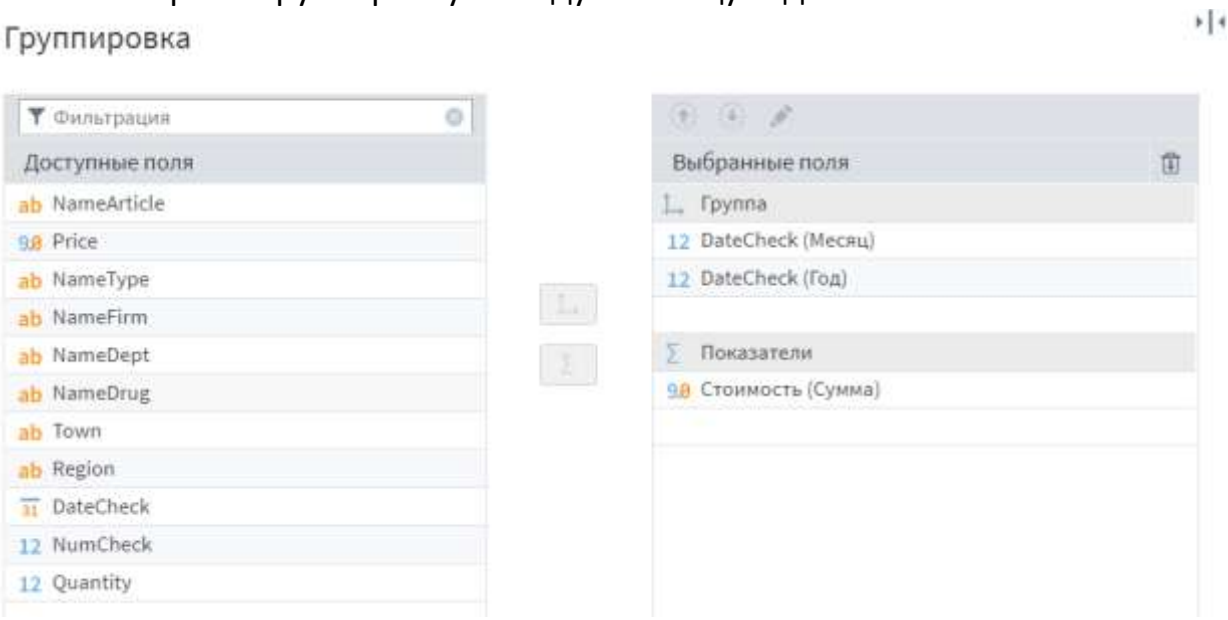
Для этого мы можем использовать другое преобразование даты: получение года и месяца по отдельности. Создадим новые узлы типа **Дата и время** и **Группировка**:



В настройке узла **Дата и время выберем** Год и Месяц по отдельности в формате чисел:

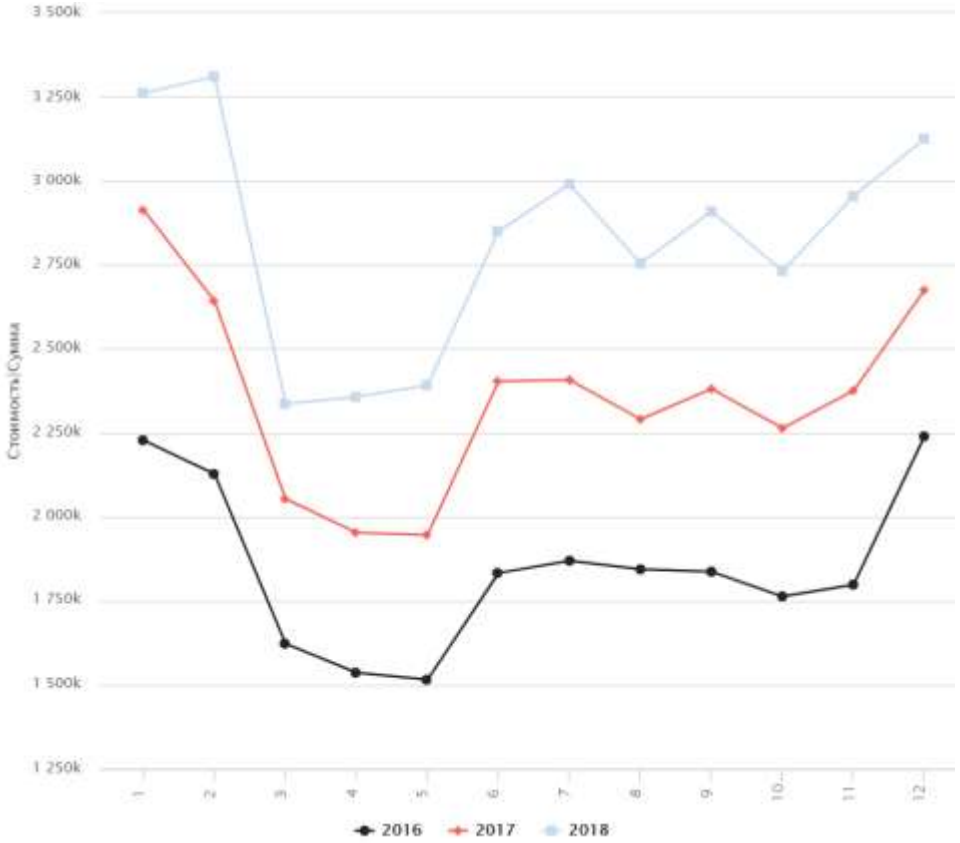


И настроим группировку по году и месяцу отдельно:

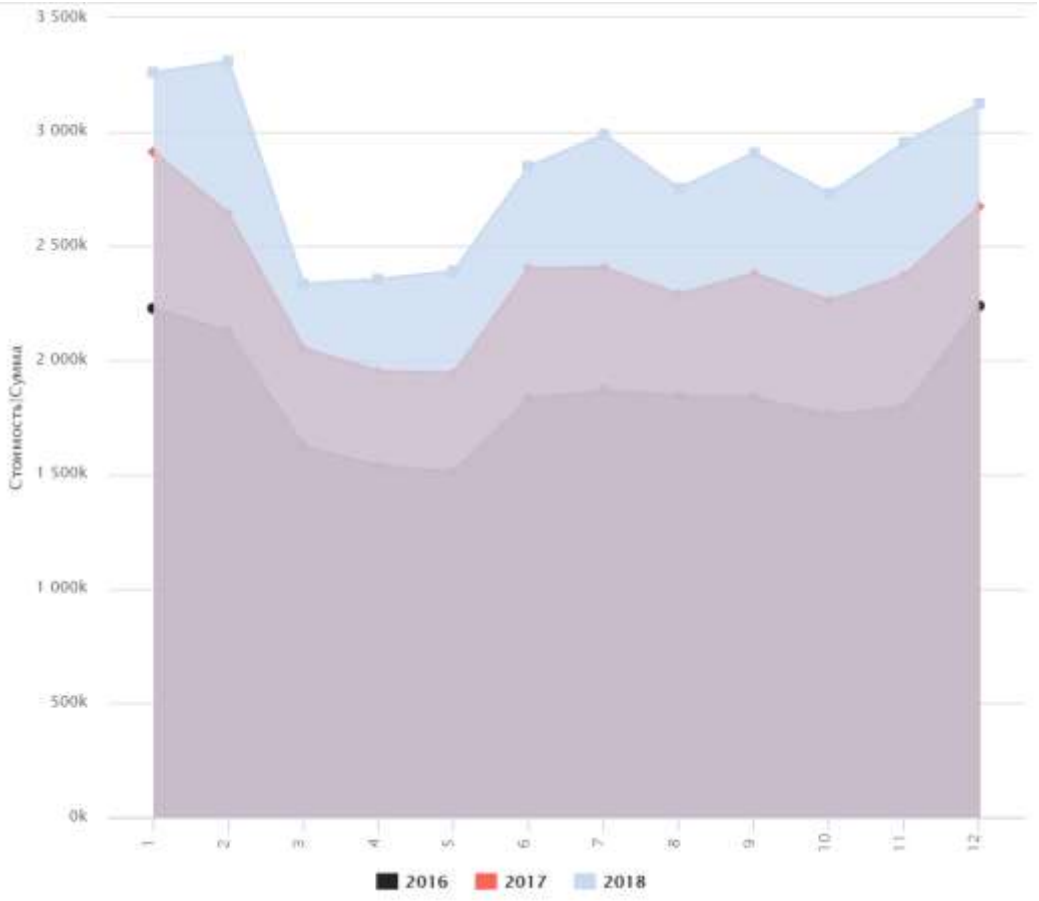


Date...	DateCheck (Год)	2016	2017	2018	Итого:
	1	2 227 882,37	2 912 605,97	3 261 619,73	8 402 108,07
	2	2 127 423,75	2 642 508,10	3 310 446,20	8 080 378,05
	3	1 622 655,67	2 053 339,76	2 336 318,55	6 012 313,98
	4	1 536 007,63	1 953 180,93	2 356 293,47	5 845 482,03
	5	1 515 086,56	1 946 082,56	2 391 798,24	5 852 967,36
	6	1 832 738,83	2 403 617,07	2 850 472,36	7 086 828,26
	7	1 869 246,56	2 407 278,91	2 990 184,01	7 266 709,48
	8	1 843 700,88	2 290 433,37	2 754 641,63	6 888 775,88
	9	1 836 680,71	2 380 761,87	2 909 392,54	7 126 835,12
	10	1 763 077,67	2 263 781,64	2 731 903,69	6 758 763,00
	11	1 798 022,17	2 375 089,43	2 954 137,46	7 127 249,06
	12	2 239 356,42	2 674 739,08	3 125 198,31	8 039 293,81
	Итого:	22 211 879,22	28 303 418,69	33 972 406,19	84 487 704,10

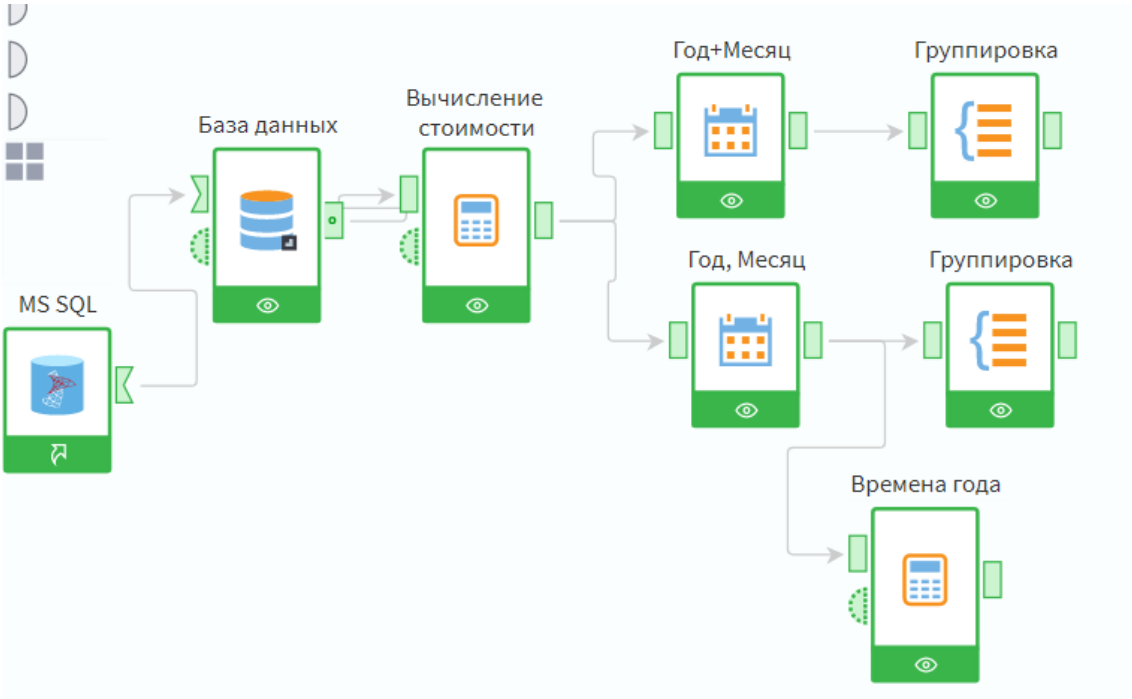
Получили таблицу за 3 года, в ней 12 строк и 3 столбца. Построим линейную диаграмму:



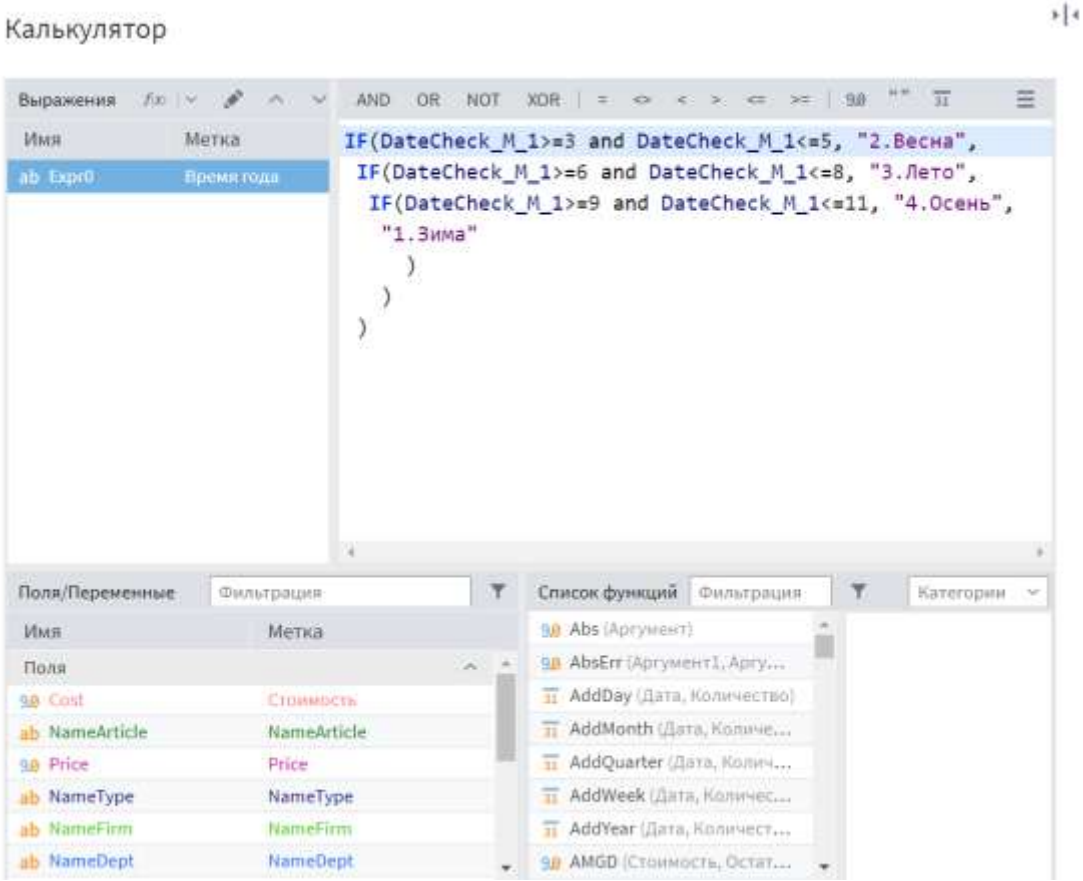
Для этих данных также подойдет диаграмма с областями:



Рассмотрим более сложный пример применения калькулятора. Пусть мы хотим построить диаграмму продаж разных типов лекарств по сезонам. Создадим узел типа **Калькулятор**, в котором потребуются данные о месяце продажи, поэтому разместим его после узла **Год, Месяц**:



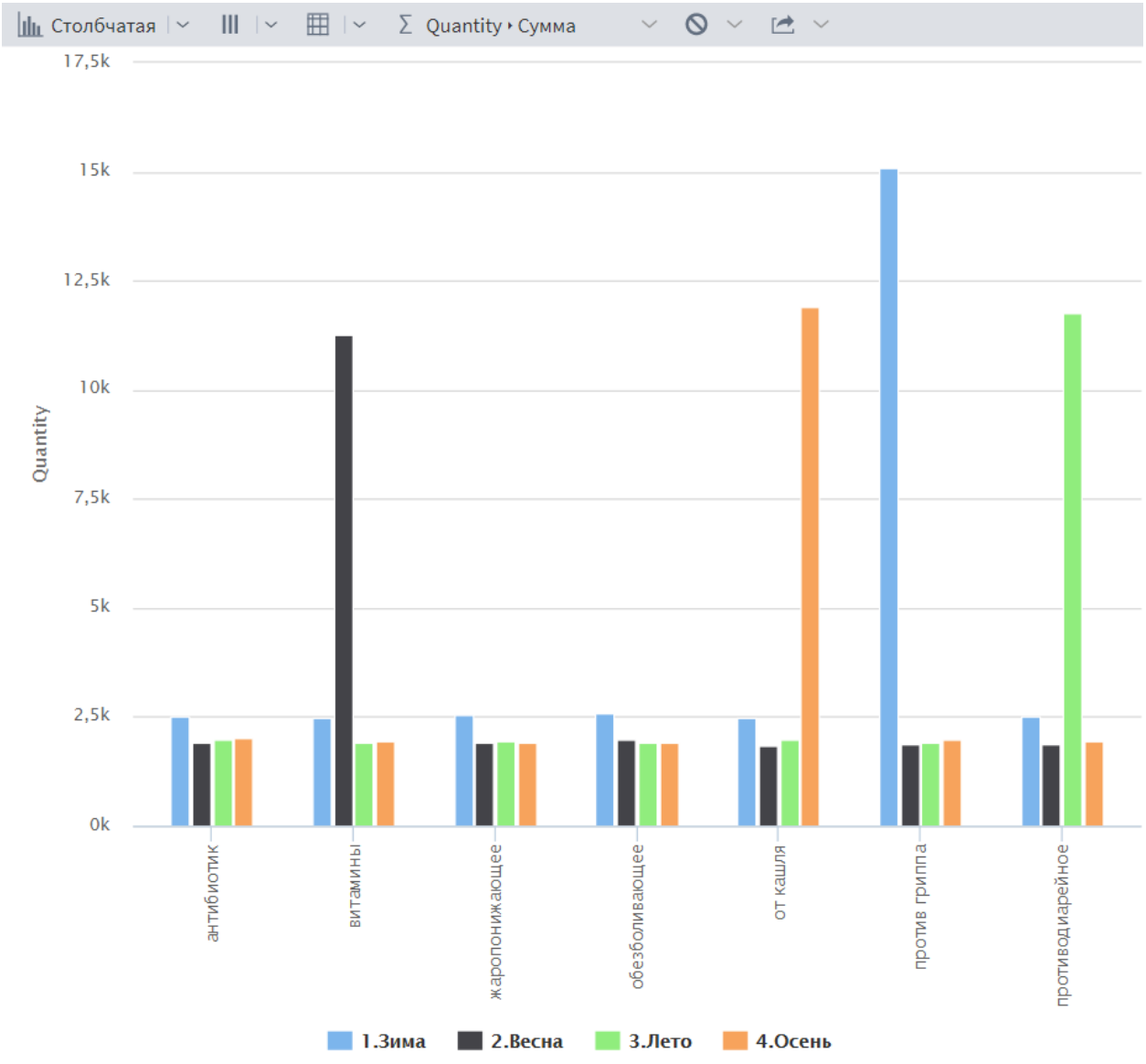
Назовем этот узел **Времена года**. В калькуляторе зададим формулу, которая в зависимости от номера месяца вычисляет название сезона. Сезоны удобно пронумеровать, чтобы они отображались не по алфавиту.



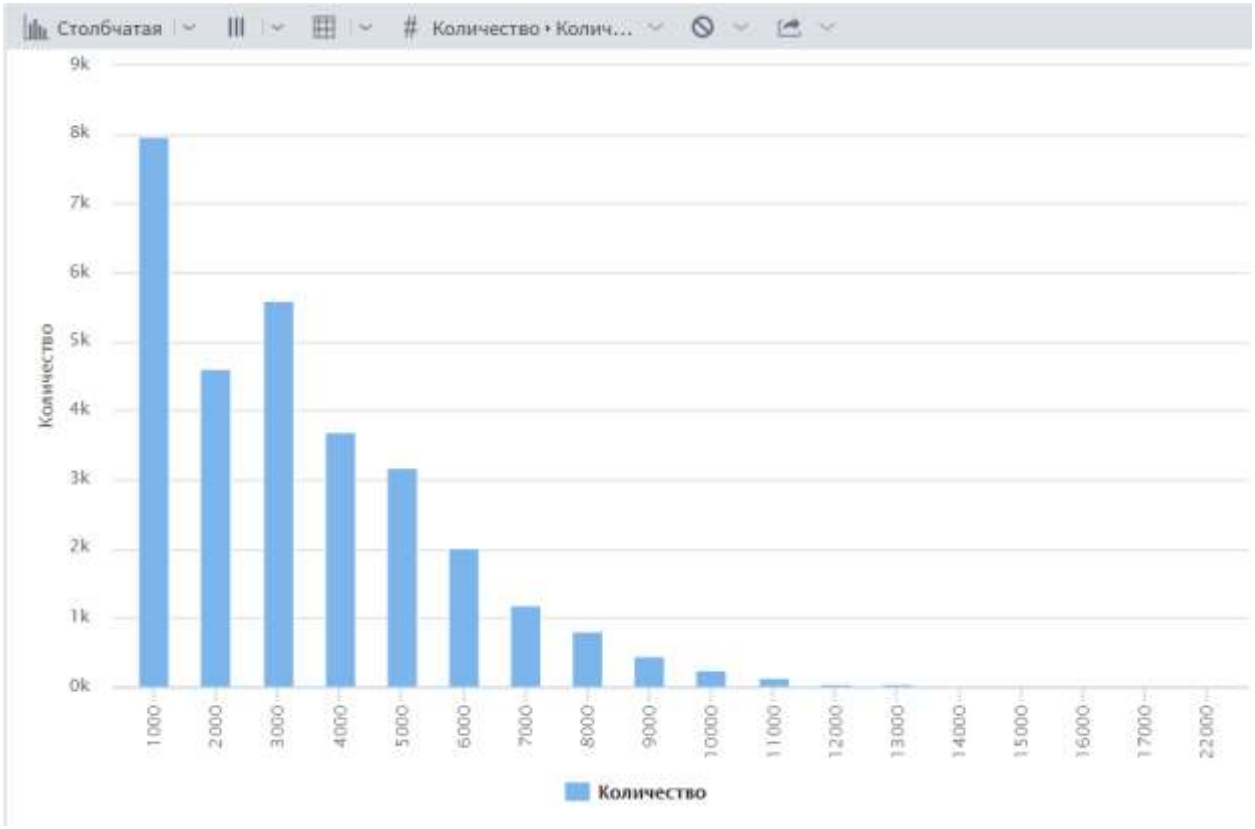
Создадим визуализатор куб

Nam...	Время года: 1.Зима	2.Весна	3.Лето	4.Осень	Итого:
антибио...	2 213 914,87	1 712 754,05	1 752 818,55	1 831 512,43	7 510 999,90
витамины	1 858 335,41	8 287 652,70	1 460 670,98	1 499 452,57	13 106 111,66
жаропо...	1 452 910,22	1 070 086,85	1 118 554,44	1 126 596,14	4 768 147,65
обезбол...	1 944 601,55	1 527 852,95	1 508 208,20	1 505 892,64	6 486 555,34
от кашля	2 417 077,29	1 800 265,66	1 934 797,02	11 436 965,06	17 589 105,03
против г...	12 121 457,60	1 445 569,30	1 572 529,37	1 690 527,42	16 830 083,69
противо...	2 513 482,99	1 866 581,86	11 894 735,06	1 921 900,92	18 196 700,83
Итого:	24 521 779,93	17 710 763,37	21 242 313,62	21 012 847,18	84 487 704,10

и диаграмму. На диаграмме явно видны закономерности, которые мы задавали при генерации данных во втором задании.



Рассмотрим пример более сложного преобразования данных. Пусть мы хотим проанализировать суммы чеков с помощью примерно такой диаграммы:



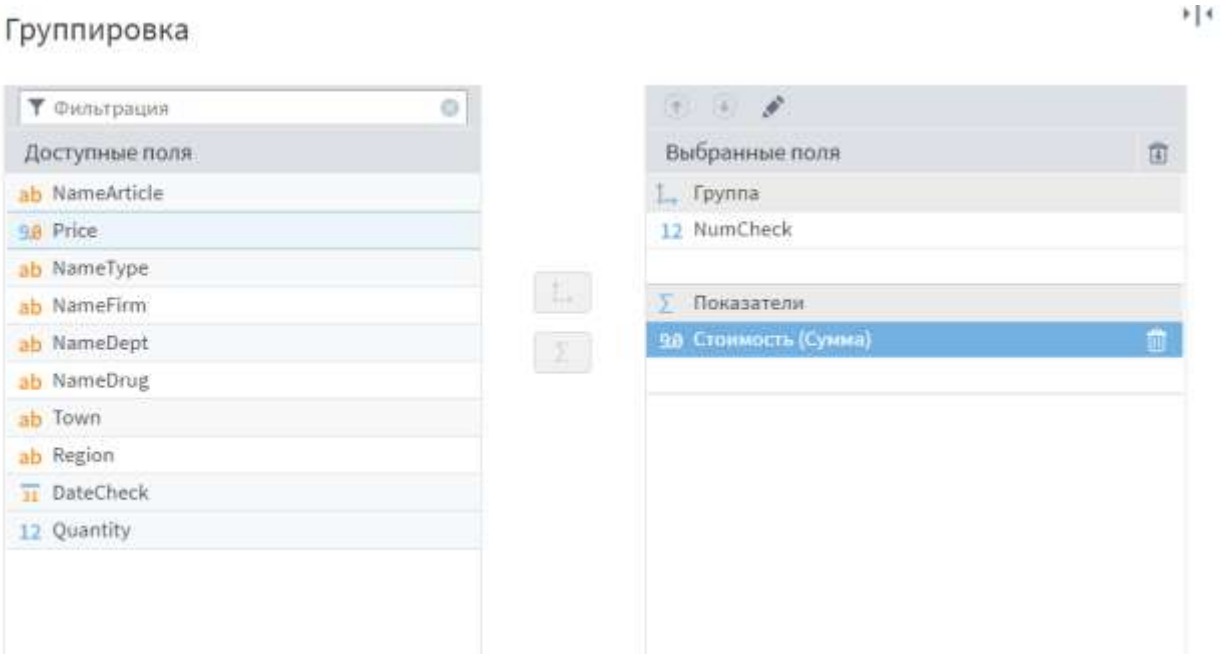
Здесь по горизонтальной оси отложены верхние границы интервалов для суммы чеков, а значения по вертикальной оси означают количество чеков, попавших в каждый интервал.

Столбцы в этой диаграмме означают, что у нас есть

- примерно 8 тысяч чеков с суммой до 1 тыс. руб.
- примерно 4.5 тысяч чеков с суммой от 1 до 2 тыс. руб.,
- примерно 5.5 тысяч чеков с суммой от 2 до 3 тыс. руб.,
- и т.п.

Как видим, для построения такой диаграммы нам не нужны точные суммы чеков, а лишь информация, в какой интервал попадает каждая сумма.

Сначала вычислим суммы чеков с помощью обработки **Группировка**. Такой вид обработки мы уже использовали раньше. Узел Группировка получает данные из узла **Вычисление стоимости** или из узла **База данных**.

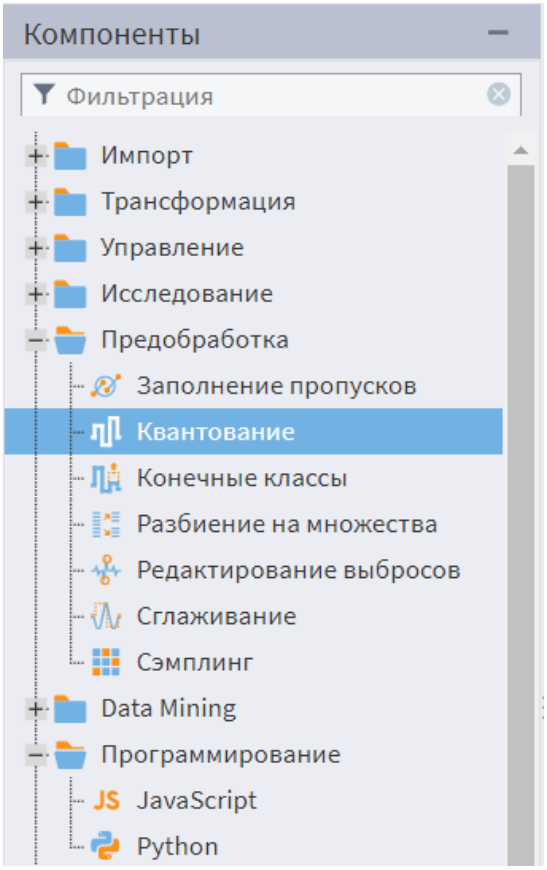


Посмотрим на суммы чеков. На второй картинке суммы упорядочены по убыванию, мы видим, что самый крупный чек на сумму 21 с лишним тыс.

Формат Сортировк

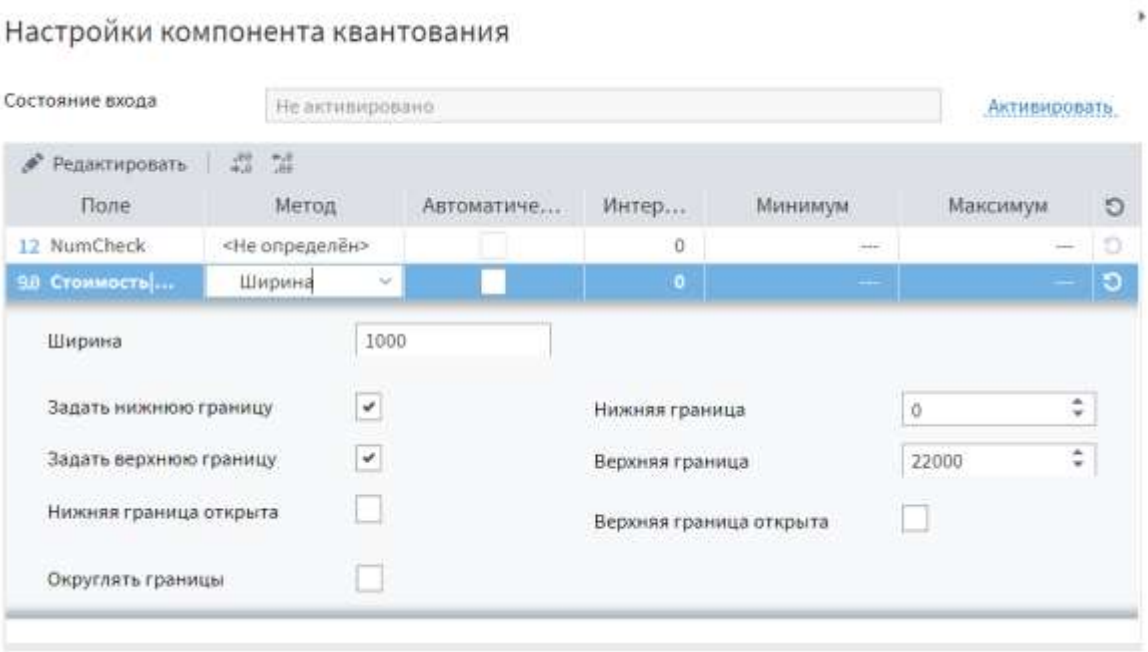
#	NumCheck	Стоимость|Сумма
1	248	3109,54
2	567	2326,52
3	791	6589,2
4	946	4421,76
5	998	189
6	1252	9251,1
7	2596	4523,68
8	3252	913
9	3298	526
10	3461	565
11	3502	189
12	3877	4824,54
13	4330	3602,48
14	4941	4820,3
15	4957	2827,3
16	5109	449
17	5188	3105,62
18	5300	4078,76
19	5623	5407,4
20	5669	5345,65
21	6009	189
22	6310	2966,46

#	NumCheck	Стоимость|Сум... 1
1	4945	21188,8
2	10001	16436,9
3	26801	16399,85
4	21045	15829,85
5	28998	14942,55
6	29663	14631,9
7	18500	14582,5
8	5479	14380,15
9	6822	14139,8
10	16329	13994,45
11	28178	13891,85
12	8437	13870,95
13	15612	13850,05
14	17813	13818,7
15	13930	13664,8
16	20573	13621,1
17	15191	13445,35
18	27723	13294,3
19	17296	13224
20	26384	13213,55
21	21274	13167,95
22	25850	13140,4



Далее будем применять обработку **Квантование**. Процесс квантования или дискретизации – это замена точных «непрерывных» значений номерами или метками интервалов, в которые эти значения попадают.

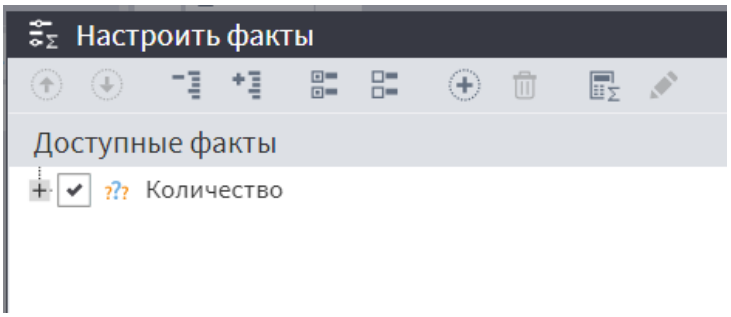
Выберем самый простой тип квантования – по ширине интервала. Пусть все интервалы имеют равную ширину 1000, а также зададим нижнюю и верхнюю границы данных:



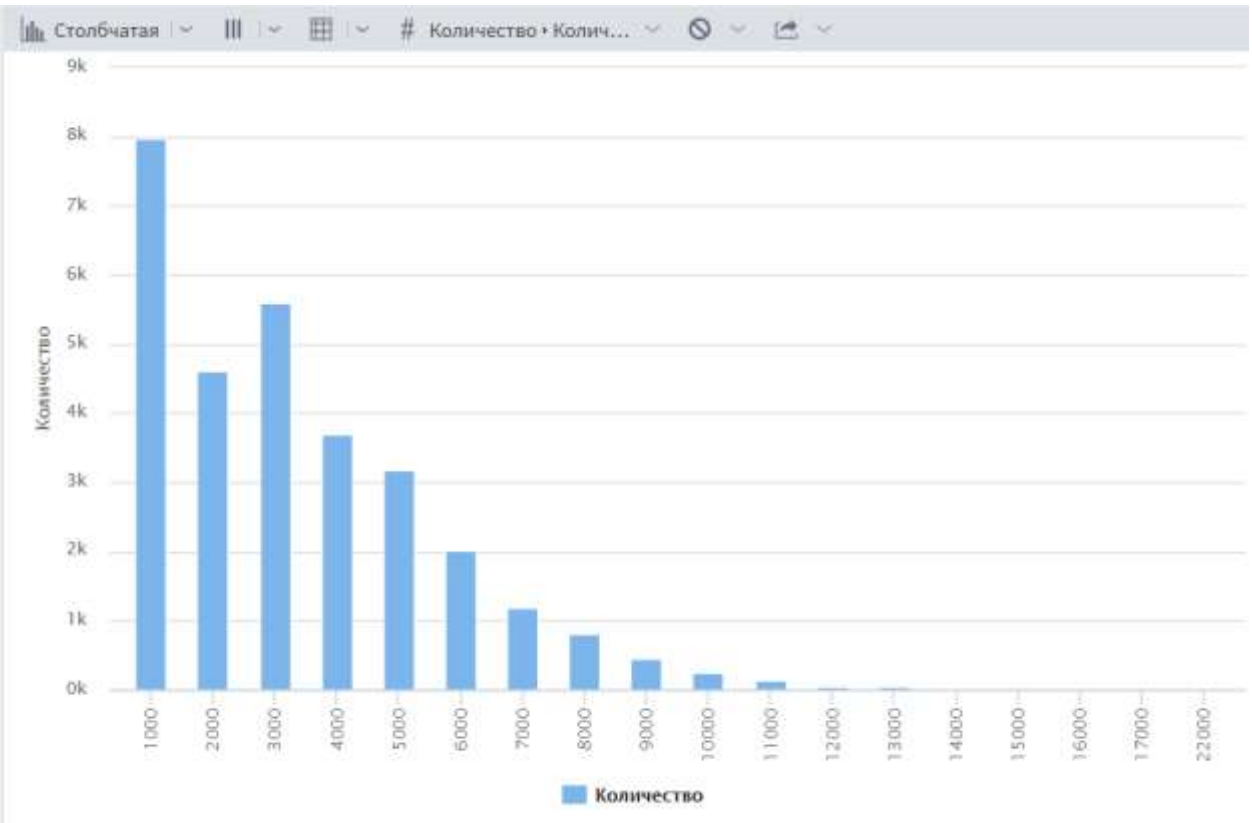
Далее создадим визуализатор Куб, у которого заданы только строки, а измерением является **Верхняя граница** интервалов:

Стоимость|Сумма Интервальный Верхняя граница	Количество
1000	7 967
2000	4 622
3000	5 603
4000	3 695
5000	3 175

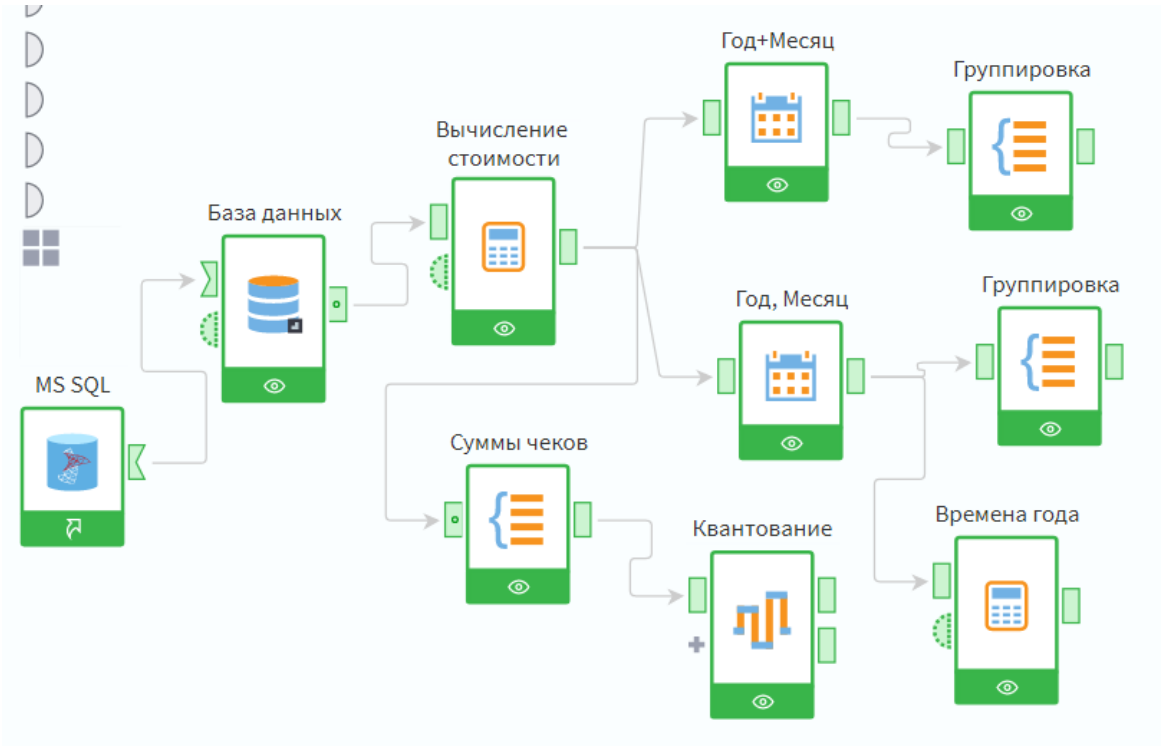
В качестве фактов используется стандартное **Количество** (оно соответствует функции COUNT в SQL):



Таким образом, мы подсчитали, сколько чеков попало в каждый интервал, шаг интервала 1000 р. Построим диаграмму:



Наш сценарий со всеми рассмотренными примерами из текущего параграфа, со всеми узлами и связями окончательно имеет такой вид:



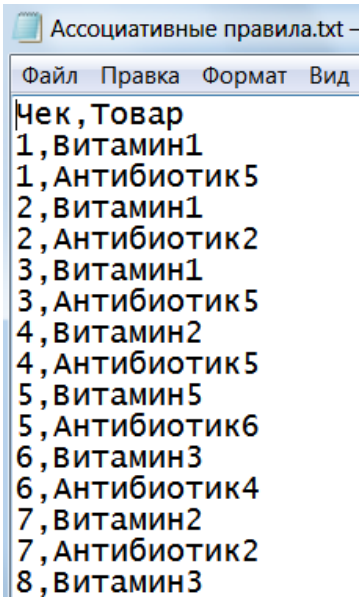
Задание 6. Создайте в **Loginom** подключение к SQL server. Разработайте сценарий (или несколько сценариев), в котором примените к вашим данным разнообразные преобразования и визуализаторы. Используйте, как минимум, преобразования: **Калькулятор, Преобразование даты, Фильтр, Группировка, Квантование** и визуализаторы: **Таблица, Куб, Диаграмма, Детализация**. (15 баллов).

Этап 7. Анализ данных в Loginom

Рассмотрим два примера решения задач анализа данных в среде **Loginom** для нашего хранилища **Аптеки**.

1. Ассоциативные правила, или задача анализа потребительских корзин

Задача анализа потребительских корзин состоит в поиске взаимосвязанных товаров, которые часто попадают вместе в корзины покупателей. При генерации данных вы, скорее всего, не предусматривали таких закономерностей. Поэтому для решения этой задачи вы можете вручную создать небольшой текстовый файл (50 - 100 строк) следующей структуры:

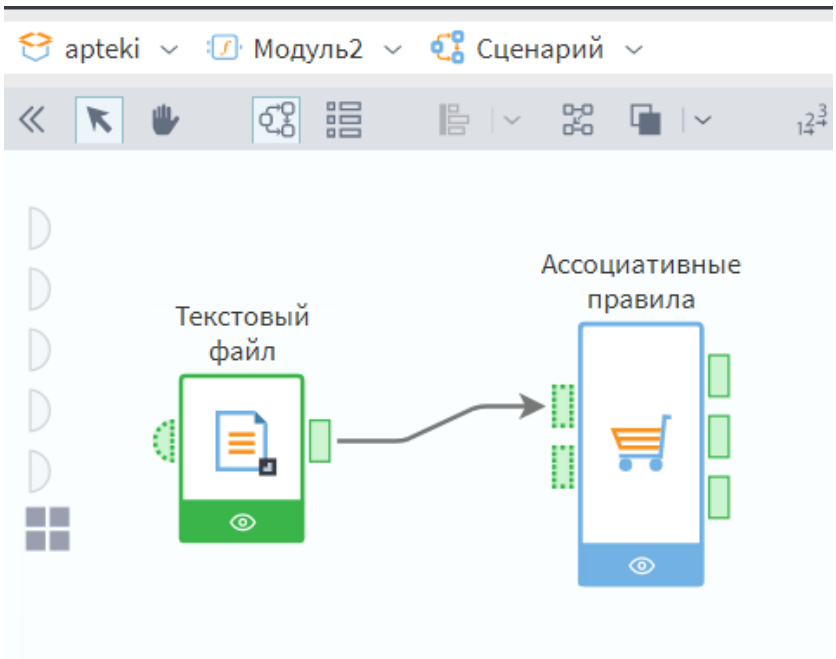


В этом файле нужно предусмотреть товары, которые часто попадают в один и тот же чек. Здесь, например, Витамин1 и Антибиотик5 часто встречаются вместе, в одном чеке.

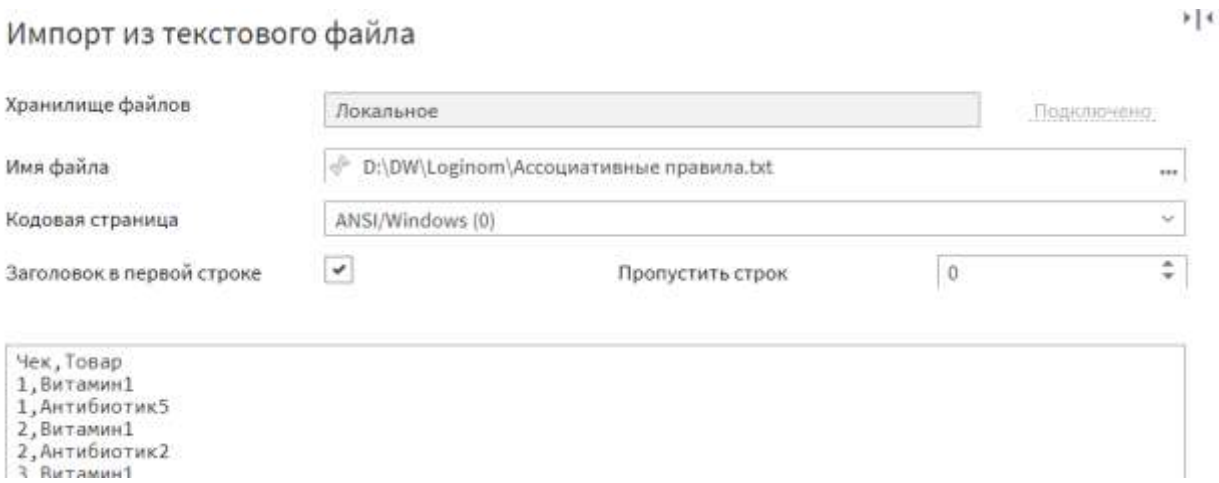
Чек является эквивалентом покупательской корзины, в анализе этот объект называют также **Транзакцией.**

Создадим новый модуль в нашем проекте (пакете).

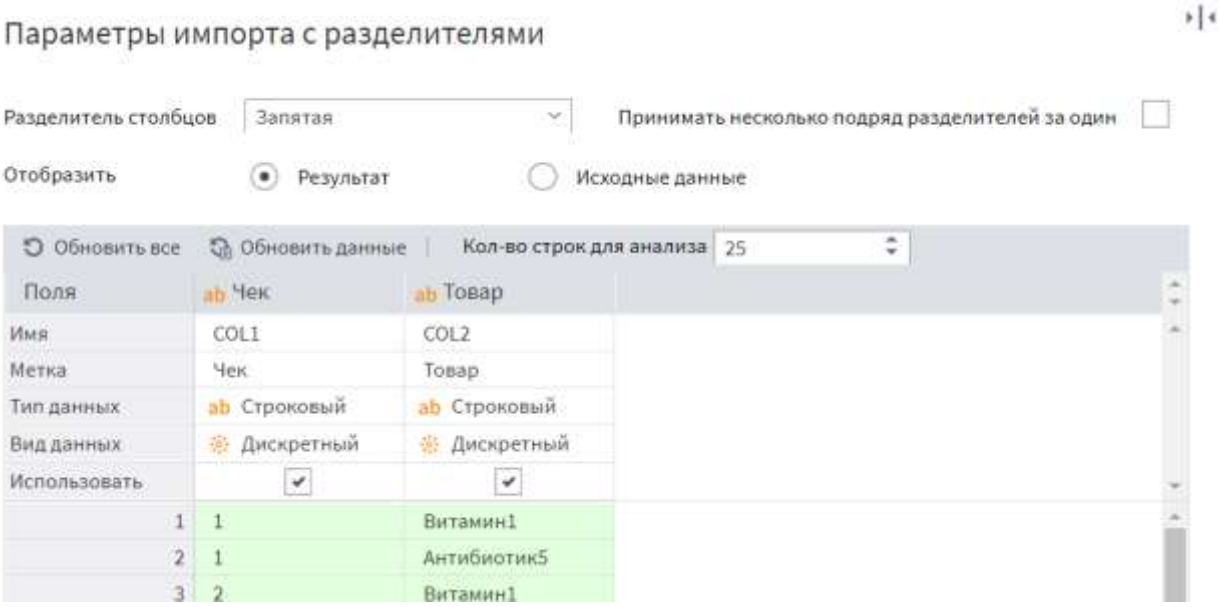
В качестве источника данных создадим узел **Текстовый** файл и настроим его:



Выберем нужный файл:



Основные параметры импорта можно оставить по умолчанию. Обязательно нужно только указать разделитель столбцов: **Запятая**, а для столбца **Чек** задать **Строковый** тип данных (по умолчанию это числовой тип).

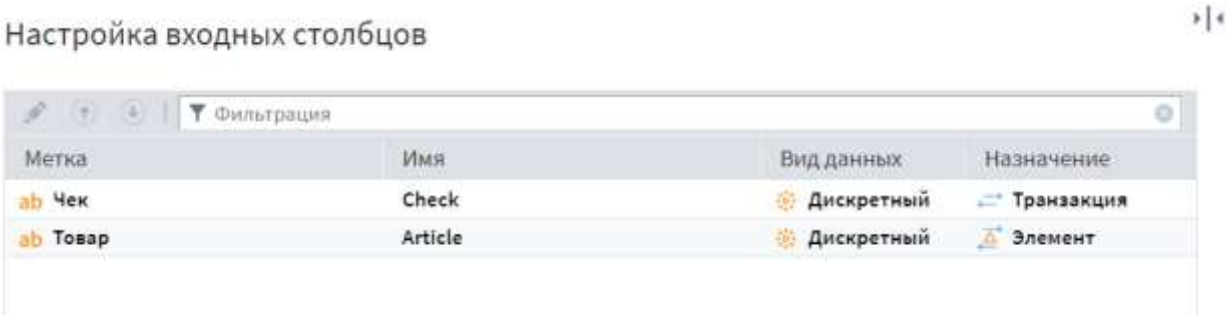


Создадим визуализатор **Таблица**, посмотрим, что получилось:

#	Чек	Товар
1	1	Витамин1
2	1	Антибиотик5
3	2	Витамин1
4	2	Антибиотик2
5	3	Витамин1
6	3	Антибиотик5
7	4	Витамин2
8	4	Антибиотик5
9	5	Витамин5

Теперь можно создавать узел **Ассоциативные правила** из секции **Data Mining.**

Настроим входные столбцы. Чек – это **Транзакция**, корзина с покупками, а Товар – **Элемент** транзакции, составная часть корзины.



Для поиска ассоциативных правил нужно обязательно задавать такие параметры, как **Поддержка** и **Достоверность**.

Поддержка (Support) - частота совместного наблюдения событий, процент корзин, содержащих конкретный набор товаров в данном ассоциативном правиле. *Это аналог совместной вероятности событий.*

Достоверность (Confidence) – это процент корзин из числа корзин, содержащих первый товар, в которых также находится и второй товар. *Это аналог условной вероятности событий.*

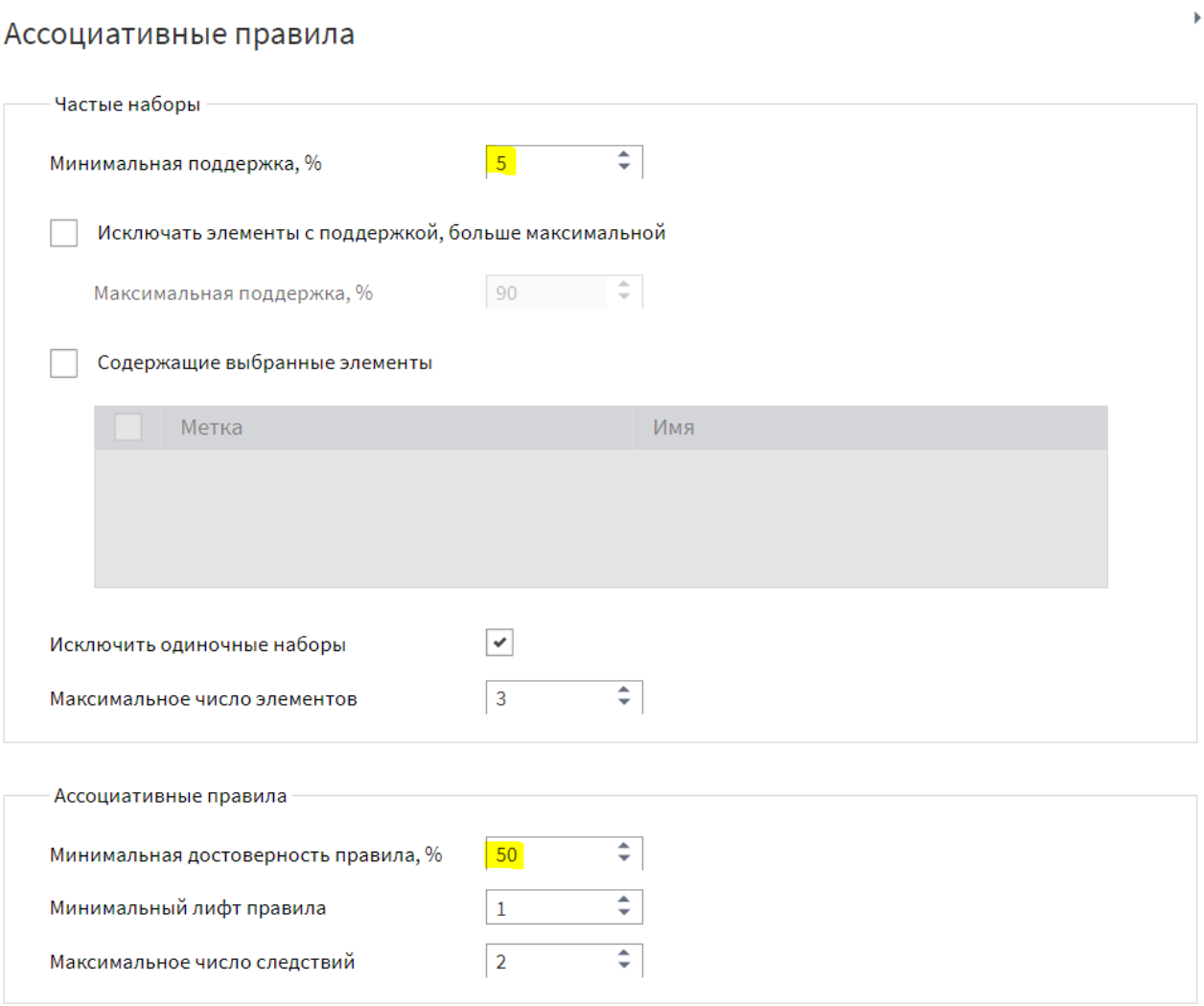
Таким образом, если мы задаем Поддержку = 5, значит, нас интересуют правила (наборы товаров), которые присутствуют **не менее чем в 5%** от всех корзин.

Таким образом, если мы задаем Достоверность = 50, значит, нас интересуют правила (наборы товаров), в которых присутствие второго товара (следствия) наблюдается **не менее чем в 50%** корзин, содержащих первый товар (условие).

Кроме того, можно отметить флажок «**Исключить одиночные наборы**» - тем самым мы исключаем из рассмотрения корзины, состоящие только из одного товара. При этом показатели поддержки и достоверности найденных правил будут выше, чем при наличии таких корзин.

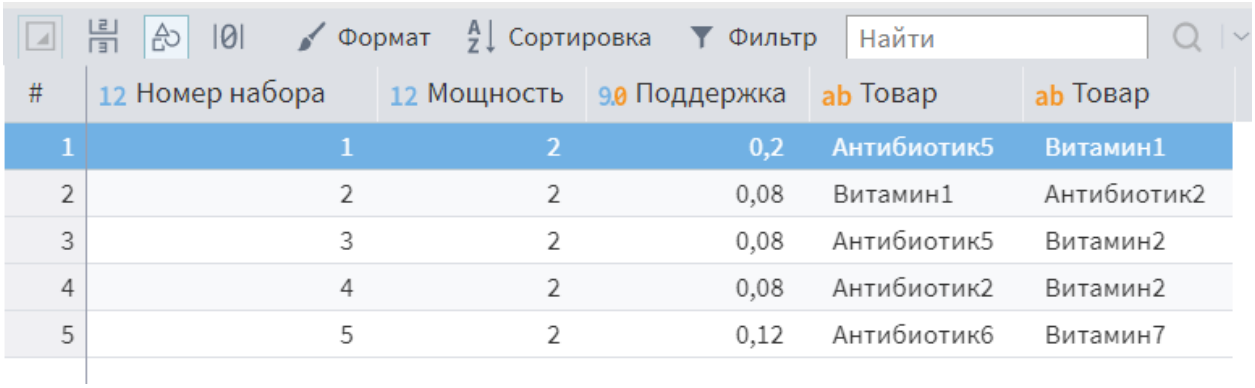
Показатель **Лифт** служит для оценки, насколько элементы найденного правила отличаются от независимых величин (для независимых величин Лифт = 1). *Вычисляется этот показатель как частота совместного*

наступления событий, деленная на произведение частот отдельных событий.



После задания параметров активируем (или переобучим) узел через контекстное меню, а потом создадим визуализаторы.

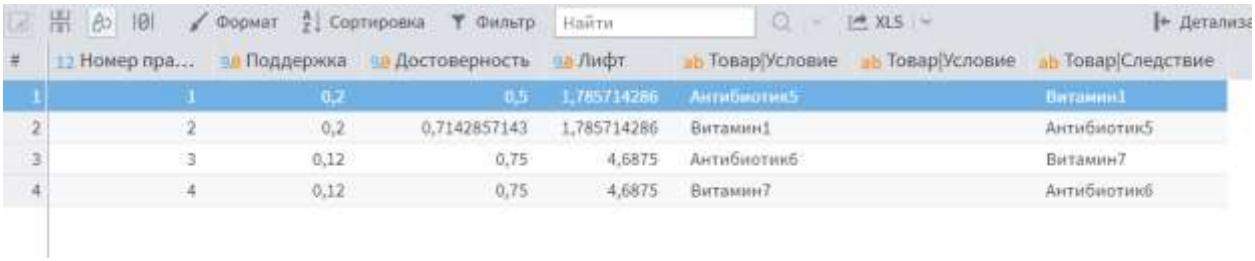
Визуализатор **Популярные наборы** показывает наборы, частота наблюдения которых (поддержка) более 5%. Таких наборов нашлось 5:



#	12 Номер набора	12 Мощность	9.0 Поддержка	ab Товар	ab Товар
1	1	2	0,2	Антибиотик5	Витамин1
2	2	2	0,08	Витамин1	Антибиотик2
3	3	2	0,08	Антибиотик5	Витамин2
4	4	2	0,08	Антибиотик2	Витамин2
5	5	2	0,12	Антибиотик6	Витамин7

Самый популярный набор (Антибиотик5, Витамин1) встречается в 20% корзин.

Теперь создадим визуализатор **Ассоциативные правила**:



#	Номер пра…	Поддержка	Достоверность	Лифт	Товар|Условие	Товар|Условие	Товар|Следствие
1	1	0,2	0,5	1,785714286	Антибиотик5		Витамин1
2	2	0,2	0,7142857143	1,785714286	Витамин1		Антибиотик5
3	3	0,12	0,75	4,6875	Антибиотик6		Витамин7
4	4	0,12	0,75	4,6875	Витамин7		Антибиотик6

Нашлись 4 правила, соответствующие заданным параметрам модели. Обратите внимание, что четвертое и второе правила являются «обратной» версией третьего и первого, соответственно. При этом показатели поддержки и лифта у «прямого» и «обратного» правила, разумеется, совпадают, а достоверность может различаться.

Из всех корзин, содержащих Антибиотик5, 50% содержат также и Витамин1.

Наоборот, из всех корзин, содержащих Витамин1, более 71% содержат также и Антибиотик5.

Дополнительный вопрос: какой из этих двух товаров более популярен?

Почему в список ассоциативных правил попали только первый и пятый наборы из списка популярных наборов?

Для иллюстрации показателя Лифт приведем следующий пример. Пусть имеются 2 товара: Товар1 и Товар2, очень популярные, но совершенно независимые друг от друга. Пусть 50% всех корзин содержит Товар1, 50% всех корзин содержит Товар2. Они совершенно независимы, поэтому вместе они попадают в 25% корзин (0.5 * 0.5 *100). Таким образом, Поддержка для этого набора = 0.25, Достоверность = 0.5, но зависимости товаров нет, Лифт = 1. Как видим, показатель Лифт также должен применяться для детального анализа ассоциативных правил.

2. ABC-анализ и XYZ-анализ

В аналитической отчетности очень полезными часто оказываются ABC и XYZ анализ – распределение объектов (например товаров, клиентов, поставщиков) по доходности и стабильности продаж. Если в качестве объекта анализа взять товары, а в качестве критерия анализа - объем продаж, то в результате получим:

- при ABC-анализе продаж – наиболее и наименее пользующиеся спросом товары;
- при XYZ-анализе продаж – группировку товарных позиций в зависимости от стабильности продаж (по количеству проданных единиц)

Результатом ABC-анализа является группировка объектов по трем категориям:

- категория A – наиболее ценные объекты, сумма долей с накопительным итогом которых составляет первые 75 % общей суммы;
- категория B – промежуточные объекты, следующие за группой A, сумма долей с накопительным итогом которых составляет от 75 до 90 % общей суммы;
- категория C – оставшиеся наименее ценные объекты, сумма долей с накопительным итогом которых составляет от 90 до 100 % обшей суммы.

Алгоритм ABC-анализа включает следующие шаги [1].

1. Определить **объект анализа**: клиент, поставщик, товарная группа, товарная позиция. Для ABC-анализа продаж это, как правило, товарная позиция.

2. Определить **параметр анализа**: объем продаж, доход, количество проданных единиц, частота покупок товара, количество заказов, средний товарный запас и т. д, Для ABC-анализа продаж это, как правило, объем продаж в денежном выражении.

3, Определить **период**, за который будет проводиться ABC-анализ, и подготовить агрегированные данные по продажам за этот период. Так, если период равен кварталу, то нужно для каждой товарной позиции получить сумму квартальных продаж.

4. Отсортировать объекты анализа в порядке убывания значения параметра.

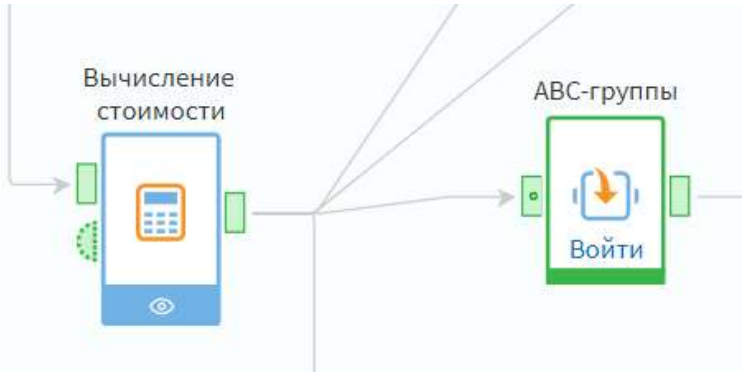
5. Чтобы определить принадлежность выбранного объекта к группам A, B и C, необходимо:

- рассчитать долю параметра от общей суммы параметров выбранных объектов;

- рассчитать эту долю с накопительным итогом;
- присвоить значения групп выбранным объектам.

Итак, будем анализировать объем продаж товаров за последние 3 месяца. За основу возьмем узел **Вычисление стоимости**, в котором уже рассчитана стоимость продажи в качестве отдельного столбца.

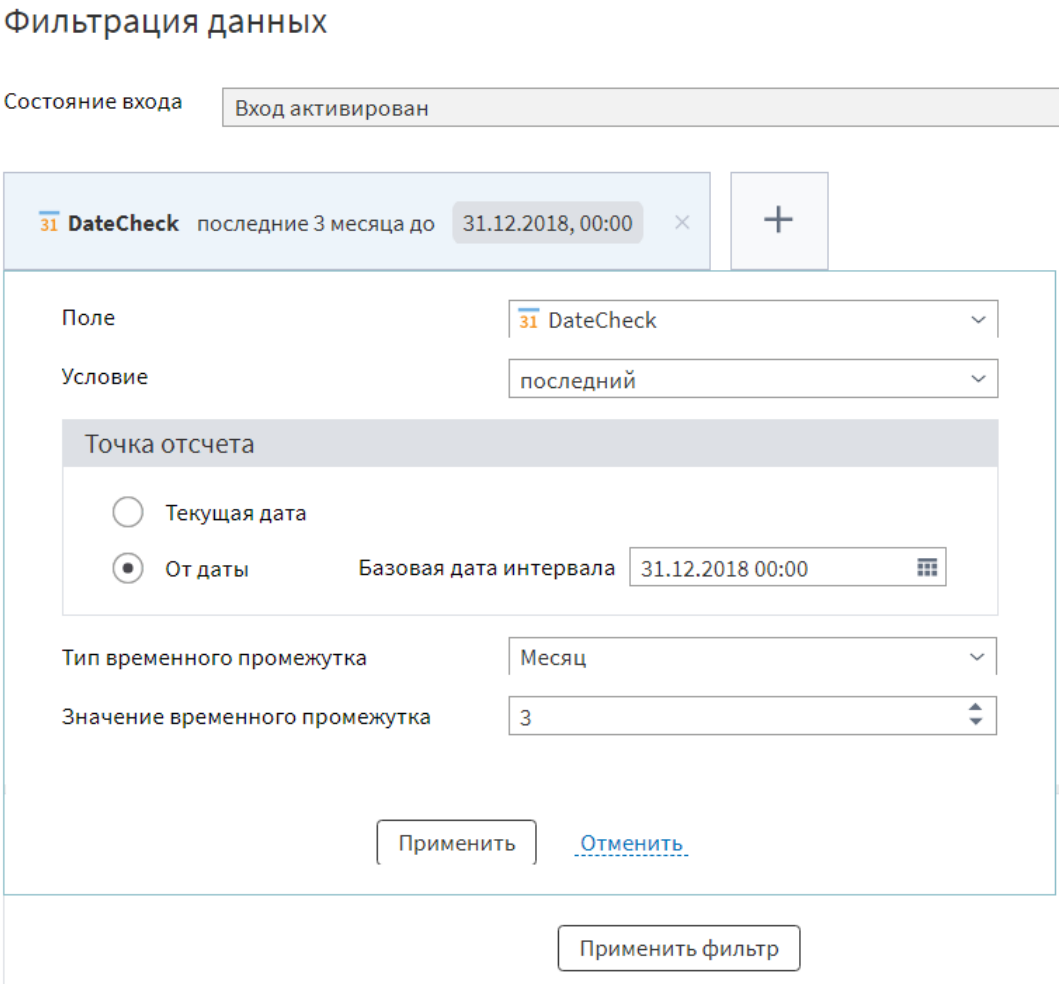
АВС-анализ оформим в виде подмодели. Создадим новый узел **Подмодель** из секции **Управление,** переименуем его:



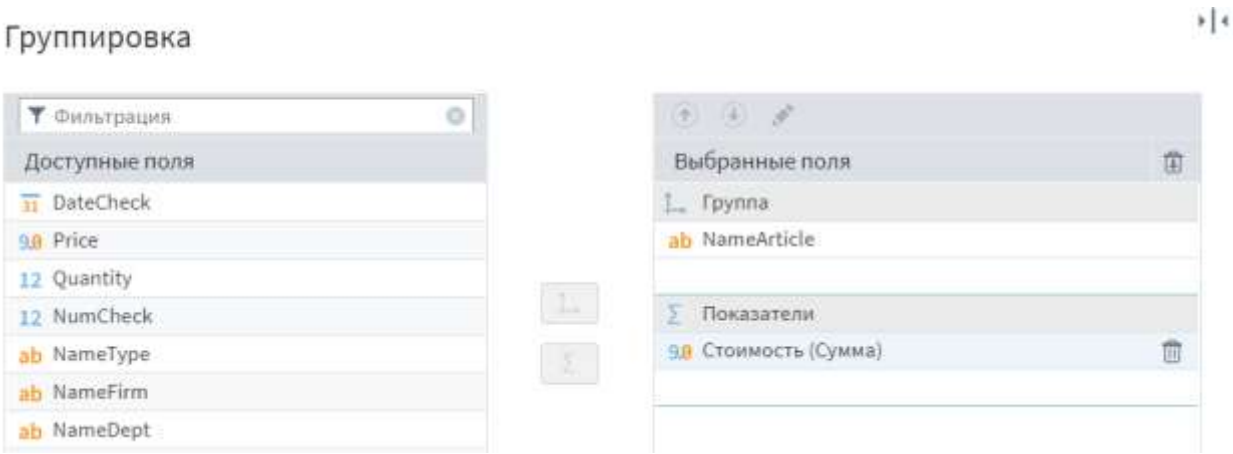
По умолчанию у подмодели нет никаких входных и выходных портов данных. Зайдем в настройку подмодели и создадим входной и выходной порты данных типа Таблица:

Теперь подмодель можно встраивать в общий сценарий. Свяжем выходной порт узла **Вычисление стоимости** и входной порт подмодели. Зайдем внутрь подмодели. Все последующие действия выполняем внутри подмодели.

1 шаг. Создадим узел типа **Фильтр строк** - отфильтруем данные за последние 3 месяца:

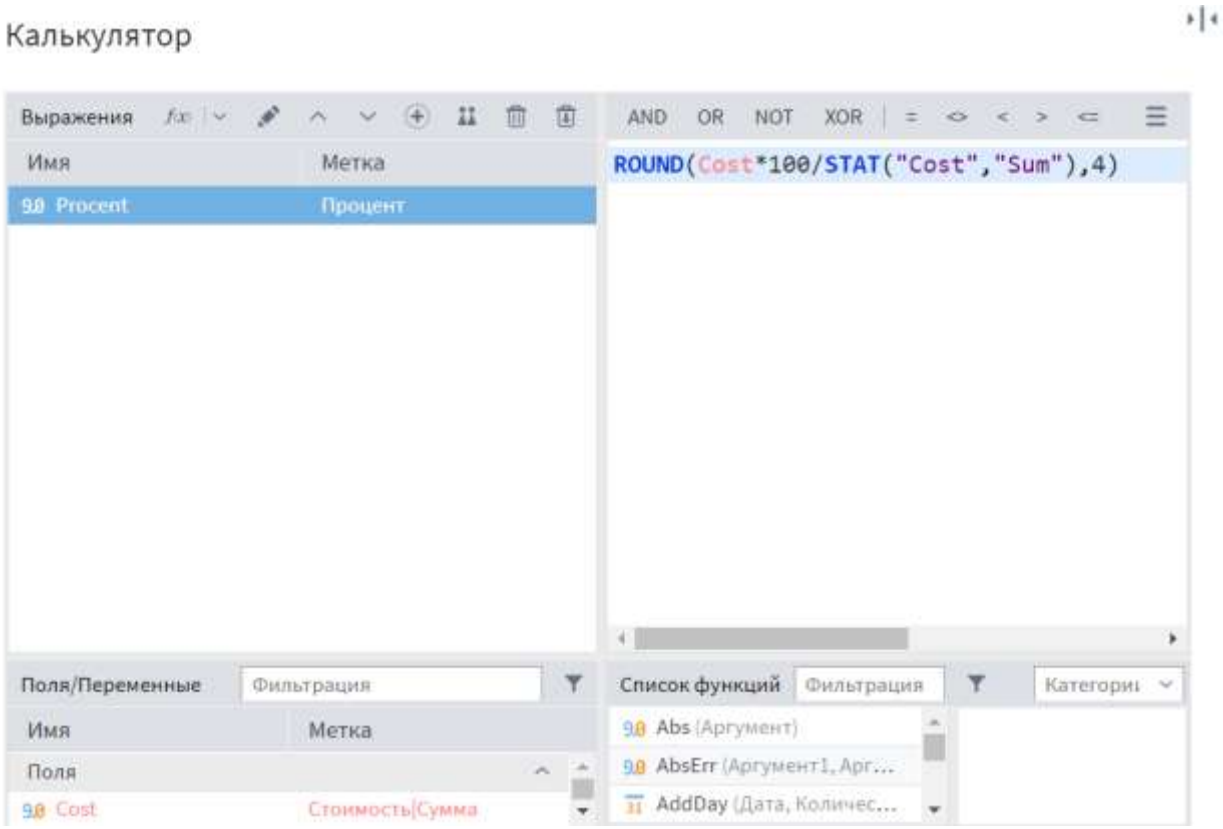


2 шаг. Создадим узел **Группировка** по нужному нам столбцу (по товарам) :



3 шаг. Создадим узел **Сортировка** по убыванию суммы.

4 шаг. Создадим узел **Калькулятор**, который подсчитывает долю текущего товара от общей суммы (в процентах).



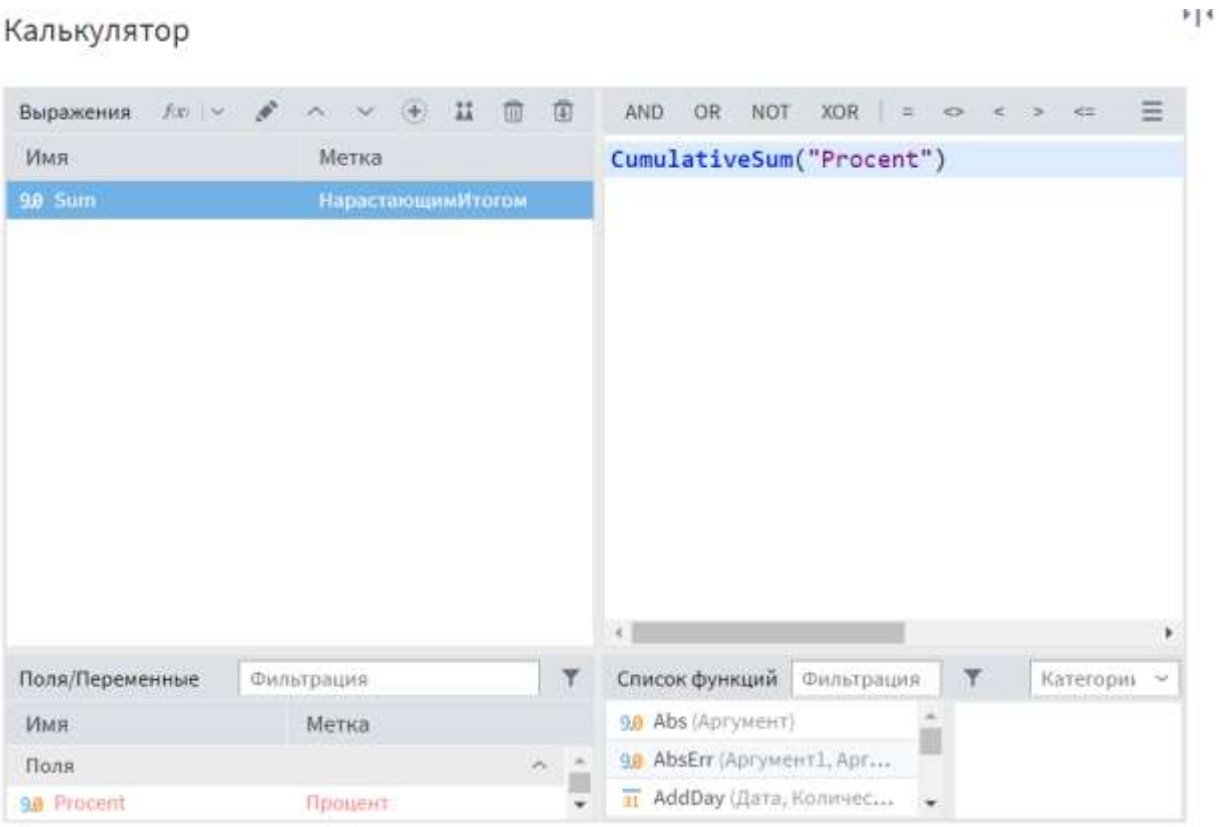
Здесь функция **ROUND**(*число, количество_знаков*) округляет *число* до *количества_знаков* после запятой.

Функция **STAT** вычисляет итоговое значение для всего столбца, имя которого задано в первом аргументе функции. Второй аргумент означает тип итогового значения – количество, сумму, среднее и т.п.

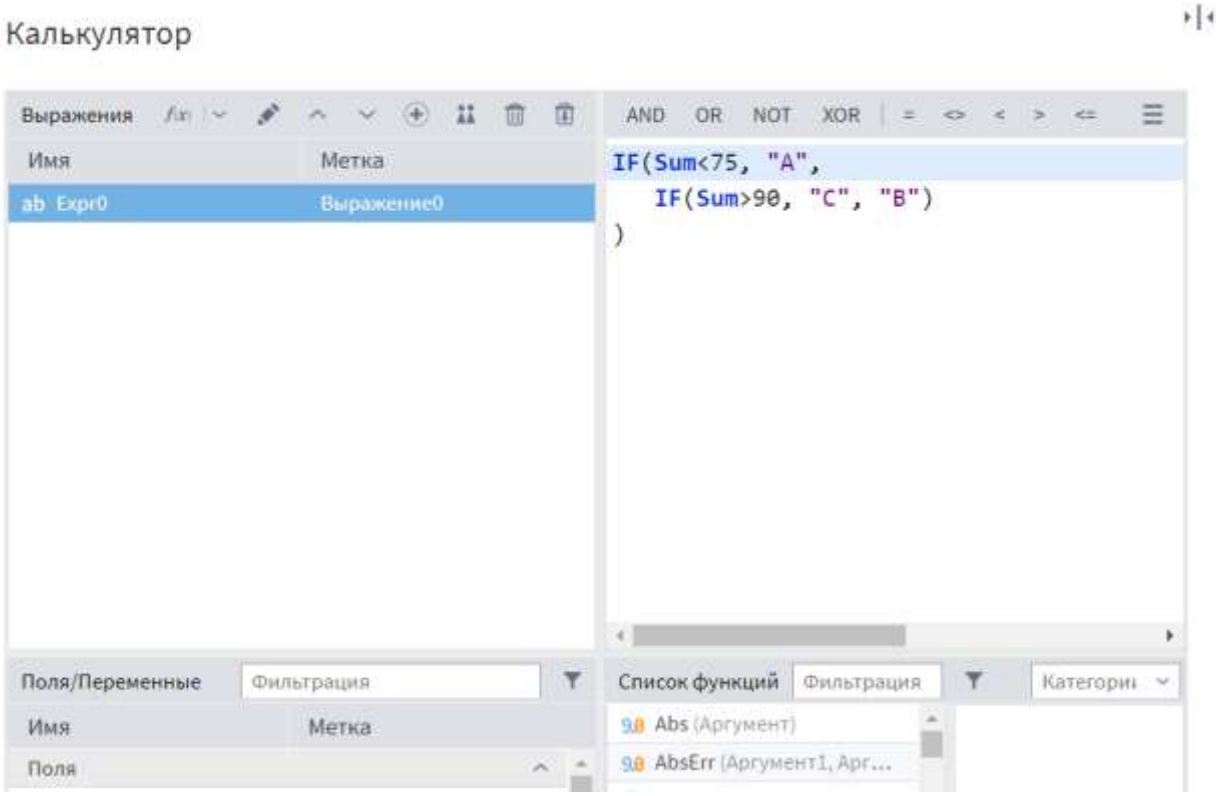
Посмотрим, что получилось, через визуализатор **Таблица**:

#	ab NameArticle	9.0 Стоимость|Сумма	9.0 Процент
1	Товар 300	208028,5	2,3609
2	Товар 254	195892,8	2,2232
3	Товар 258	192089,04	2,18
4	Товар 328	187342,47	2,1262
5	Товар 277	179376,3	2,0358
6	Товар 283	174100,92	1,9759
7	Товар 295	172715,4	1,9602
8	Товар 348	165739,2	1,881
9	Товар 278	156241,87	1,7732
10	Товар 264	155254,86	1,762
11	Товар 289	151049,52	1,7143

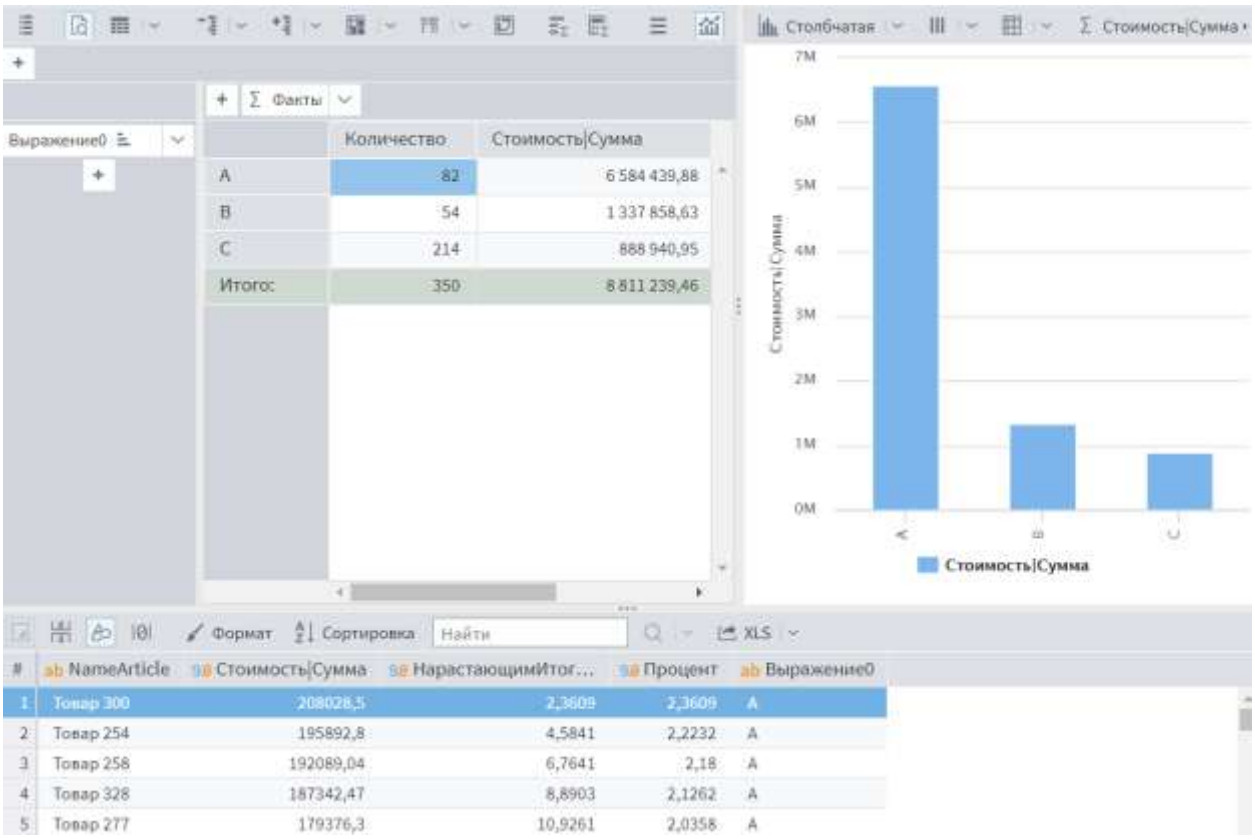
5 шаг. Создадим узел «**Калькулятор**», который подсчитывает долю текущего товара от общей суммы нарастающим итогом с помощью функции **CumulativeSum**.



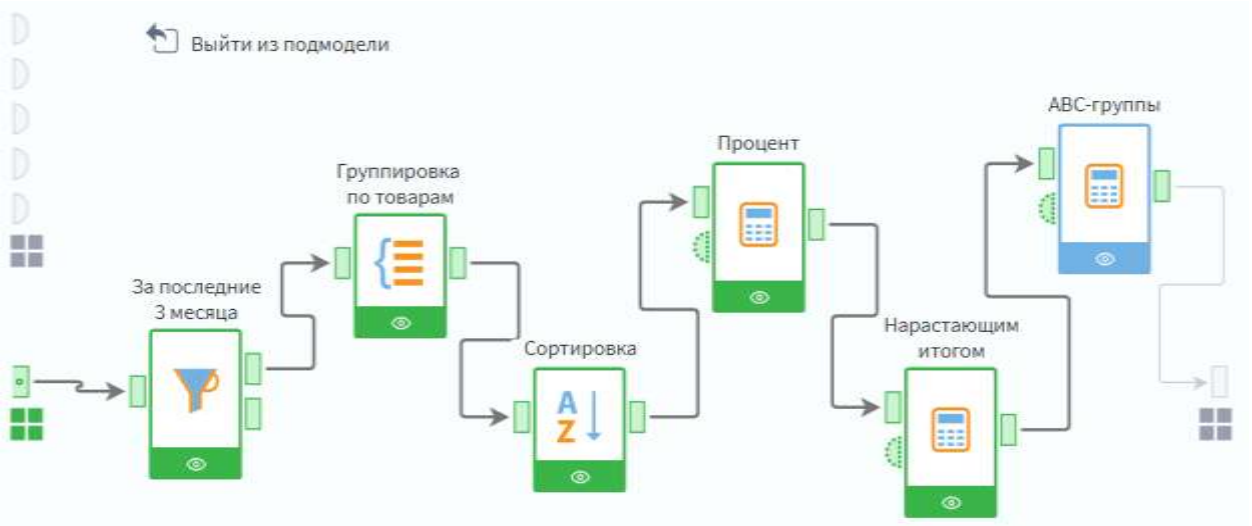
6 шаг. Наконец, создадим узел «**Калькулятор**», который распределяет товары по группам. Поскольку новый столбец будет содержать строковые значения, обязательно нужно задать ему **строковый тип** (по умолчанию в Калькуляторе новый столбец получает числовой тип).



Далее выводим полученные данные в виде куба, у которого строками являются АВС-группы. Параллельно выводим диаграмму, показывающую размеры групп, и таблицу с детальной информацией. Теперь мы можем оценить, какие товары приносят нам больше выручки из общего объема, а какие, наоборот, являются невыгодными с этой точки зрения. Точно так же можно оценивать и другие объекты – покупателей, отделы, фирмы и т.п.



Вот так в целом выглядит наша подмодель, узлы переименованы в соответствии с конкретными этапами алгоритма.



Результат XYZ-анализа - группировка ресурсов по трем категориям на основе предварительно рассчитанного коэффициента вариации (отклонения от среднего):

- категория X - объекты, коэффициент вариации которых не превышает 10 %. Характеризуются стабильной величиной потребления;
- категория Y - объекты, коэффициент вариации которых составляет 10-25 %. Потребность в них обычно зависит от известных тенденций, например от сезонных колебаний;
- категория Z - объекты, коэффициент вариации которых превышает 25 %. Потребление ресурсов нерегулярное, какие-либо тенденции отсутствуют.

Коэффициент вариации рассчитывается по следующей формуле:

$$v = \frac{\sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}}}{\bar{x}} \cdot 100\%,$$

где x_i - значение параметра по оцениваемому объекту за i-й период;

$\bar{x}$ - среднее значение параметра по оцениваемому объекту анализа;

n - число периодов.

Прибегать к данному методу анализа имеет смысл, если количество анализируемых периодов больше трех. Чем больше количество периодов, тем более показательными будут результаты. При этом сам период должен быть не меньше, чем горизонт планирования, принятый в компании, иначе велика вероятность того, что все товары попадут в категорию Z.

Алгоритм XYZ анализа состоит из нескольких шагов.

1, Определить **объект анализа**: клиент, поставщик, товарная группа, товарная позиция. Для XYZ-анализа продаж это, как правило, товарная позиция.

2, Определить **параметр анализа**: объем продаж, доход, количество проданных единиц, частота покупок товара, количество заказов, средний товарный запас и т.д. Для XYZ-анализа продаж это, как правило, количество проданных единиц товара или частота покупок.

3, Определить период, за который будет проводиться XYZ-анализ, и подготовить агрегированные данные по продажам за этот период.

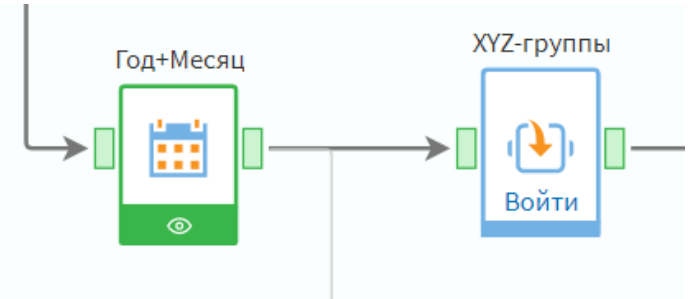
4. Рассчитать коэффициент вариации.

5. Отсортировать объекты анализа по возрастанию значения коэффициента вариации (не обязательно).

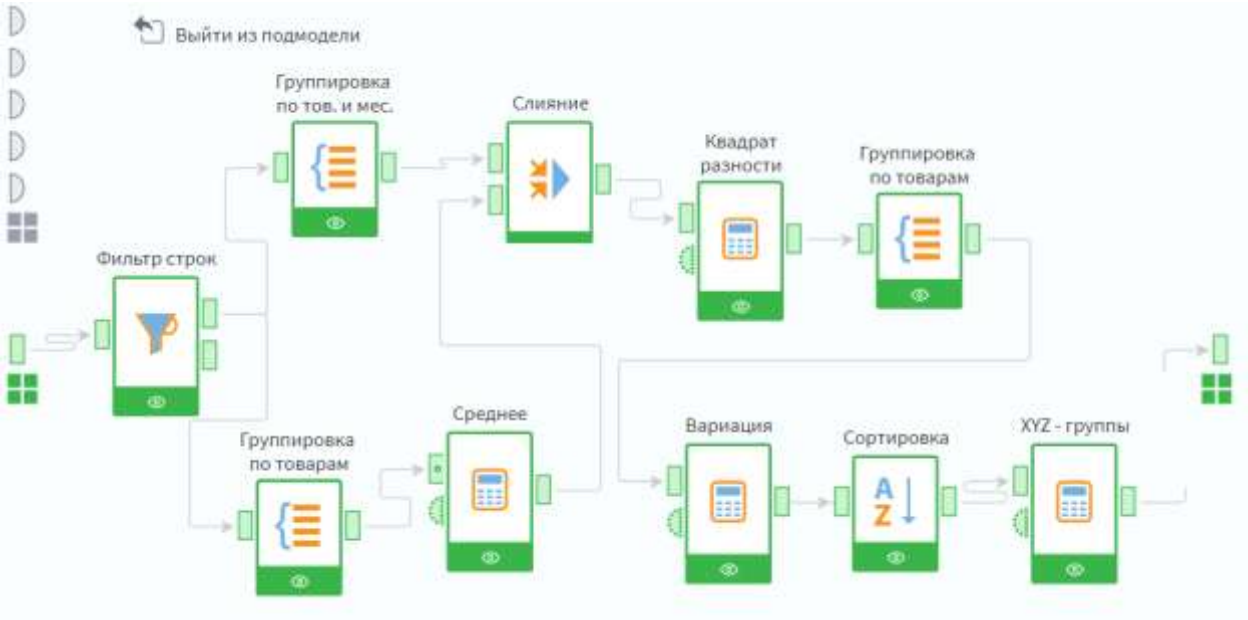
6. Определить группы X, Y и Z.

Итак, будем анализировать продажи товаров в количественном выражении за последние 3 месяца. В качестве предыдущего узла возьмем узел типа **Преобразование даты,** в котором уже выделены месяц + год продаж.

XYZ-анализ также оформим в виде подмодели.



В результате у нас получится такой сценарий:

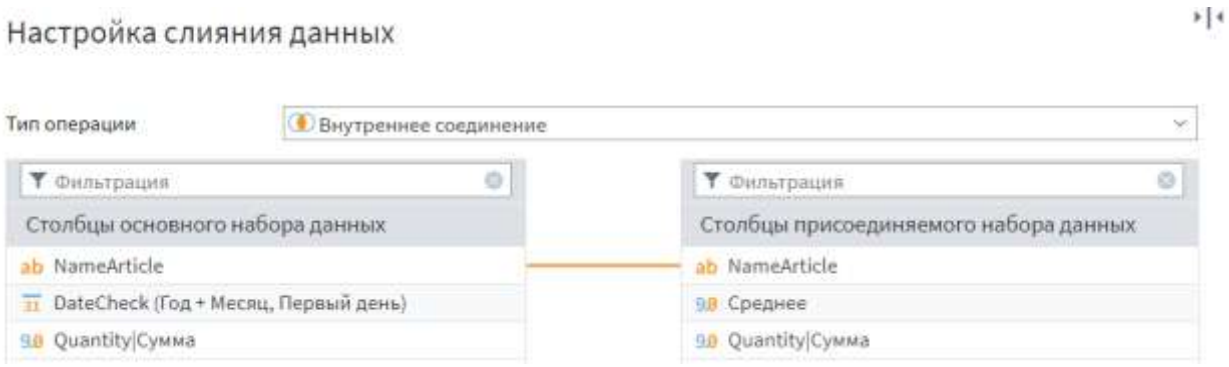


Несколько замечаний по поводу этого сценария. В узле **Фильтр строк** берем данные за последние 3 месяца, как и в АВС-анализе.

Далее отдельно выполняем **Группировку по товарам и месяцам**, и отдельно **Группировку по товарам** для дальнейшего вычисления среднего. В качестве фактов в обеих группировках берем количество продаж в штуках.

Среднее вычисляется как объем продаж по товару, деленный на 3 (так как анализируем данные за 3 месяца).

Далее выполняем **Слияние** итогов по товарам и месяцам и средних значений по товарам. В качестве **Типа** операции выберем **Внутреннее соединение** (соответствует JOIN в SQL), укажем условие связи между соединяемыми таблицами (название товара).



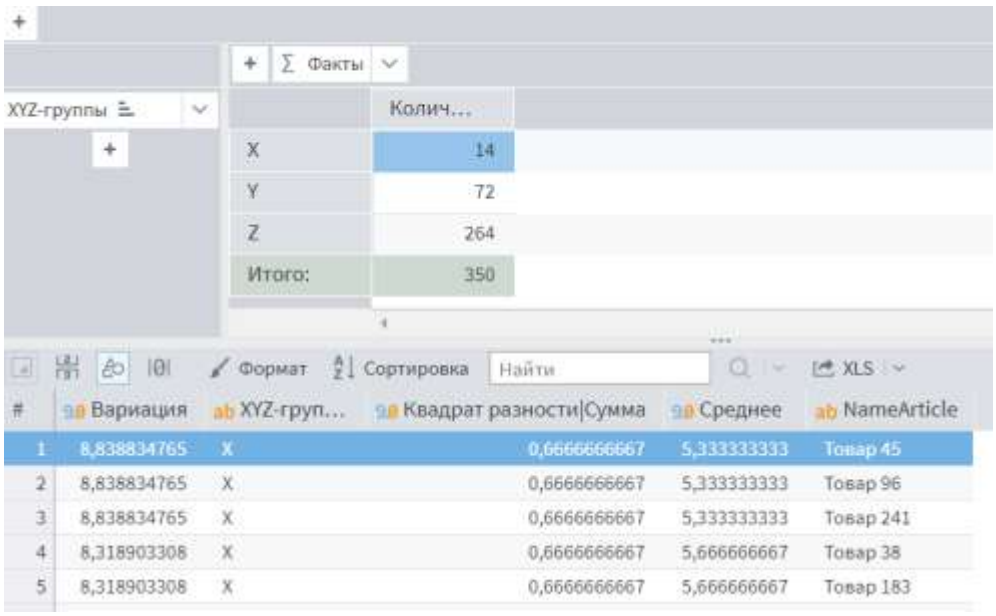
Следующий узел **Калькулятор** вычисляет квадраты разностей по каждому товару и месяцу.

Далее производим **Группировку** по товарам, суммируем квадраты разностей по каждому товару.

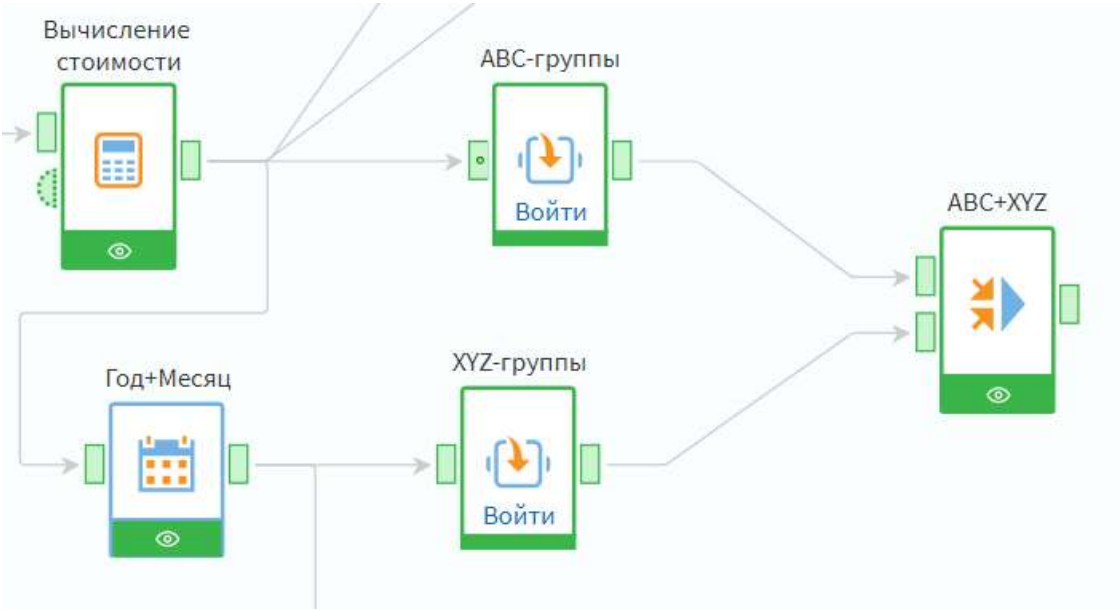
Следующий **Калькулятор** вычисляет коэффициент вариации.

Сортировку производить не обязательно, но так удобнее представлять данные.

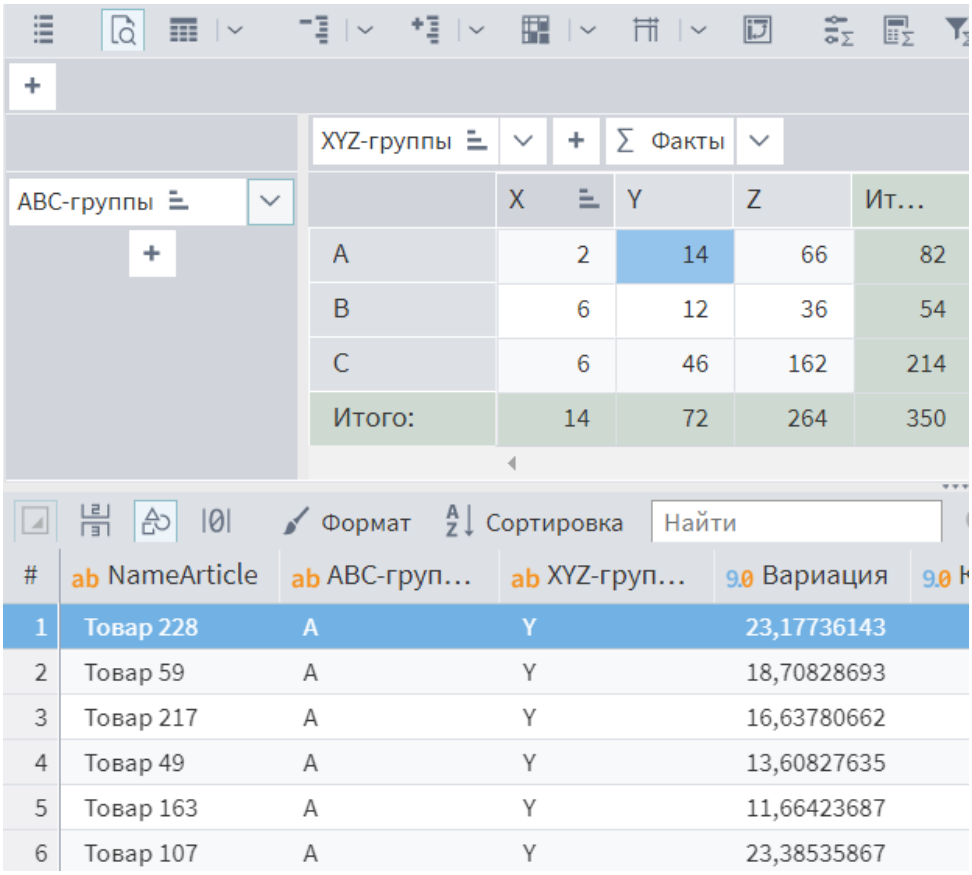
Наконец, последний **Калькулятор** вычисляет группы XYZ по всем товарам. Получим следующий результат в виде куба:



Теперь соединим итоги ABC-и XYZ-анализа (узел Слияние):



Получим следующий куб с детализацией:



XYZ-группы | Факты

ABC-группы

	X	Y	Z	Ит…
A	2	14	66	82
B	6	12	36	54
C	6	46	162	214
Итого:	14	72	264	350

Формат | Сортировка | Найти

#	NameArticle	ABC-груп…	XYZ-груп…	Вариация	К
1	Товар 228	A	Y	23,17736143	
2	Товар 59	A	Y	18,70828693	
3	Товар 217	A	Y	16,63780662	
4	Товар 49	A	Y	13,60827635	
5	Товар 163	A	Y	11,66423687	
6	Товар 107	A	Y	23,38535867	

Объекты, попавшие в разные группы, имеют следующие характеристики:

Критерии	X	Y	Z
A	AX (высокая значимость товара в ассортименте, высокая степень надежности прогноза вследствие стабильности продаж)	AY (высокая значимость товара в ассортименте, средняя степень надежности прогноза вследствие нестабильности продаж)	AZ (высокая значимость товара в ассортименте, низкая степень надежности прогноза вследствие случайного потребления)
B	BX (средняя значимость товара в ассортименте, высокая степень надежности прогноза вследствие стабильности продаж)	BY (средняя значимость товара в ассортименте, средняя степень надежности прогноза вследствие нестабильности продаж)	BZ (средняя значимость товара в ассортименте, низкая степень надежности прогноза вследствие случайного характера продаж)
C	CX (низкая значимость товара в ассортименте, высокая степень надежности прогноза вследствие стабильности продаж)	CY (низкая значимость товара в ассортименте, средняя степень надежности прогноза вследствие нестабильности продаж)	CZ (низкая значимость товара в ассортименте, низкая степень надежности прогноза вследствие случайного характера продаж)

Задание 7. Примените все рассмотренные в данном параграфе методы анализа к своим данным.

Итого 15 баллов.

Работа в программе Deductor Studio

Этап 6. Выгрузка данных в текстовые файлы

Учебная версия программы DEDUCTOR, к сожалению, не позволяет напрямую подключаться к SQL Server (у промышленной версии такого ограничения нет). Поэтому придется обмениваться данными через текстовый файл.

У вас есть следующие варианты выполнения данного задания. Самый простой способ состоит в том, что вы выгружаете **одну** многомерную таблицу со всей информацией, созданную на 3 этапе. Тогда в программе DEDUCTOR вы не будете создавать **настоящее** хранилище данных (и не получите за это баллы в количестве 7 шт.), а в качестве хранилища будет выступать этот текстовый файл. Все возможности работы с OLAP-кубами данных, диаграммами, анализ данных для такого источника данных будут доступны.

Более сложный вариант, предполагающий создание **настоящего** хранилища данных в DEDUCTOR (в формате СУБД FireBird), требует создания отдельных текстовых файлов для таблицы фактов (Продажи) и для всех измерений (Типы, Товары, Аптеки и пр.) Рассмотрим обе эти возможности.

Примеры выгрузки:

1) Выгрузим подготовленную таблицу с детальной информацией о продажах и льготной реализации в текстовый файл. Для этого снова обратимся к программе массового копирования **bcp (bulk copy procedure)**.

Вспомним, что эта команда запускается из командной строки Windows (Пуск – Поиск – cmd – Enter). В нашем случае формат команды следующий:

```
bcp drugstores.dbo.fullRealization out fullRealization.txt -T -w
-S HOME-PC -C 1251 -t; -r\n
```

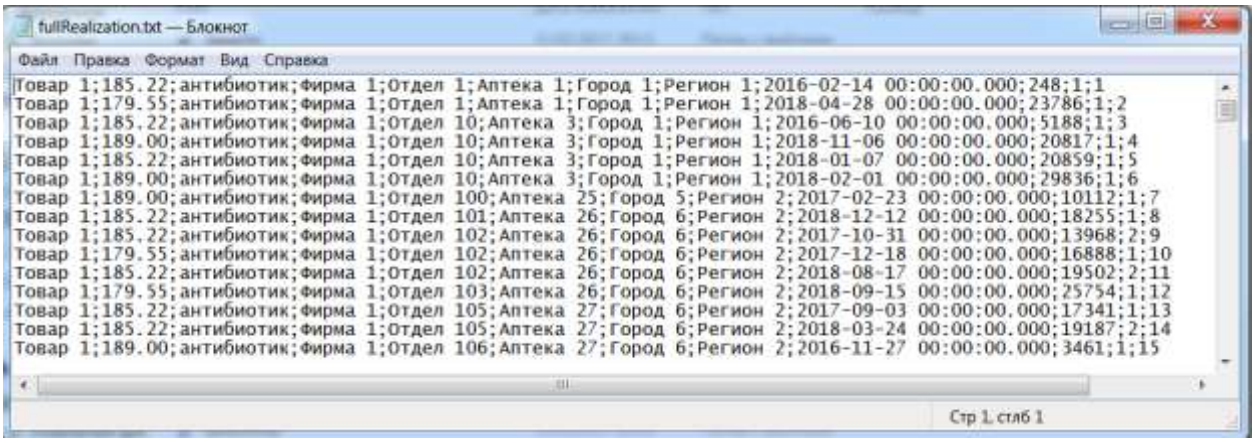
Рассмотрим параметры этой команды, которые не встречались нам раньше.

`out` – направление потока данных: из SQL server наружу;

`-w` – выполняет операцию массового копирования, используя символы Юникода. При использовании этого параметра не запрашивается тип данных каждого поля, для хранения данных используется тип nchar, отсутствуют префиксы, в качестве разделителя полей используется символ табуляции **\t**, а в качестве признака конца строки — символ новой строки **\n**.

-r\n – задается разделитель конца строки – символ ‘\n’.

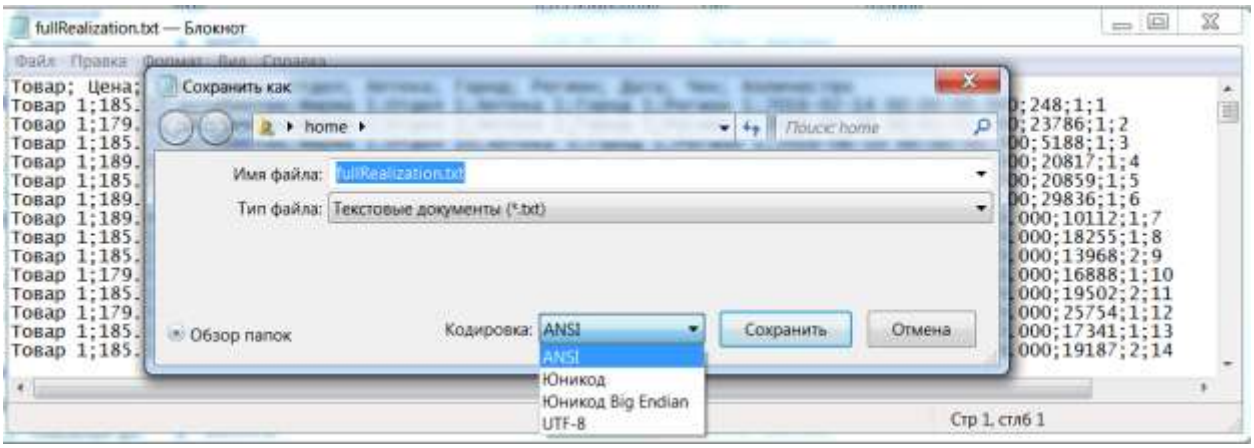
Получим файл с таким содержимым:



Для того чтобы формат нашего файла соответствовал программе DEDUCTOR, откроем полученный файл в «Блокноте», добавим в качестве первой строки заголовки столбцов:

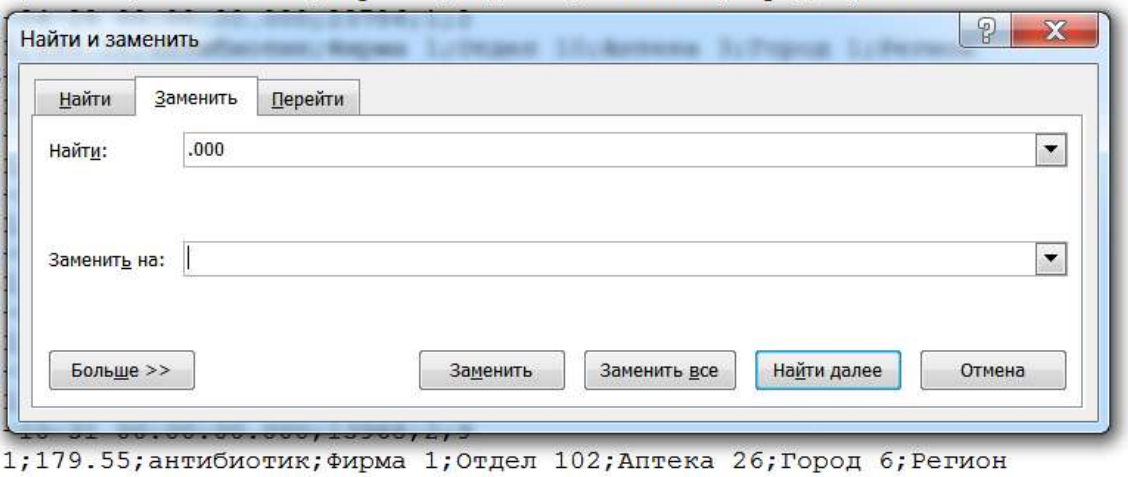
Товар; Цена; Тип; Фирма; Отдел; Аптека; Город; Регион; Дата; Чек; Количество

а затем сохраним его в формате ANSI:



Примечание: в выгруженном файле имеются даты, которые в своем составе содержат миллисекунды в виде .000. Эти миллисекунды в дальнейшем вызовут **большие** проблемы при загрузке в Deductor Studio, поэтому от них следует избавиться. Самый простой способ – открыть файл в текстовом редакторе и произвести автоматическую замену, например, заменить строку «.000» на строку «». Программа «Блокнот» для этой операции может быть слишком медленной, лучше использовать, например, MS Word.

```
Товар 1;185.22;антибиотик;Фирма 1;Отдел 1;Аптека 1;Город 1;Регион
1;2016-02-14 00:00:00.000;248;1;1
Товар 1;179.55;антибиотик;Фирма 1;Отдел 1;Аптека 1;Город 1;Регион
1;2018
Товар
1;2016
Товар
1;2018
Товар
1;2018
Товар
1;2018
Товар
2;2017
Товар
2;2018
Товар
2;2017
Товар 1;179.55;антибиотик;Фирма 1;Отдел 102;Аптека 26;Город 6;Регион
2;2017-12-18 00:00:00.000;16888;1;10
```



2) Теперь рассмотрим выгрузку данных **в несколько** текстовых файлов – отдельно для таблицы фактов и отдельно для всех измерений. (Выгружаем только продажи, льготную реализацию для упрощения работы брать не будем.)

Типы лекарств:

```
bcp drugstores.dbo.Type out types.txt -T -w -S HOME-PC -C 1251 -t; -
r\n
```

Фирмы-производители:

```
bcp drugstores.dbo.Firm out firms.txt -T -w -S HOME-PC -C 1251 -t; -
r\n
```

Чеки:

```
bcp drugstores.dbo.Bill out bills.txt -T -w -S HOME-PC -C 1251 -t; -
r\n
```

Продажи:

```
bcp drugstores.dbo.Sale out sales.txt -T -w -S HOME-PC -C 1251 -t; -
r\n
```

Отделы:

```
bcp drugstores.dbo.Department out departments.txt -T -w -S HOME-PC -C
1251 -t; -r\n
```

Из таблицы «Аптеки» не будем брать телефон, для этого придется написать SELECT и указать, что тип операции – не “out”, а “queryout”:

```
bcp "SELECT NumDrug, NameDrug, Town, Region FROM
drugstores.dbo.drugstore" queryout drugstores.txt -T -w -S HOME-PC -C
1251 -t; -r\n
```

Из таблицы «Товары» тоже возьмем только некоторые столбцы, для этого придется написать SELECT и указать, что тип операции – не “out”, а “queryout”:

```
bcp "SELECT NumArticle, NameArticle, NumType, NumFirm FROM
drugstores.dbo.article" queryout articles.txt -T -w -S HOME-PC -C 1251
-t; -r\n
```

Если не указывать полный путь к файлам, то по умолчанию все файлы создаются в **корневом** каталоге **текущего** пользователя.

Задание 6а. Выгрузите подготовленные данные в один или несколько текстовых файлов. Задайте в этих файлах заголовки столбцов. Не забудьте сохранить файлы в формате ANSI. В тех файлах, где используется тип дата+время, избавьтесь от миллисекунд.

Итого 5 баллов.

Программа **Deductor Studio** поставляется в нескольких вариантах. Для целей обучения имеется бесплатная версия **Deductor Studio Academic**. Основное ограничение этой версии заключается в том, что для хранилищ данных можно использовать только **СУБД FireBird** (входит в поставку программы), а загружать данные извне можно только из текстовых файлов.

Перед тем как начинать работу в **Deductor Studio**, рекомендуется прочитать руководство **«Базовые навыки работы в Deductor Studio.pdf».**

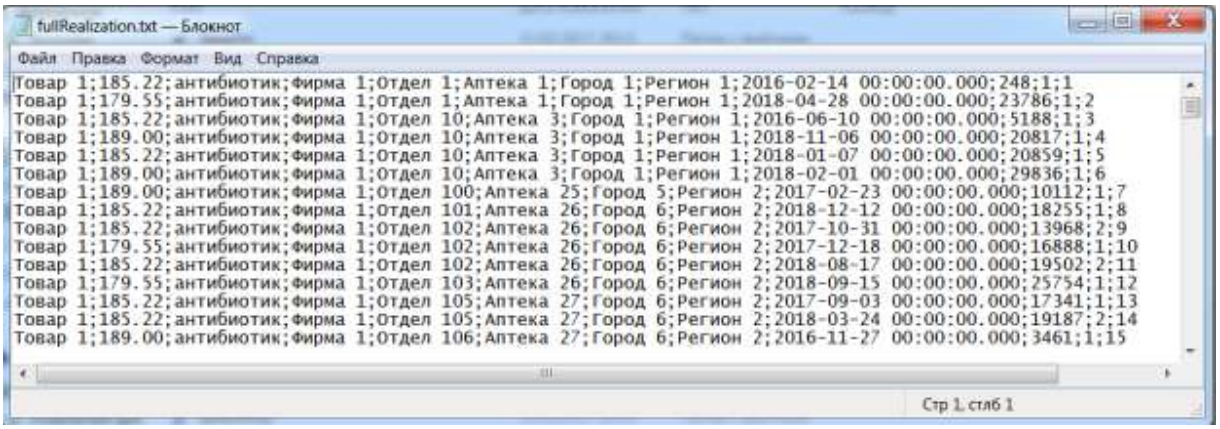
Этап 6б: Загрузка данных в Deductor Studio из текстовых файлов, создание хранилища данных в СУБД FireBird.

Как уже упоминалось в 5 задании, можно выбрать один из двух вариантов загрузки данных в **Deductor Studio**.

Первый вариант (простой) предполагает загрузку данных из **одного** большого текстового файла, созданного на предыдущем этапе и представляющего собой многомерную таблицу. Настоящее хранилище данных не создается. За этот вариант можно получить только 1 балл.

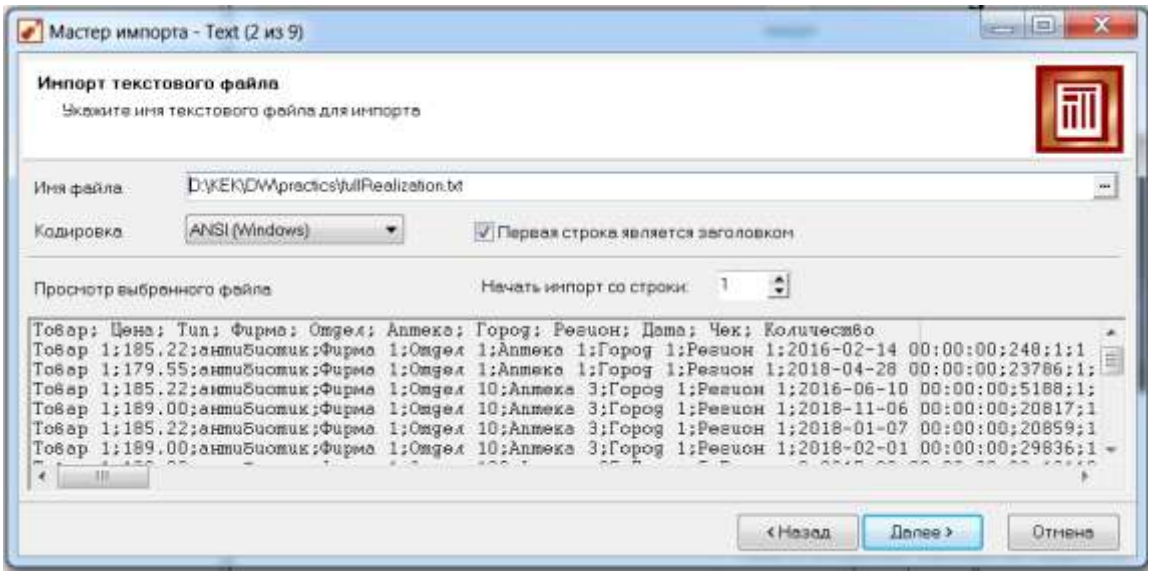
Второй вариант (сложный) предполагает создание настоящего хранилища данных и загрузку информации из **нескольких** текстовых файлов, соответствующих таблицам SQL server. Этот вариант оценивается максимум в 7 баллов.

Рассмотрим сначала **первый** вариант. У нас есть текстовый файл **fullRealisation.txt**, полученный на 5 этапе и представляющий собой общую информацию о продажах в аптеках и о бесплатной реализации лекарств льготникам. Этот файл содержит многомерные данные и выглядит примерно следующим образом:



Запускаем **Deductor Studio**, при этом автоматически создается пустой проект, в котором есть единственная строка *«Сценарии»*. Теперь можно щелкнуть правой кнопкой мыши по этой строке и из контекстного меню выбрать пункт «**Мастер импорта**», либо щелкнуть по кнопке «**Мастер импорта**»  и выбрать в качестве источника текстовые файлы.

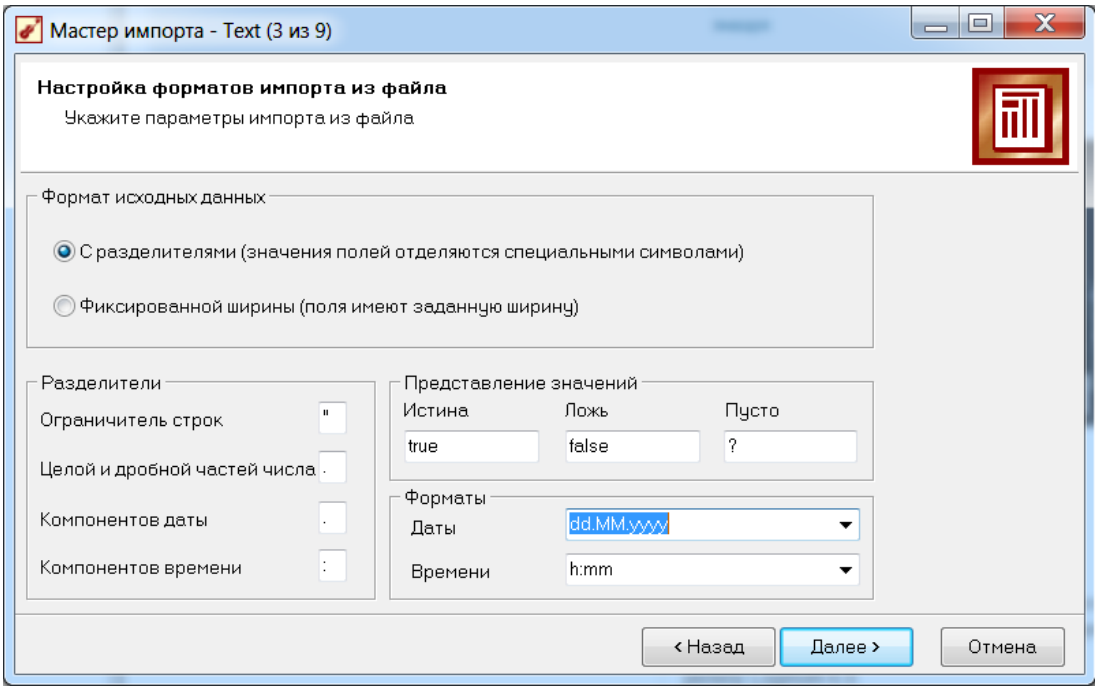
На 2-м шаге выбираем наш файл:



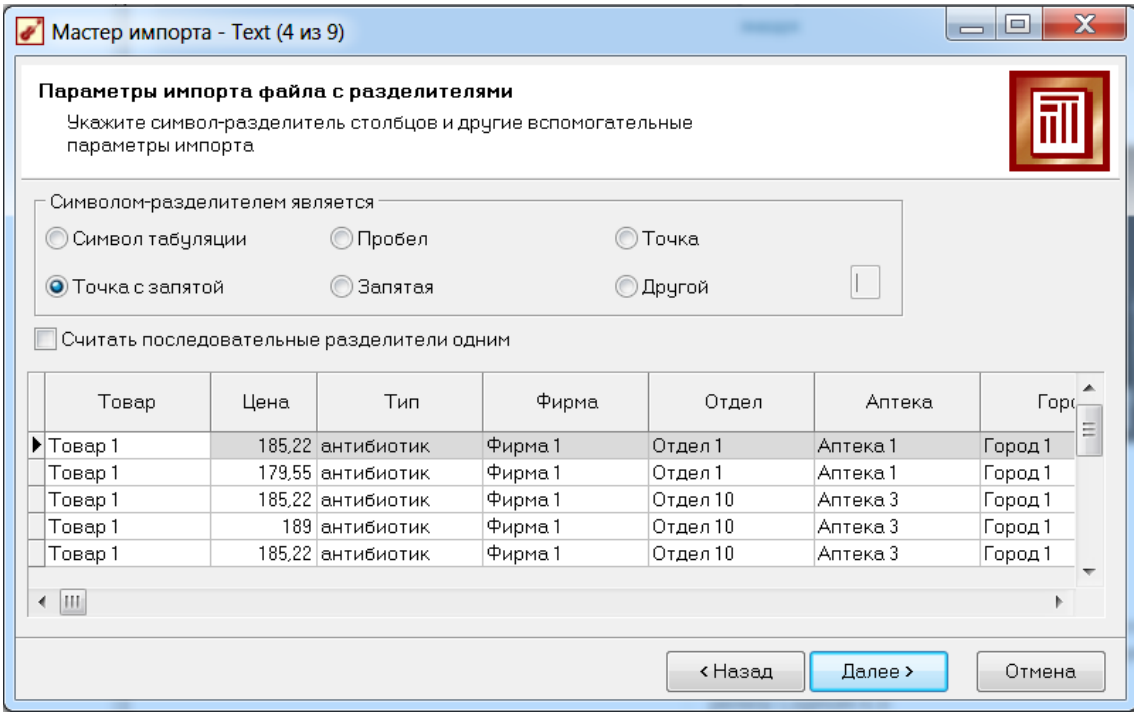
После выбора файла первые несколько строк отобразятся в окне.

Обратите внимание на флажок **«Первая строка является заголовком».**

На 3-м шаге можно установить некоторые параметры форматирования. В качестве разделителя целой и дробной части числа следует указать точку (по умолчанию – запятая). Формат даты, в принципе, можно не менять.

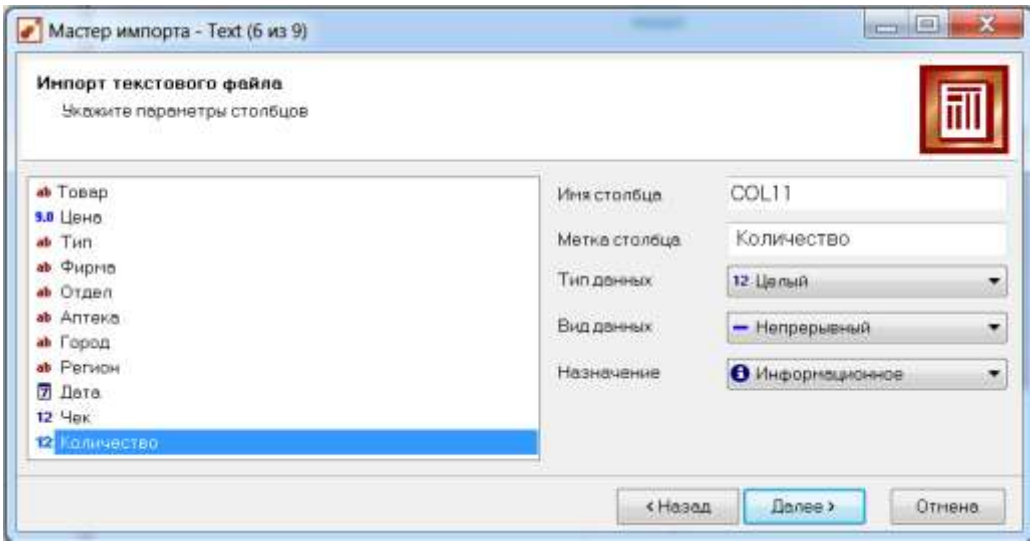


На 4-м шаге **обязательно** нужно указать верный формат разделителя полей (у нас – точка с запятой).

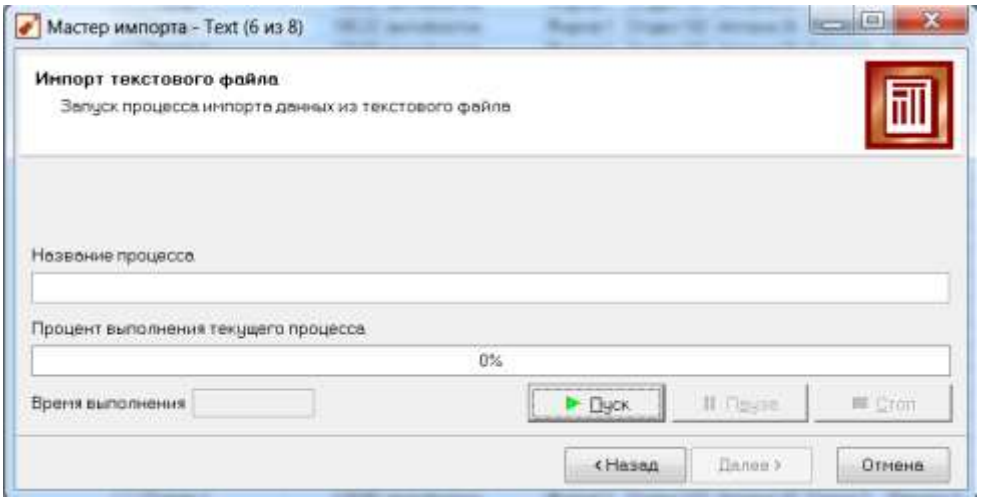


Данные из файла отображаются в таблице. Сразу можно увидеть, правильно ли воспринимаются числа (они выравниваются по правому краю) и даты.

На 6-м шаге *(5-го шага не было?)* следует задать типы столбцов таблицы. Здесь (в основном) можно оставить предлагаемые значения. Единственное замечание – все числовые типы по умолчанию предлагаются *вещественными*. Имеет смысл поменять тип на **«Целый»** и вид на **«Дискретный»** для столбцов, которые действительно содержат **только** целочисленные значения. В нашем случае это **Чек** и **Количество**. Назначение у всех столбцов пока «Информационное», отнести столбцы к измерениям и фактам можно будет потом.

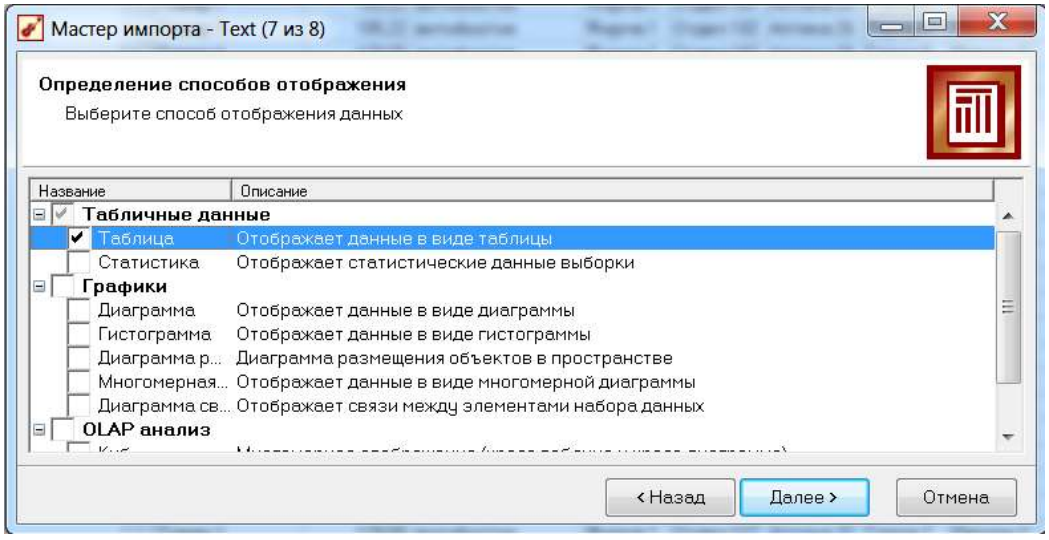


На следующем шаге нажимаем на кнопку «Пуск»:

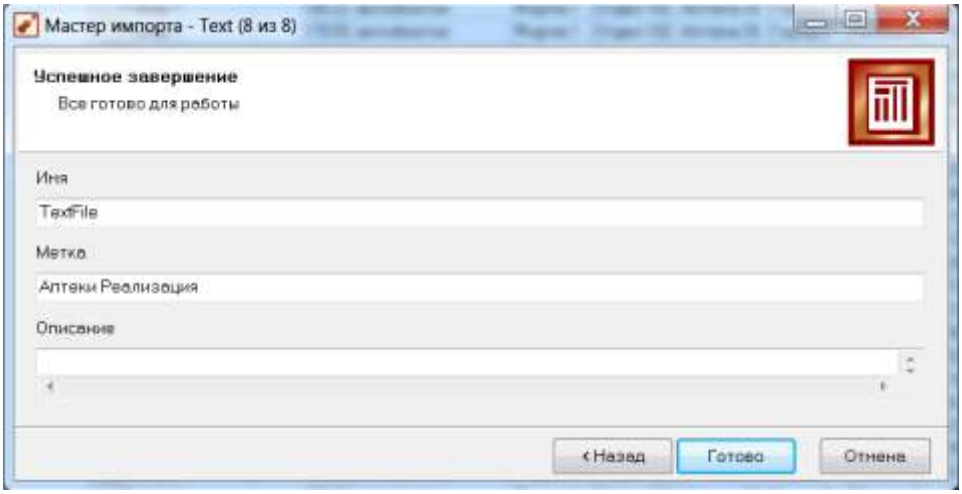


Импорт данных успешно выполнен.

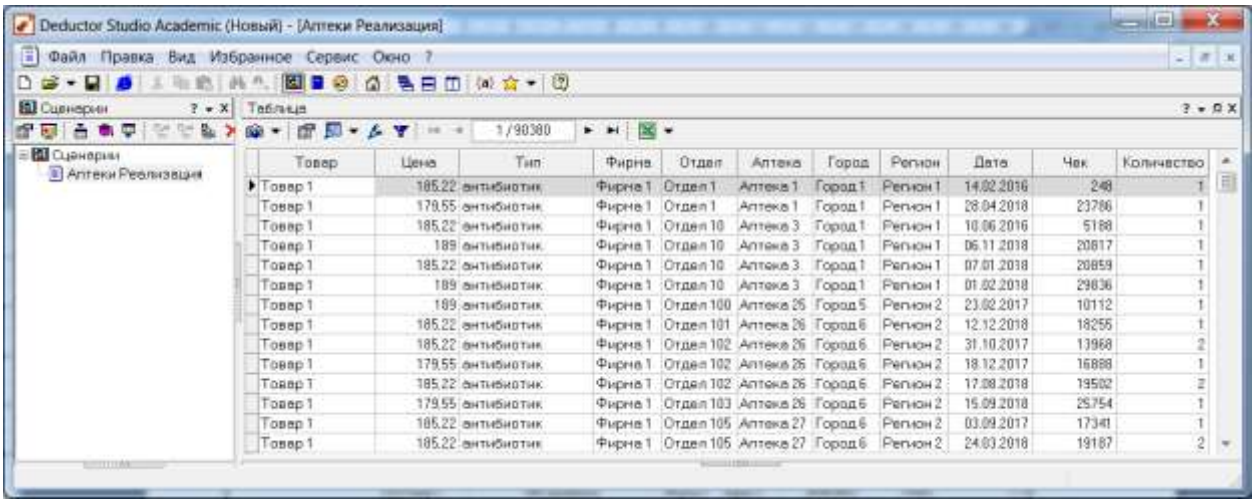
Далее выбираем способ отображения информации. Оставим пока только способ «Таблица», остальные визуализаторы рассмотрим позже.



На следующей странице можем задать содержательное имя для нашего сценария импорта:

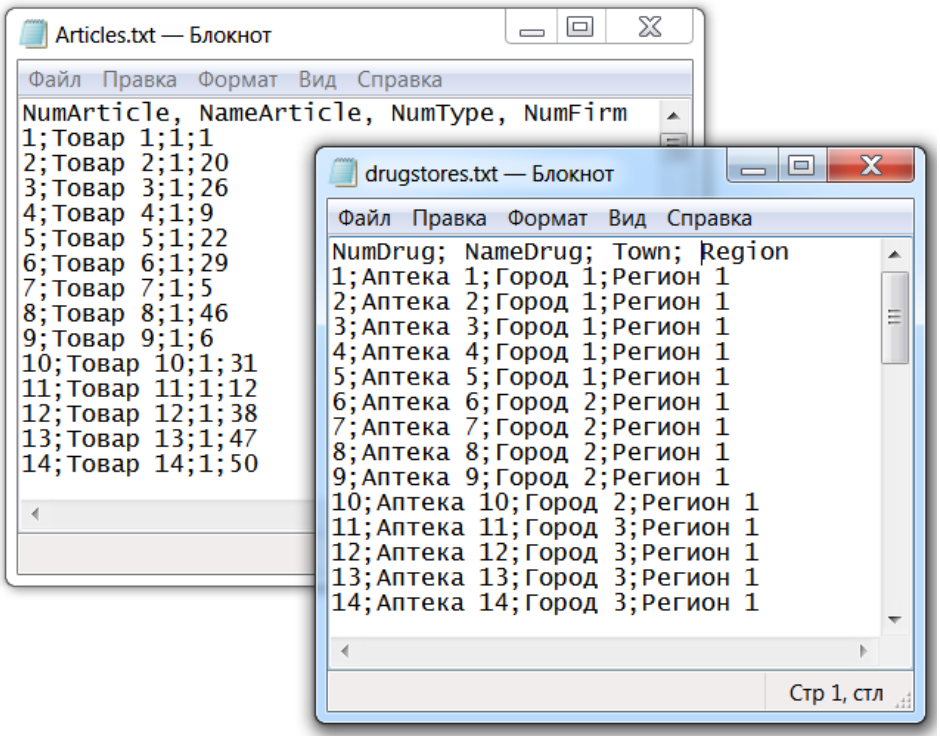


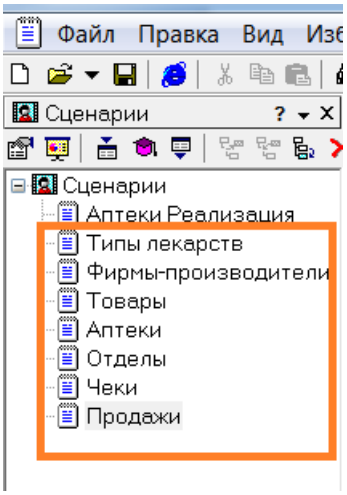
Сценарий выполнен, его результаты можно увидеть на экране в форме таблицы. Сценарии отображаются в виде иерархии слева в окне проекта Deductor Studio (у проектов тип файла **ded**).



Теперь рассмотрим более сложный вариант работы. Будем создавать хранилище данных в **СУБД FireBird**. Но предварительно загрузим данные из текстовых файлов.

Во всех текстовых файлах нужно задать строку заголовков. Поскольку данные будут загружаться в **СУБД FireBird**, а русские буквы в названиях столбцов использовать не следует, названия столбцов напишем латинскими буквами:



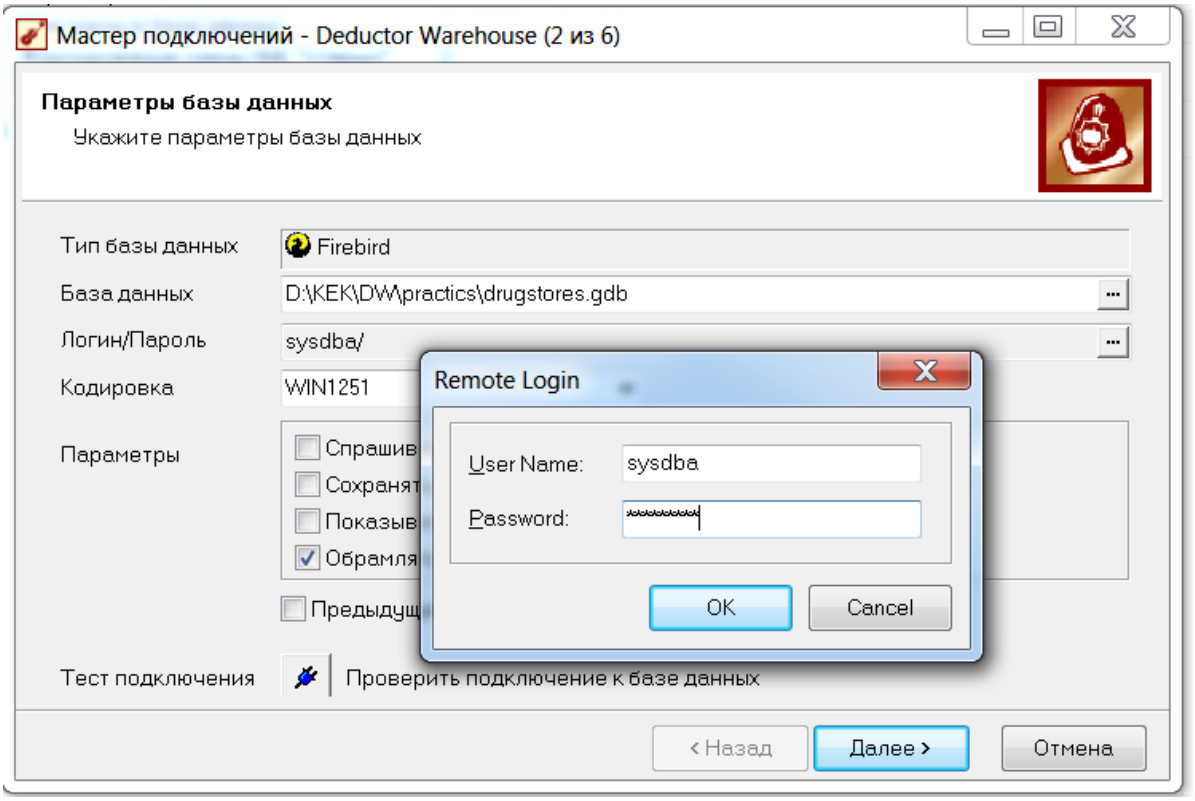


Итак, создадим 7 сценариев загрузки из текстовых файлов:

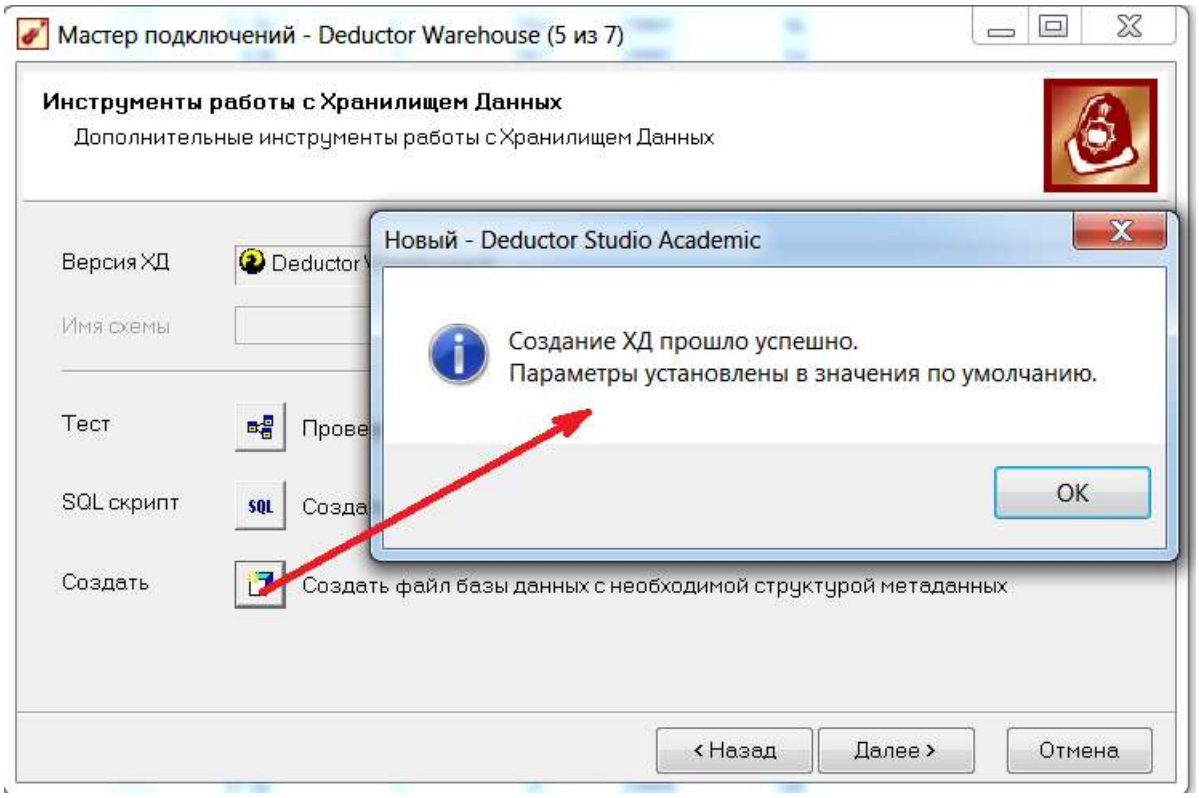
- для типов лекарств,
- фирм,
- аптек,
- отделов,
- товаров,
- чеков
- и продаж:

Теперь будем создавать хранилище данных. На вкладке «**Подключения**» создадим новое хранилище данных с помощью мастера подключений. (Тип файлов у хранилищ данных **gdb**.)

На 2-м шаге мастера нужно **задать любое** имя файла для хранилища данных (посмотрите, в каком каталоге будет создаваться файл хранилища). Кроме того, нужно задать логин и пароль: по умолчанию в СУБД FireBird есть логин администратора **sysdba** и пароль **masterkey.**

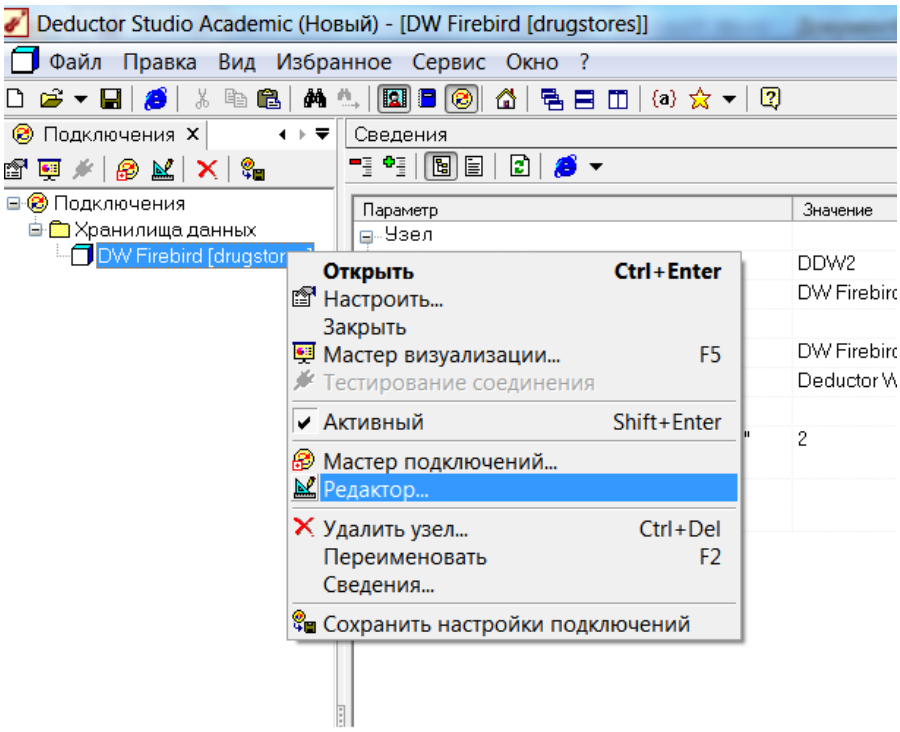


Тест подключения пока не сработает, потому что мы ещё физически не создали файл для базы данных. Это создание происходит на 5-м шаге: нужно нажать на кнопку **«Создать файл базы данных с необходимой структурой метаданных»:**

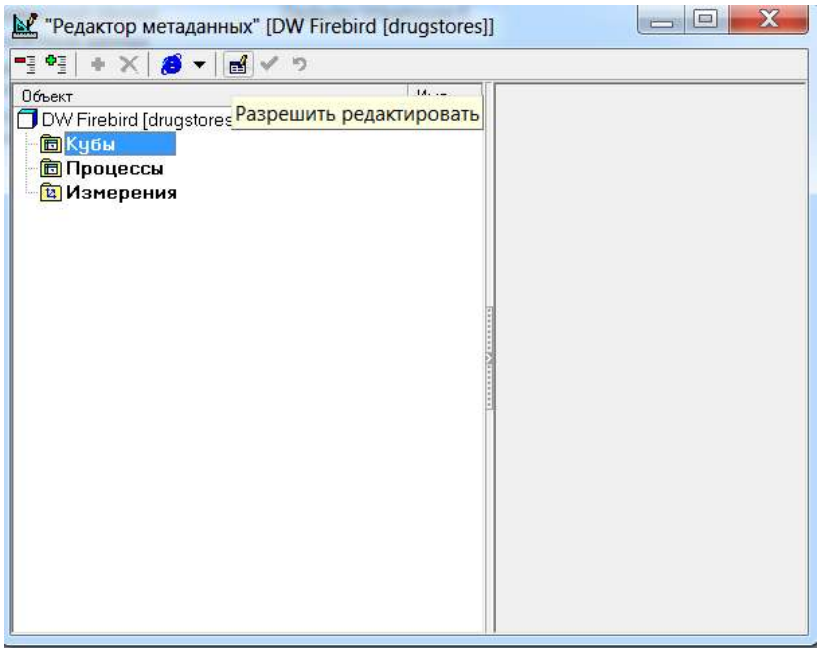


Примечание: для создания **нового** подключения к уже **существующему** хранилищу данных (например, при переносе хранилища, т.е. файла gdb на другой компьютер) на кнопку **«Создать файл базы данных с необходимой структурой метаданных»** нажимать не нужно, а все остальные шаги подключения идентичны.

Далее для создания таблиц нужно щелкнуть правой кнопкой мыши по имени хранилища данных и в контекстном меню выбрать «Редактор»:



В появившемся окне редактора следует нажать на кнопку **«Разрешить редактировать»:**

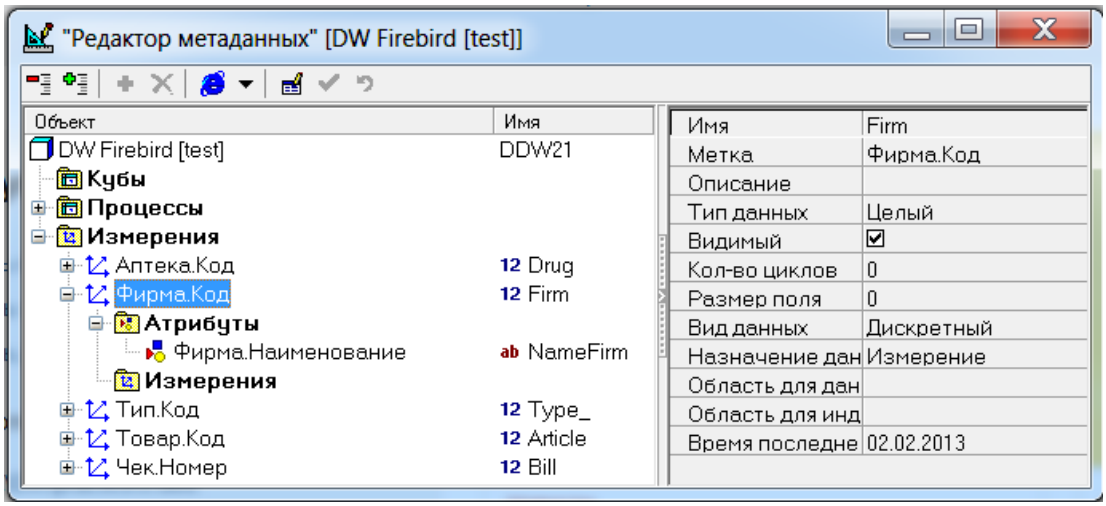


Теперь можно создавать **измерения**, затем **процессы**. Структура измерений и процессов должна соответствовать данным в текстовых файлах, которые мы выгрузили из SQL Server. Не забывайте сохранять изменения с помощью кнопки «зеленая галочка» .

Создавать измерения нужно в таком порядке: сначала **независимые**, потом **зависимые**. В нашем случае, например, сначала создаются измерения **Фирма** и **Тип**, и только затем измерение **Товар**.

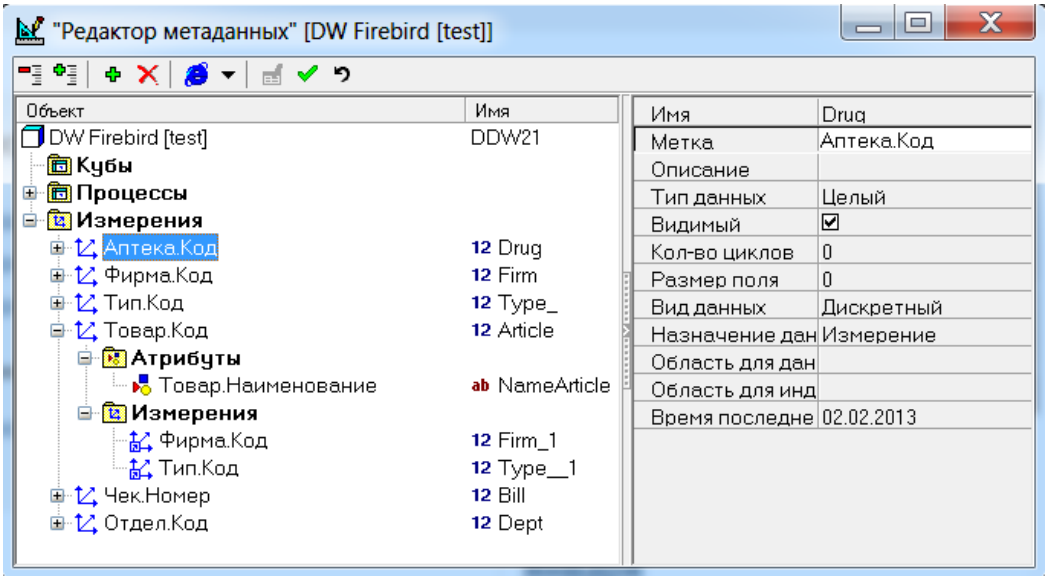
Рассмотрим подробнее создание измерения **Фирма**. В «Редакторе метаданных» щелкнем правой кнопкой на сроке «Измерения» и в контекстном меню выберем пункт Добавить.

В Deductor Studio принято, что само название измерения имеет некоторый тип. Дадим измерению имя **Firm**, в качестве метки укажем **Фирма.Код**, зададим тип данных **Целый**, вид данных **Дискретный.** Кроме того, создадим атрибут для хранения наименования: имя **NameFirm**, метка **Фирма.Наименование**, тип **Строковый.**

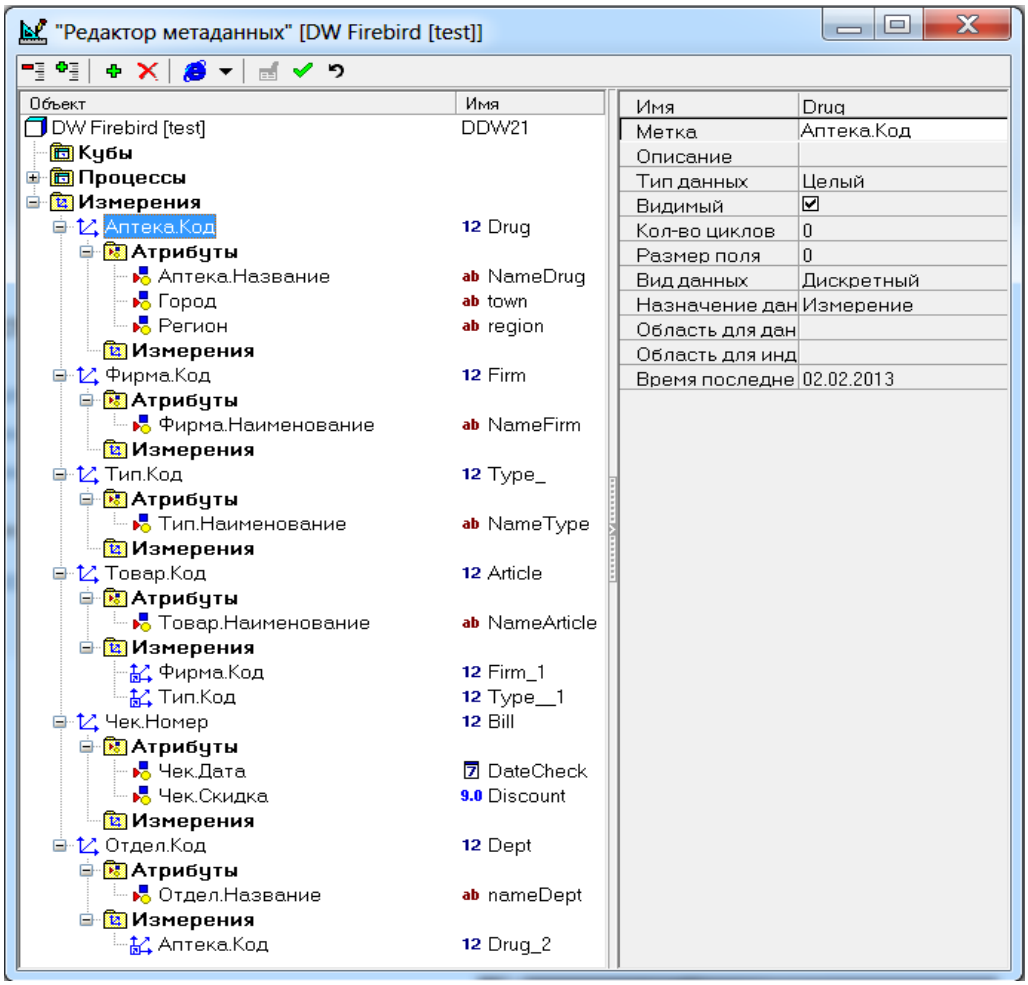


Подобным образом создаем независимые измерения **Аптека, Тип, Чек**. Обратите внимание, что типы данных в хранилище **должны соответствовать** типам данных в загруженных ранее текстовых файлах! В противном случае в дальнейшем не получится перенести данные из файлов в хранилище.

Измерение **Товар** имеет более сложную структуру: оно содержит ссылки на измерения **Тип** и **Фирма**:

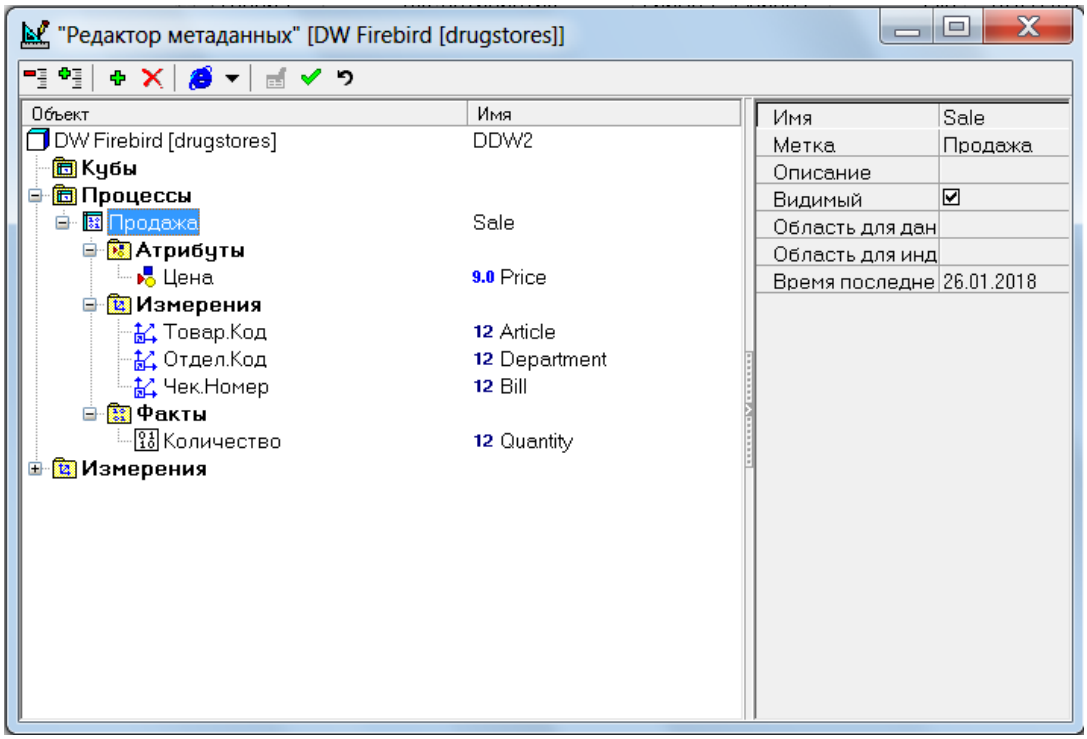


Вот так выглядят все созданные измерения:



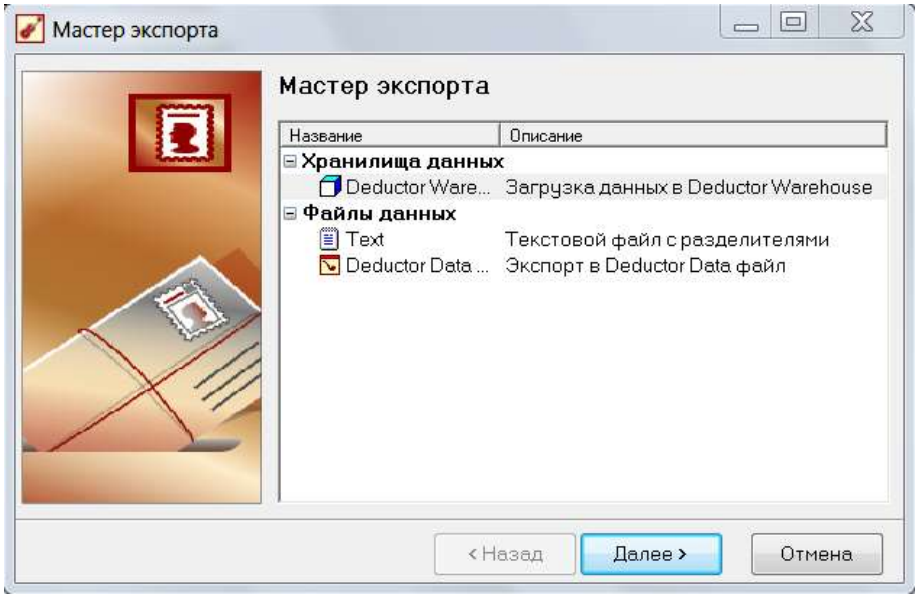
После создания измерений создадим процесс **Продажа**. У процессов, в отличие от измерений, имя не является кодом и не имеет типа.

В этом процессе есть три измерения: **Аптека, Товар**, **Чек**, факт: **Количество** и атрибут: **Цена** (цену можно было сделать и фактом).



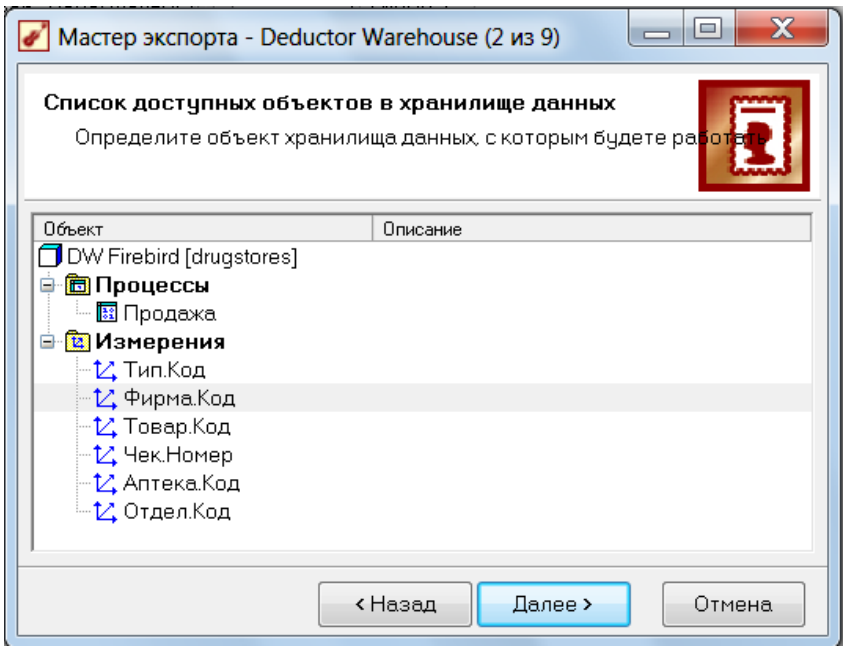
Теперь наше хранилище готово для загрузки информации из файлов. Сначала надо загружать измерения, затем процессы. Например, заполним измерение **Фирмы**. В списке сценариев нужно щелкнуть правой кнопкой мыши по текстовому файлу «**Фирмы**» и из контекстного меню выбрать «**Мастер экспорта**».

Выбираем направление загрузки: в хранилище данных.

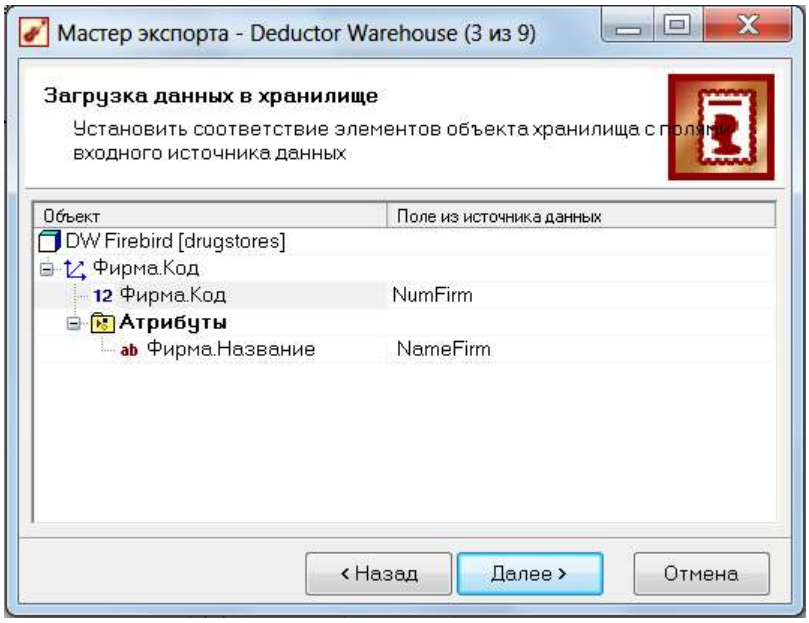


На 1-м шаге выбираем подключение.

На 2-м шаге выбираем измерение:



На 3-м шаге обязательно нужно настроить **соответствие** столбцов измерения и столбцов источника данных. Если столбцы были заданы в одном и том же порядке и в файлах, и в конструкторе хранилища, то соответствие будет предлагаться автоматически, как на рисунке:

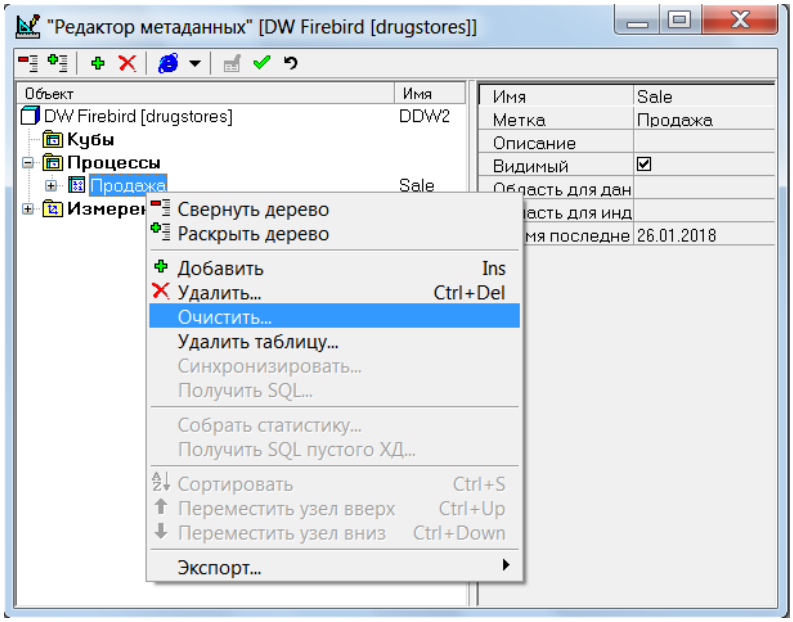


Если же столбцы назначены неправильно, исправьте назначения на правильные. Если в списке полей из источника данных **отсутствуют** какие-то столбцы, то, скорее всего, вы задали **разный** тип данных в структуре хранилища и при чтении из файлов. Вернитесь к соответствующим этапам и приведите данные к **одинаковому** типу (обратите особое внимание на целые и вещественные типы).

Остальные шаги не вызывают трудностей. Загрузите таким образом:

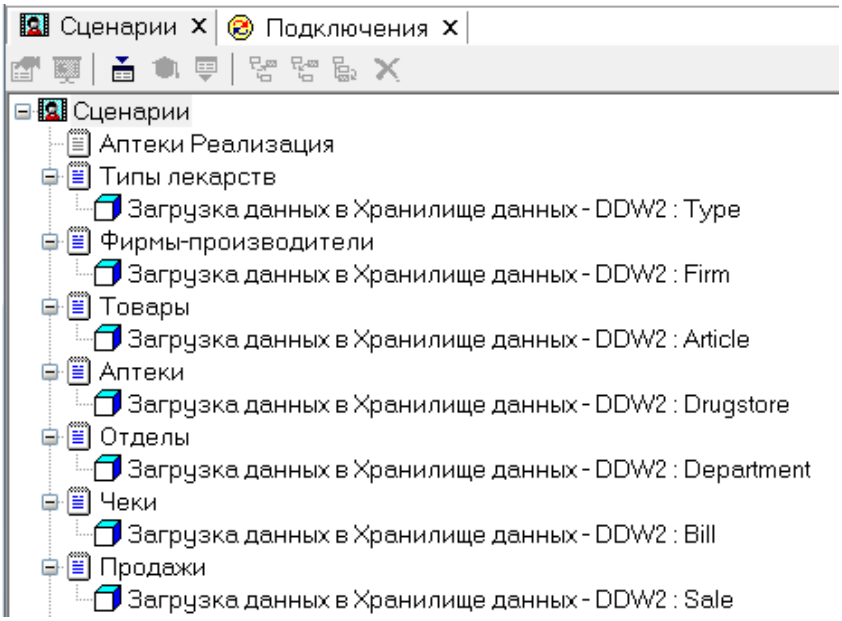
- сначала **независимые** измерения,
- затем **зависимые** измерения
- и, наконец, **процессы**.

Примечание: при **повторной** загрузке данных в **процесс** могут возникать ошибки. В этом случае проще всего явно удалить старые данные, т.е., очистить процесс в хранилище:

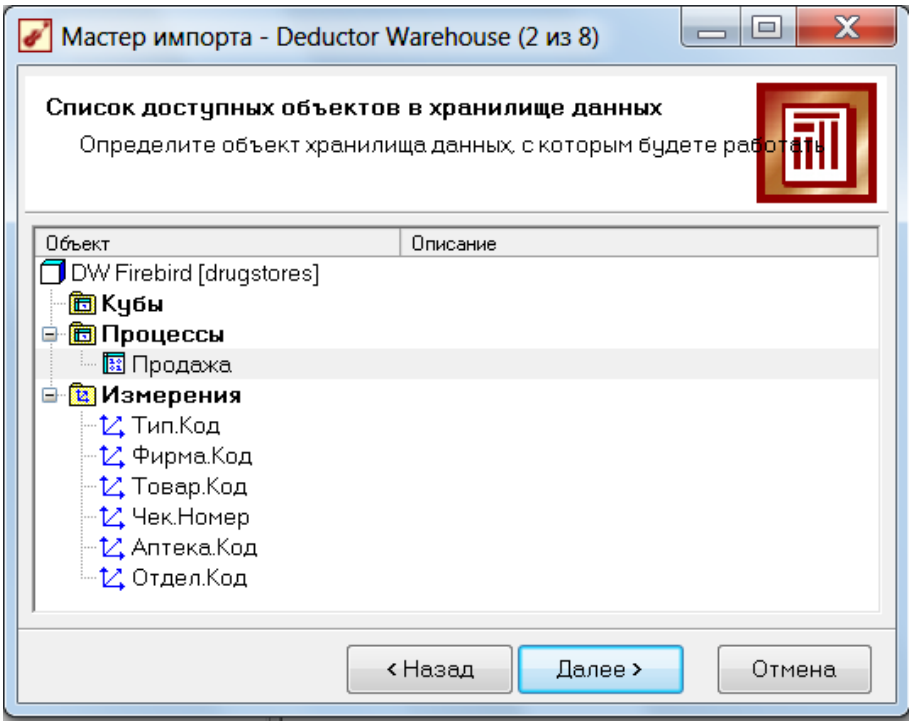


а затем заполнить его заново.

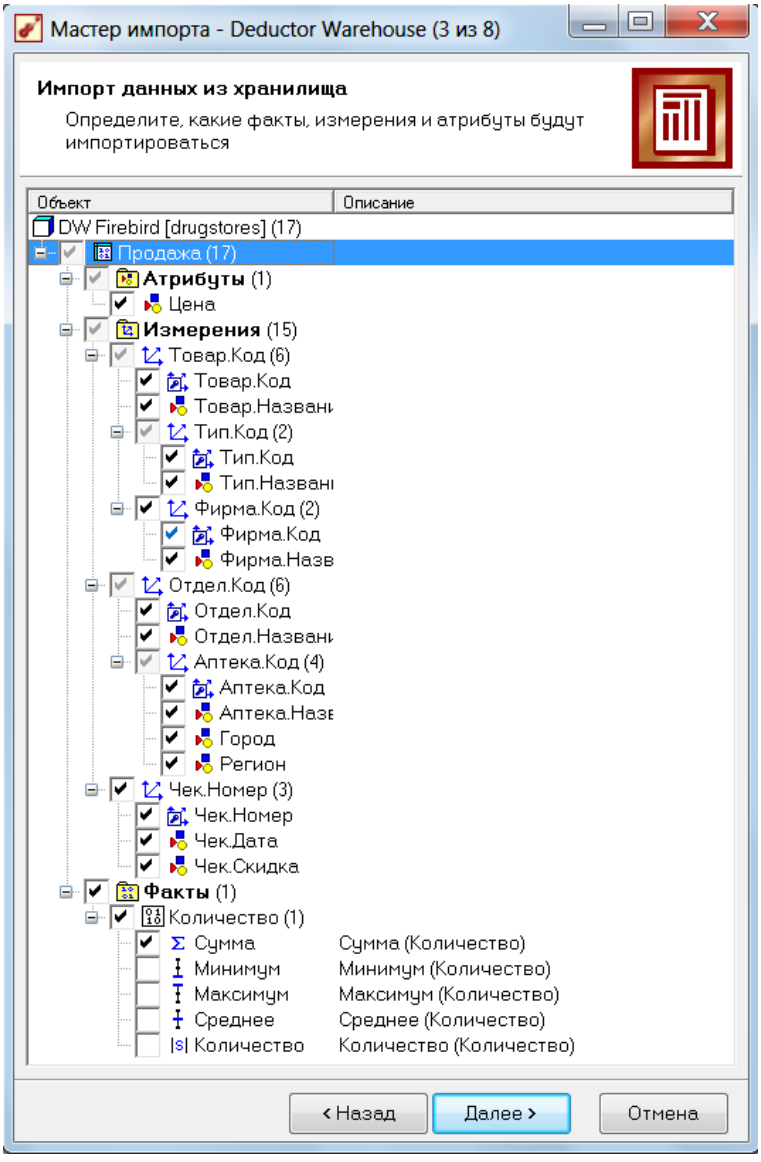
Теперь у нас есть заполненное хранилище данных:



Проверим, правильно ли оно заполнено. Для этого создадим сценарий **импорта** из хранилища данных. Запускаем **Мастер импорта** из верхнего уровня дерева сценариев, выбираем в качестве источника **Deductor Warehouse**, на 2-м шаге выбираем процесс **Продажа**:



На следующем шаге раскроем все «плюсы», для атрибутов и измерений и отметим все «галочки», а для факта **Количество** задаем способ агрегации **Сумма**:



На остальных шагах мастера оставляем всё без изменений. Мы получили из хранилища нужный нам набор данных.

По умолчанию для просмотра данных задан визуализатор «Таблица».

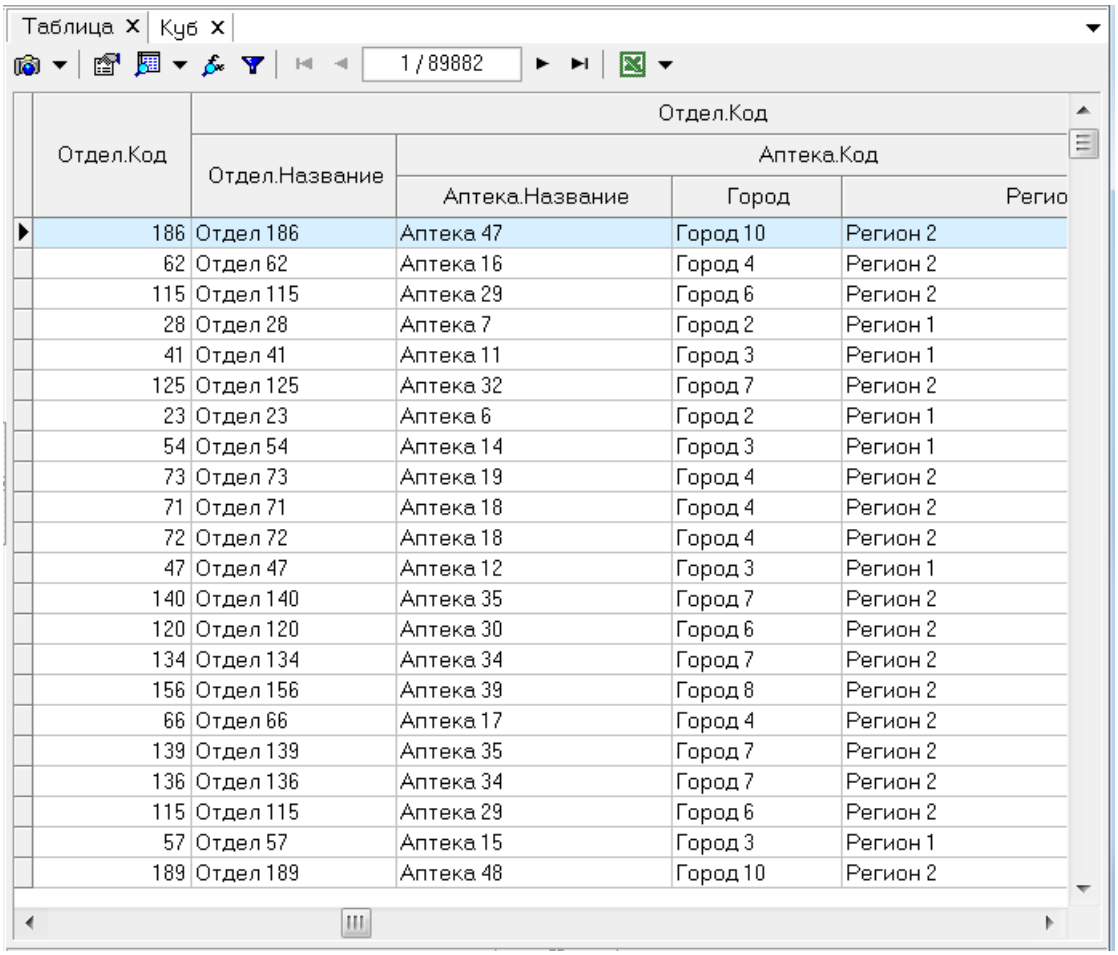


Таблица × Куб ×

1 / 89882

Отдел.Код	Отдел.Название	Аптека.Код		
		Аптека.Название	Город	Регио
186	Отдел 186	Аптека 47	Город 10	Регион 2
62	Отдел 62	Аптека 16	Город 4	Регион 2
115	Отдел 115	Аптека 29	Город 6	Регион 2
28	Отдел 28	Аптека 7	Город 2	Регион 1
41	Отдел 41	Аптека 11	Город 3	Регион 1
125	Отдел 125	Аптека 32	Город 7	Регион 2
23	Отдел 23	Аптека 6	Город 2	Регион 1
54	Отдел 54	Аптека 14	Город 3	Регион 1
73	Отдел 73	Аптека 19	Город 4	Регион 2
71	Отдел 71	Аптека 18	Город 4	Регион 2
72	Отдел 72	Аптека 18	Город 4	Регион 2
47	Отдел 47	Аптека 12	Город 3	Регион 1
140	Отдел 140	Аптека 35	Город 7	Регион 2
120	Отдел 120	Аптека 30	Город 6	Регион 2
134	Отдел 134	Аптека 34	Город 7	Регион 2
156	Отдел 156	Аптека 39	Город 8	Регион 2
66	Отдел 66	Аптека 17	Город 4	Регион 2
139	Отдел 139	Аптека 35	Город 7	Регион 2
136	Отдел 136	Аптека 34	Город 7	Регион 2
115	Отдел 115	Аптека 29	Город 6	Регион 2
57	Отдел 57	Аптека 15	Город 3	Регион 1
189	Отдел 189	Аптека 48	Город 10	Регион 2

При щелчке по столбцу таблицы можно упорядочить данные по возрастанию значений в этом столбце. Окно таблицы имеет свою небольшую панель инструментов, в которой можно задать фильтры на данные, изменить формат представления данных и т.п. Здесь же указывается количество загруженных строк. **Проверьте**, совпадает ли это значение с количеством выгруженных строк из SQL server. Полезно также будет пролистать таблицу по горизонтали и вертикали и проверить, нет ли пустых ячеек.

Задание 6б. Загрузите данные в **Deductor Studio** из текстовых файлов, с созданием хранилища данных (7 баллов), либо без него (1 балл).

Этап 6в: Преобразования и визуализаторы.

Мы не будем отделять эти две темы друг от друга, поскольку они друг от друга зависят и друг друга дополняют.

Разнообразные визуализаторы позволяют показать данные в наглядном виде. Рассмотрим визуализаторы:

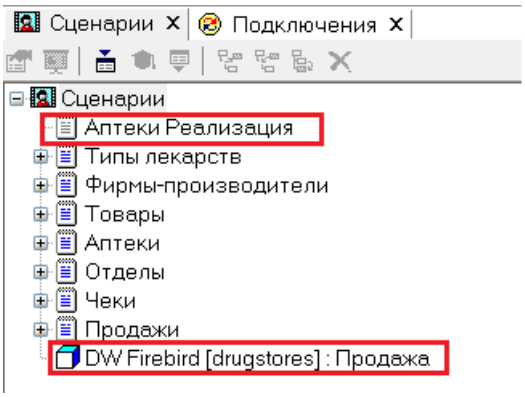
- Куб,
- Диаграмма,
- Детализация.

Разнообразные преобразования позволяют приводить данные к нужному виду. Рассмотрим преобразования:

- Калькулятор,
- Преобразование даты,
- Фильтр,
- Группировка,
- Квантование.

В рамках предыдущего задания вы либо загрузили информацию из единого текстового файла, либо создали хранилище данных.

Дальнейшие преобразования и визуализацию данных можно применять к следующим узлам дерева сценариев:

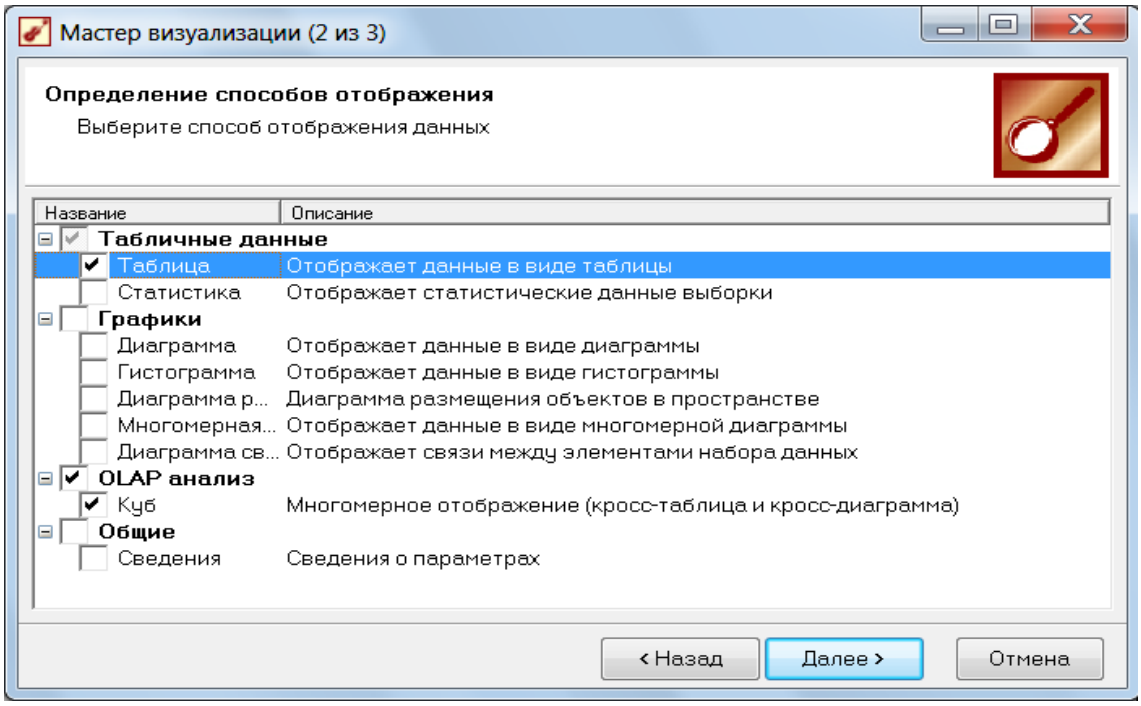


Пункт «**Аптеки Реализация**» соответствует сценарию загрузки данных из единого текстового файла.

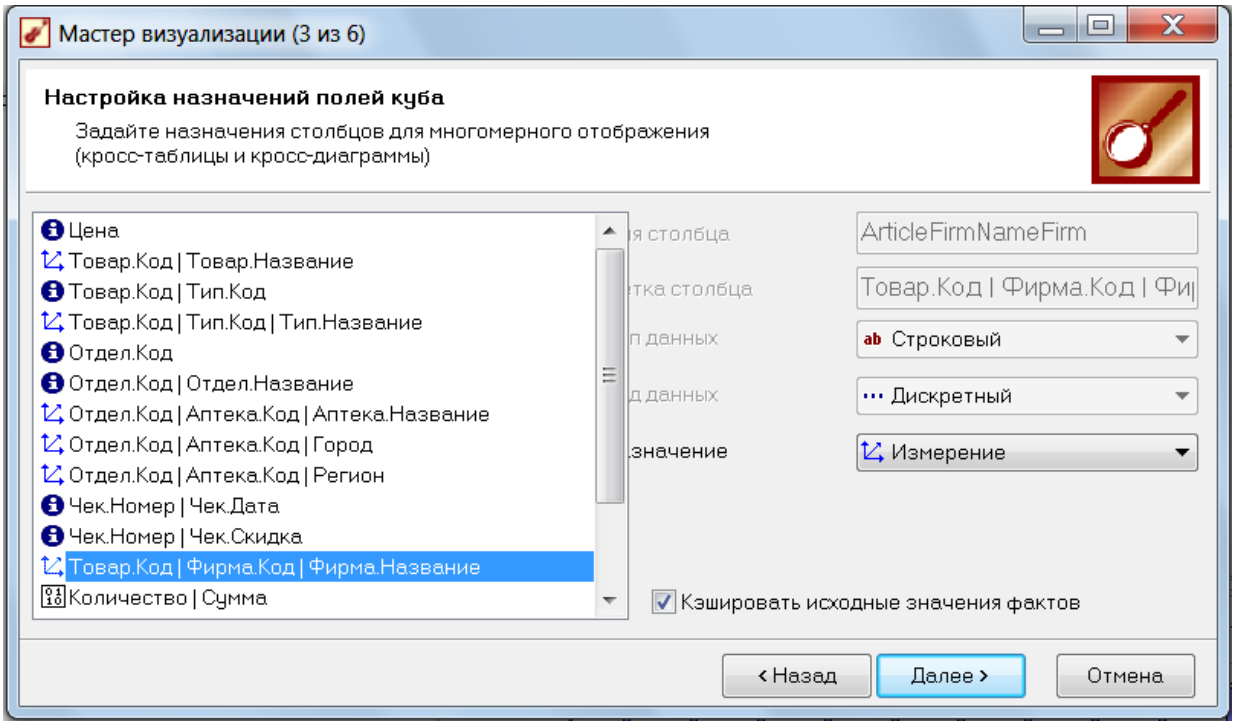
Пункт «**DW Firebird (drugstores): Продажа**» соответствует сценарию загрузки данных из хранилища.

Рассмотрим самый распространенный визуализатор «**Куб**». Для его построения нужно щелкнуть правой кнопкой мыши на нужном сценарии и выбрать «**Мастер визуализации**».

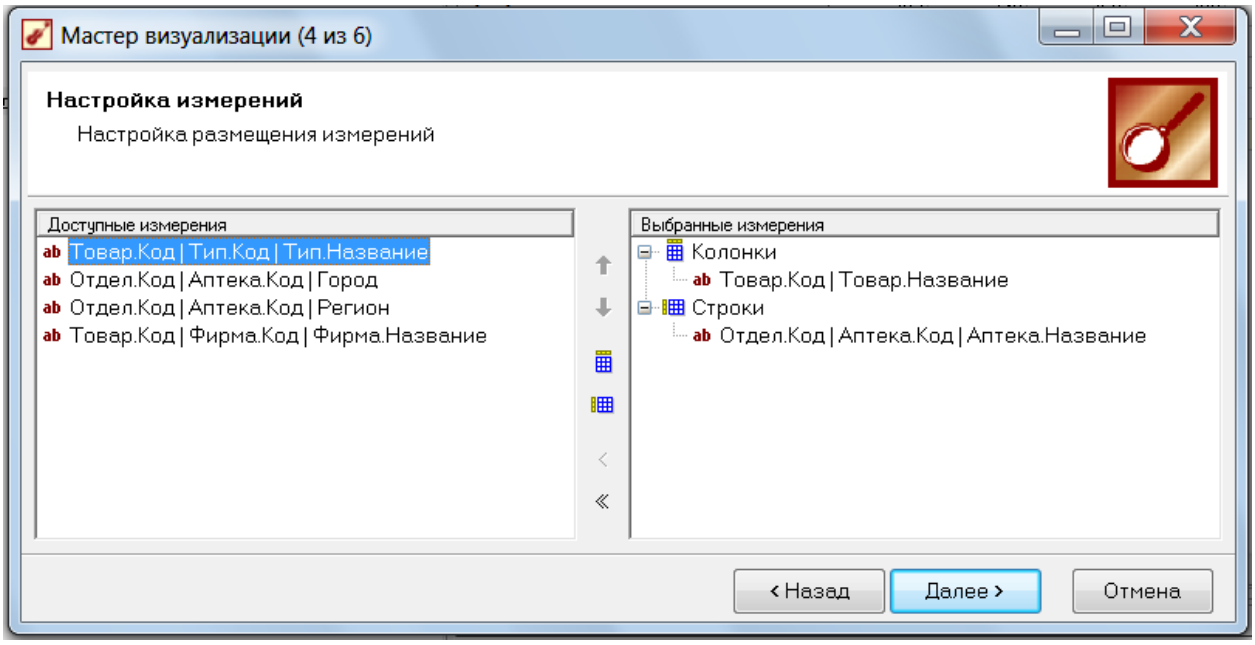
На 2-м шаге выбираем способ визуализации:



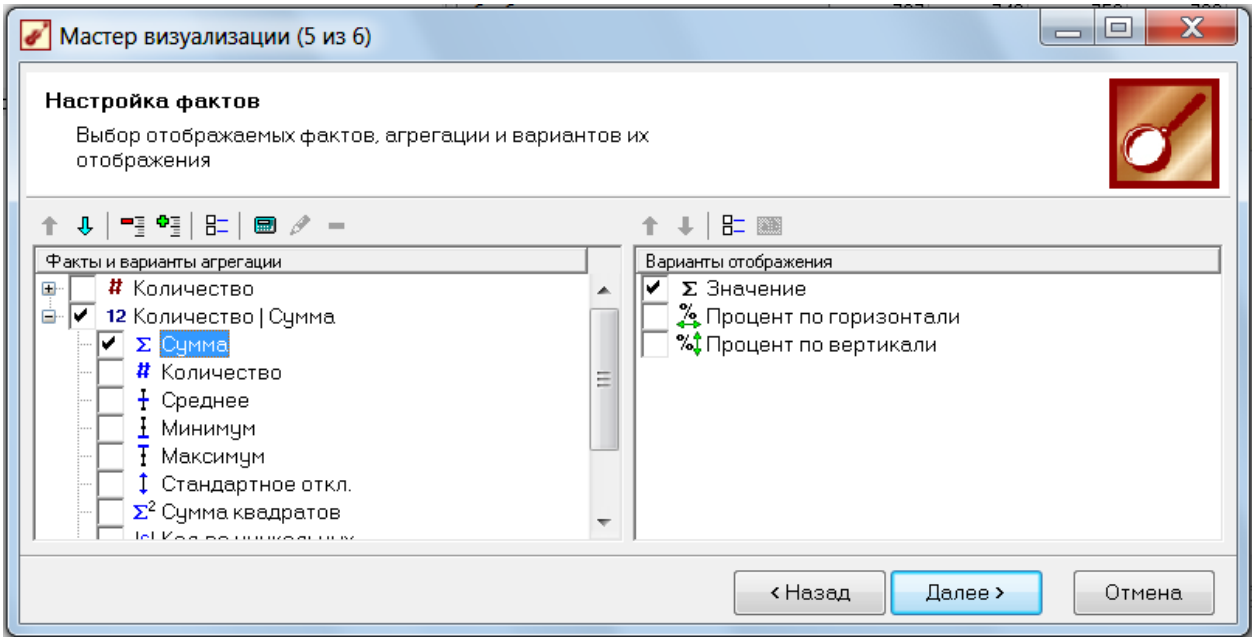
На 3-м шаге уточняем, какие столбцы нашего хранилища являются измерениями, а какие – фактами. Можно также исключить столбец из рассмотрения, если задать ему назначение «**Неиспользуемый**». По неиспользуемым измерениям данные суммируются.



На 4-м шаге настраиваем размещения измерений: какие из них будут указаны в строках таблицы, а какие – в колонках. В дальнейшем при работе с кубом их можно будет поменять местами, а также выбрать другие измерения.



На 5-м шаге уточняем, какую агрегирующую функцию применять к фактам. Для **Количества** применяем суммирование (это вторая строка в списке).



Обратите внимание на первую строку в списке с названием #Количество. Этот режим удобен для таких случаев, когда в бизнес-процессе **нет** явных числовых мер (таких как сумма сделки или количество товара). К таким ситуациям можно отнести, например, бизнес-процесс выдачи книг читателям в библиотеке, или участие спортсменов в соревнованиях. В таких случаях просто приходится подсчитывать количество строк в таблице фактов, как раз именно с помощью параметра #Количество.

Итак, мы получили куб в разрезе аптек и товаров. Этот куб содержит информацию о том, сколько раз каждый товар был куплен в каждой аптеке.

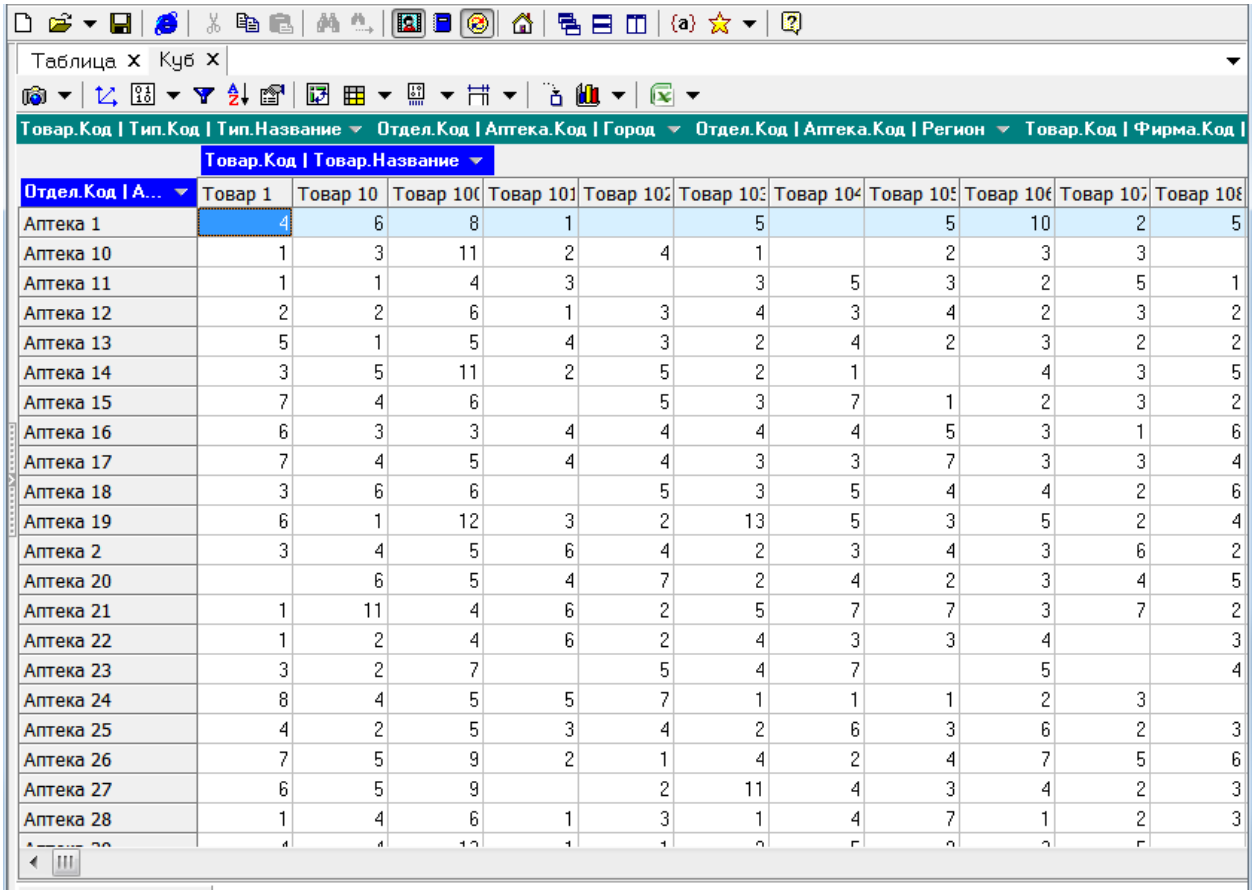


Таблица × Куб ×

Товар.Код | Тип.Код | Тип.Название ▾ Отдел.Код | Аптека.Код | Город ▾ Отдел.Код | Аптека.Код | Регион ▾ Товар.Код | Фирма.Код |

Товар.Код | Товар.Название ▾

Отдел.Код | А... ▾	Товар 1	Товар 10	Товар 10(	Товар 101	Товар 10:	Товар 10:	Товар 10⁴	Товар 10:	Товар 10(	Товар 10:	Товар 10(
Аптека 1	4	6	8	1		5		5	10	2	5
Аптека 10	1	3	11	2	4	1		2	3	3	
Аптека 11	1	1	4	3		3	5	3	2	5	1
Аптека 12	2	2	6	1	3	4	3	4	2	3	2
Аптека 13	5	1	5	4	3	2	4	2	3	2	2
Аптека 14	3	5	11	2	5	2	1		4	3	5
Аптека 15	7	4	6		5	3	7	1	2	3	2
Аптека 16	6	3	3	4	4	4	4	5	3	1	6
Аптека 17	7	4	5	4	4	3	3	7	3	3	4
Аптека 18	3	6	6		5	3	5	4	4	2	6
Аптека 19	6	1	12	3	2	13	5	3	5	2	4
Аптека 2	3	4	5	6	4	2	3	4	3	6	2
Аптека 20		6	5	4	7	2	4	2	3	4	5
Аптека 21	1	11	4	6	2	5	7	7	3	7	2
Аптека 22	1	2	4	6	2	4	3	3	4		3
Аптека 23	3	2	7		5	4	7		5		4
Аптека 24	8	4	5	5	7	1	1	1	2	3	
Аптека 25	4	2	5	3	4	2	6	3	6	2	3
Аптека 26	7	5	9	2	1	4	2	4	7	5	6
Аптека 27	6	5	9		2	11	4	3	4	2	3
Аптека 28	1	4	6	1	3	1	4	7	1	2	3

Остальные измерения также доступны для пользователя, их можно просто перетаскивать между панелью инструментов и таблицей. Например, перетаскиванием поменяем измерение **Товар** на **Тип товара**, а **Аптеку** на город:

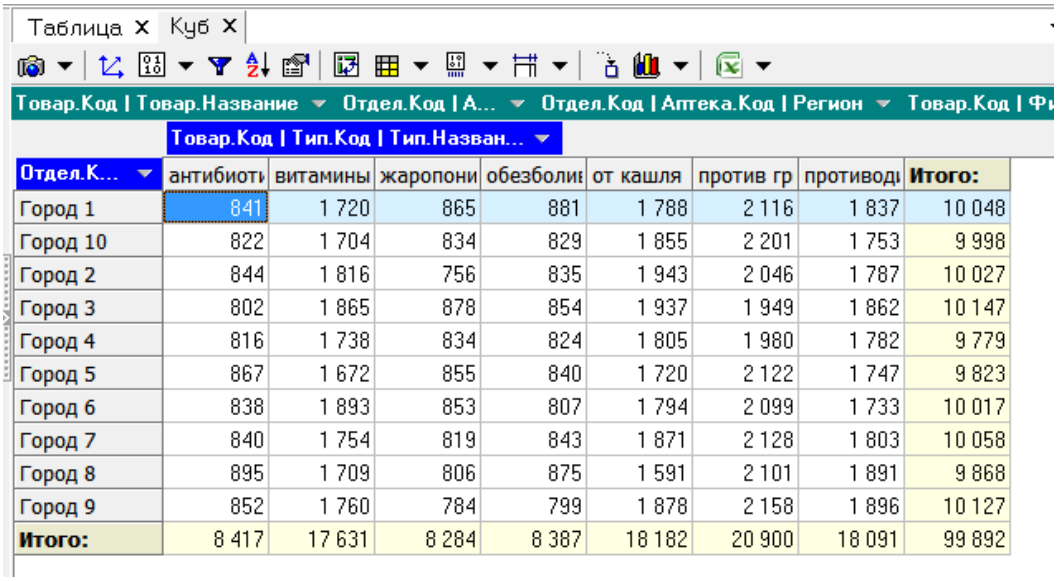


Таблица × Куб ×

Товар.Код | Товар.Название ▾ Отдел.Код | А... ▾ Отдел.Код | Аптека.Код | Регион ▾ Товар.Код | Фи

Товар.Код | Тип.Код | Тип.Назван... ▾

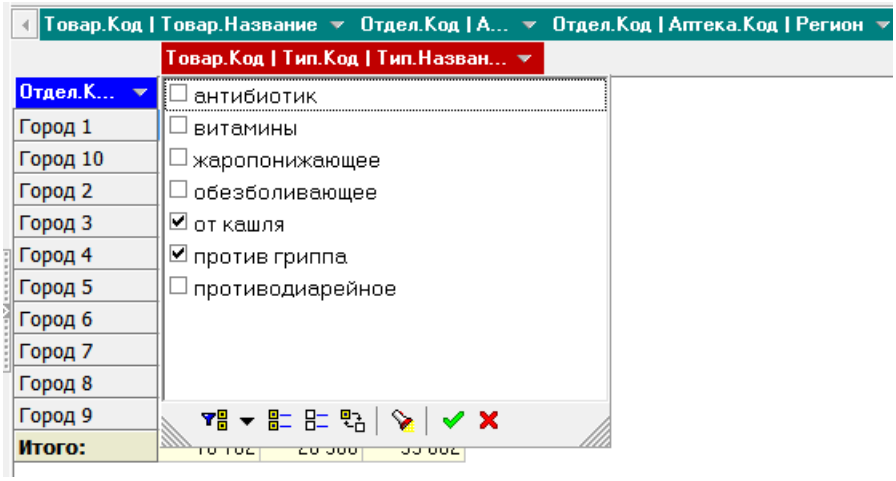
Отдел.К... ▾	антибиоти	витамины	жаропони	обезболи	от кашля	против гр	противоди	Итого:
Город 1	841	1 720	865	881	1 788	2 116	1 837	10 048
Город 10	822	1 704	834	829	1 855	2 201	1 753	9 998
Город 2	844	1 816	756	835	1 943	2 046	1 787	10 027
Город 3	802	1 865	878	854	1 937	1 949	1 862	10 147
Город 4	816	1 738	834	824	1 805	1 980	1 782	9 779
Город 5	867	1 672	855	840	1 720	2 122	1 747	9 823
Город 6	838	1 893	853	807	1 794	2 099	1 733	10 017
Город 7	840	1 754	819	843	1 871	2 128	1 803	10 058
Город 8	895	1 709	806	875	1 591	2 101	1 891	9 868
Город 9	852	1 760	784	799	1 878	2 158	1 896	10 127
Итого:	8 417	17 631	8 284	8 387	18 182	20 900	18 091	99 892

Панель инструментов над кубом позволяет «на лету» выполнить некоторые преобразования данных, отфильтровать их, отсортировать, задать формат визуализации.

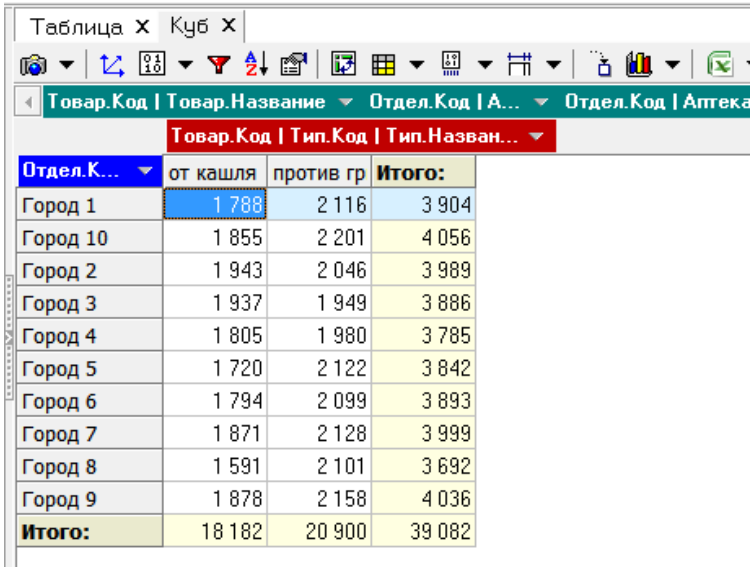
Кнопка позволяет настроить измерения.

Кнопка позволяет настроить факты и добавить вычисляемый факт.

Кнопка позволяет отфильтровать данные. Задать простые фильтры можно также, щелкнув по синему прямоугольнику с названием измерения. При задании фильтра прямоугольник становится красным. Например, зададим фильтр для типа лекарства:



Получим результат:

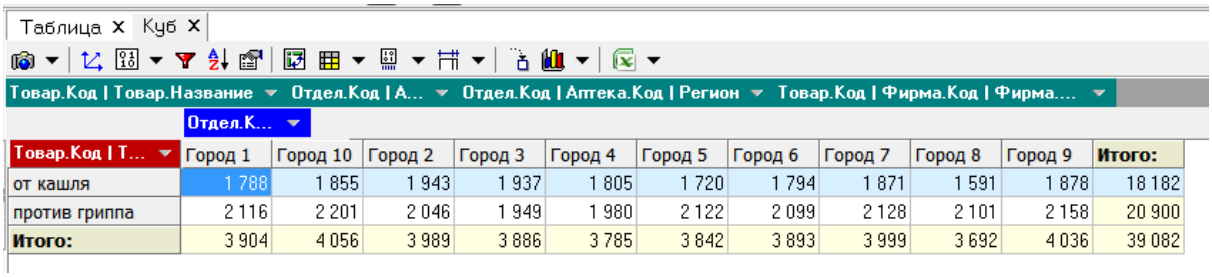


Отдел.К...	от кашля	против гр	Итого:
Город 1	1 788	2 116	3 904
Город 10	1 855	2 201	4 056
Город 2	1 943	2 046	3 989
Город 3	1 937	1 949	3 886
Город 4	1 805	1 980	3 785
Город 5	1 720	2 122	3 842
Город 6	1 794	2 099	3 893
Город 7	1 871	2 128	3 999
Город 8	1 591	2 101	3 692
Город 9	1 878	2 158	4 036
Итого:	18 182	20 900	39 082

Кнопка позволяет задать порядок сортировки.

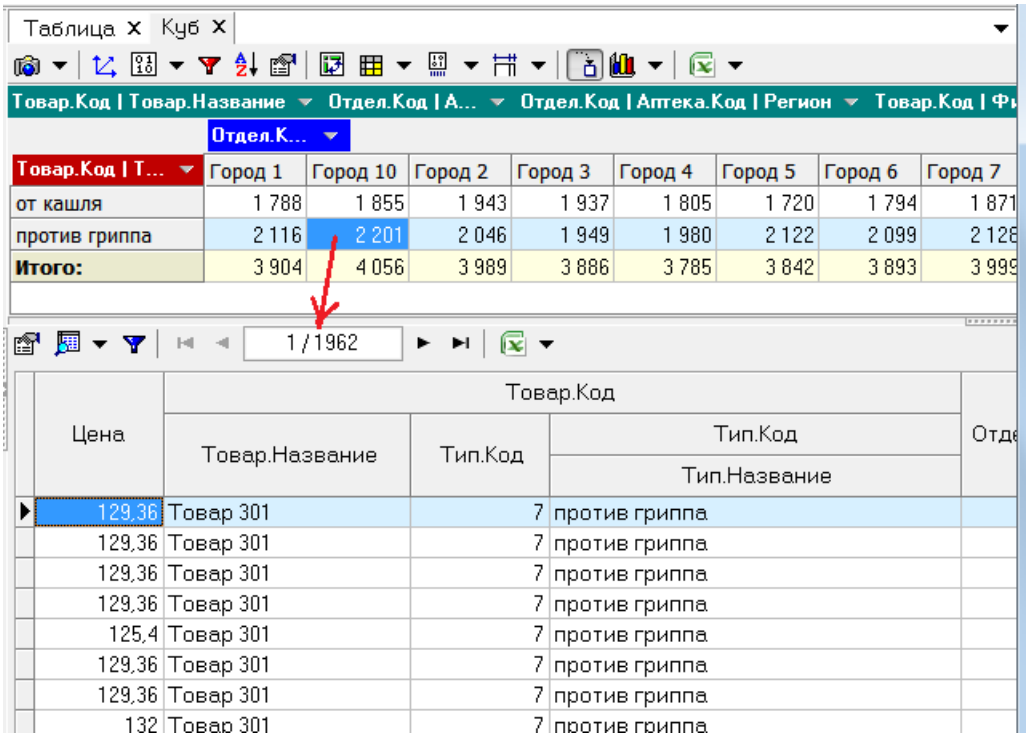
Кнопка позволяет задать формат отображения измерений и фактов.

При нажатии на кнопку  происходит транспонирование таблицы данных (строки и столбцы меняются местами):

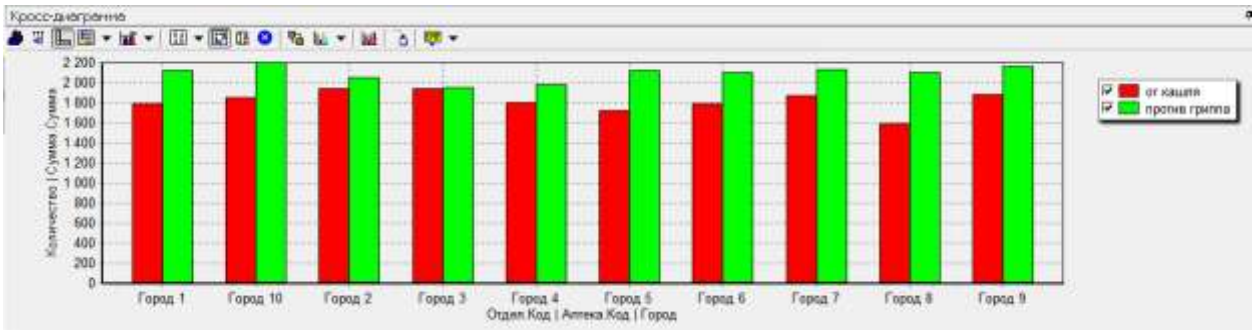


Кнопка  управляет выдачей итоговых строки и столбца.

При нажатой кнопке  внизу появляется окно с подробной детализацией текущей ячейки таблицы:

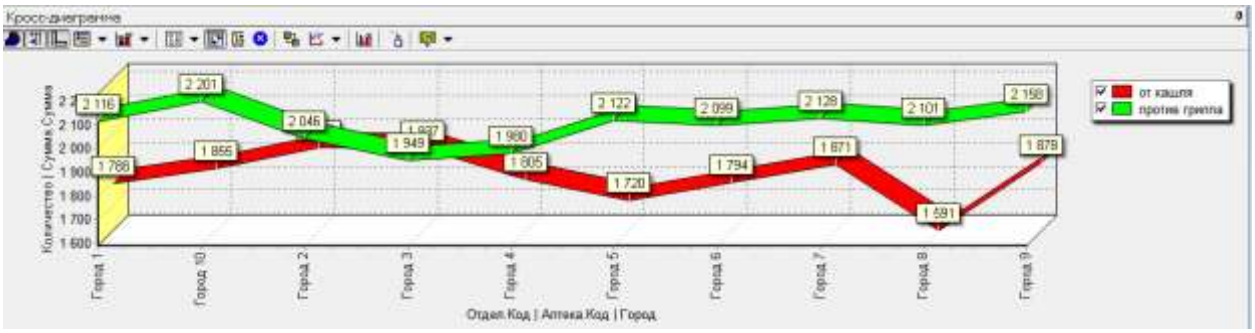


Наконец, при нажатии кнопки  из данных текущей таблицы (кроме итогов) создается диаграмма:



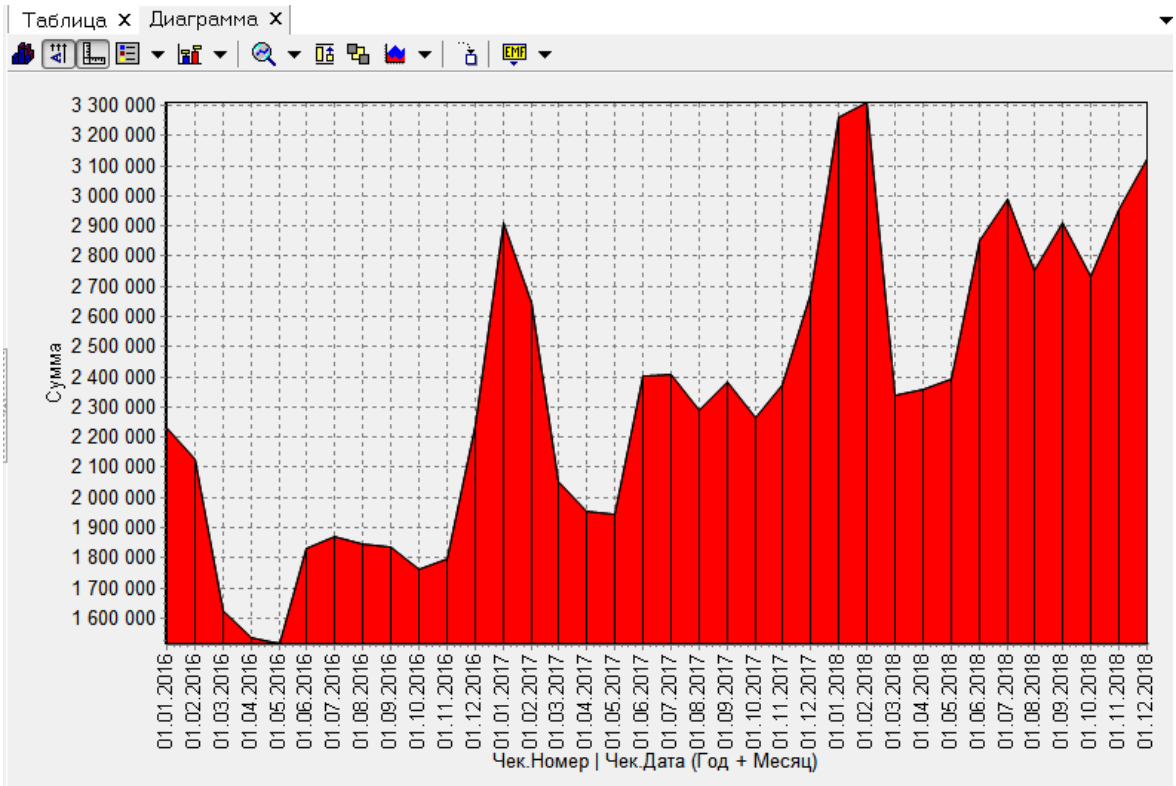
У диаграммы есть собственная панель инструментов, с помощью которой можно настраивать внешний вид диаграммы. Например, изменим

столбцовую диаграмму на линейную, установим трехмерное представление, выведем метки со значениями, изменим направление надписей на горизонтальной оси:

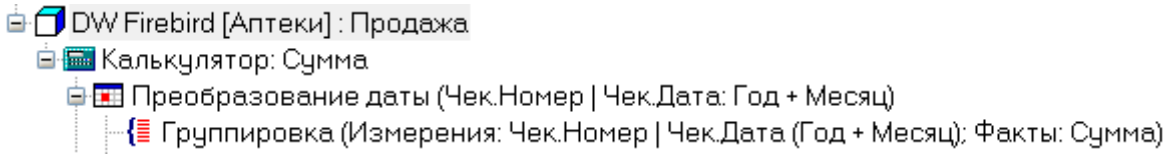


Хотя некоторые преобразования данных можно выполнить «на лету», для реализации более сложных задач в программе **Deductor Studio** можно создавать последовательности сценариев.

Например, мы хотим построить обычную диаграмму по объему продаж в денежном выражении с детализацией по месяцам:



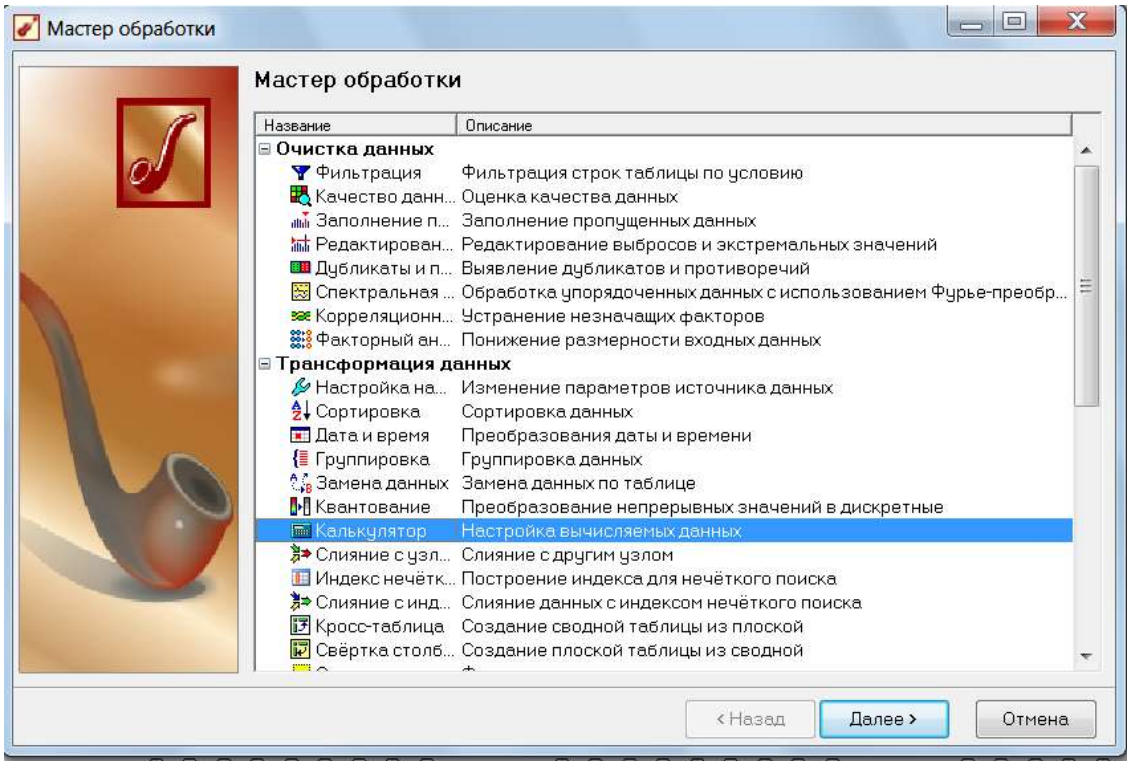
Для этого нам потребуется выполнить следующие шаги сценария:



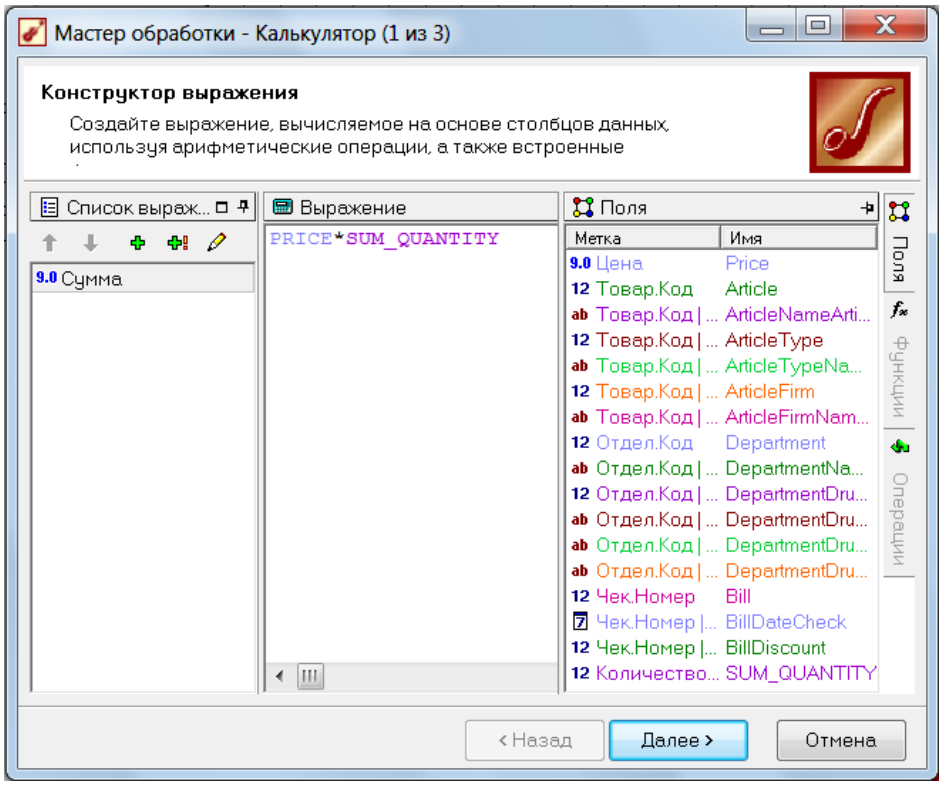
а затем использовать визуализатор «Диаграмма».

Итак, первый шаг состоит в создании вычисляемого выражения. В таблице у нас есть столбцы **Цена** и **Количество**, создадим из них вычисляемое выражение **Сумма**.

Щелкнем правой кнопкой по строке «**DW Firebird (Drugstores) : Продажа»**, выберем пункт **«Мастер обработки»**, в появившемся окне в группе **«Трансформация данных»** выберем пункт **«Калькулятор»:**



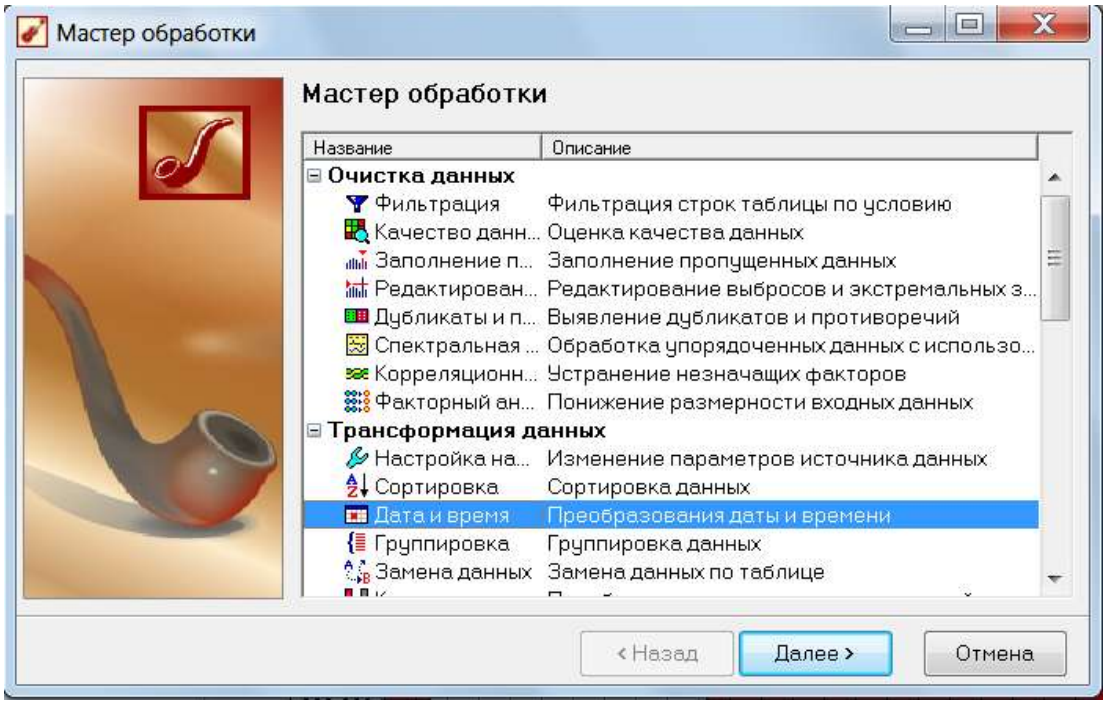
В конструкторе выражений назовем выражение «Сумма» и зададим для нее соответствующую формулу:



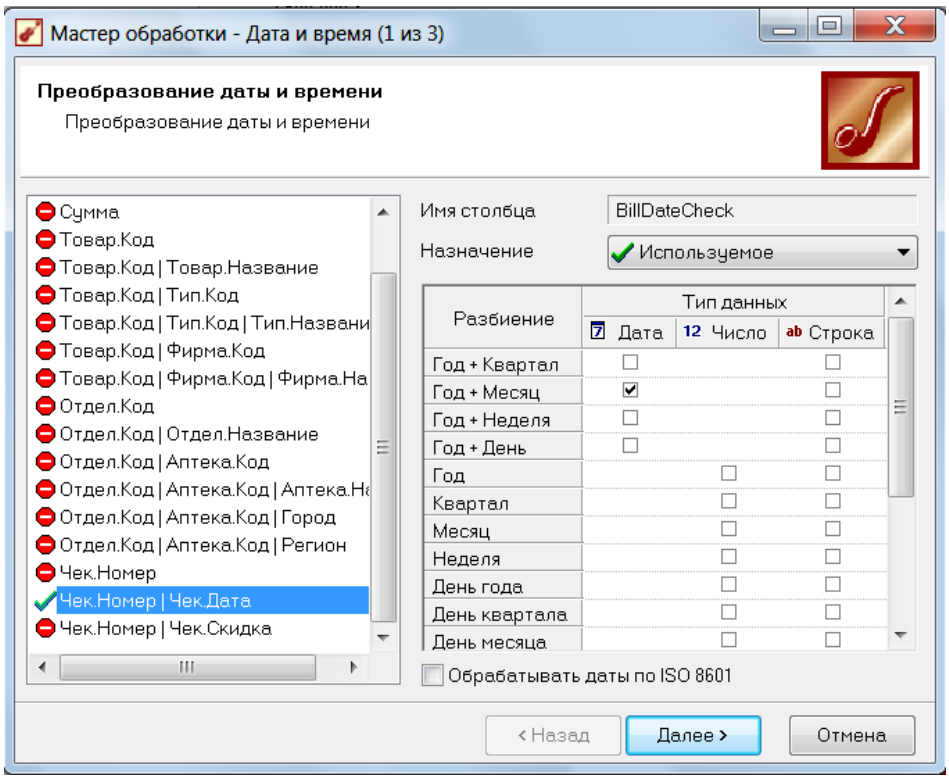
В результате работы этого мастера в таблице появится новый столбец с вычисленной суммой продажи.

Далее нам нужно сгруппировать данные по месяцам и годам, а в детализированном виде у нас есть только даты. Нужно будет выполнить преобразование дат.

Щелкнем правой кнопкой мыши по сценарию **«Калькулятор: сумма»**, запустим **«Мастер обработки»** и в группе **«Трансформация данных»** выберем **«Дата и время – преобразование даты и времени»**:



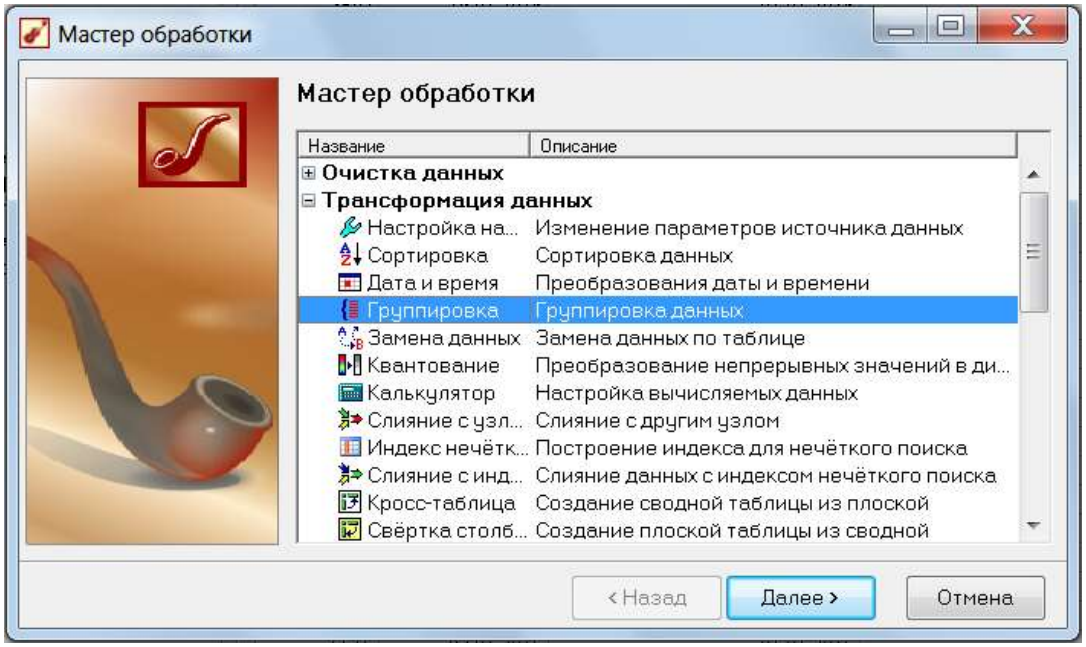
Нам нужны данные в масштабе «Год+Месяц». Выбираем в левом списке пункт Чек.Номер|Чек.Дата и в правом списке отмечаем соответствующий флажок:



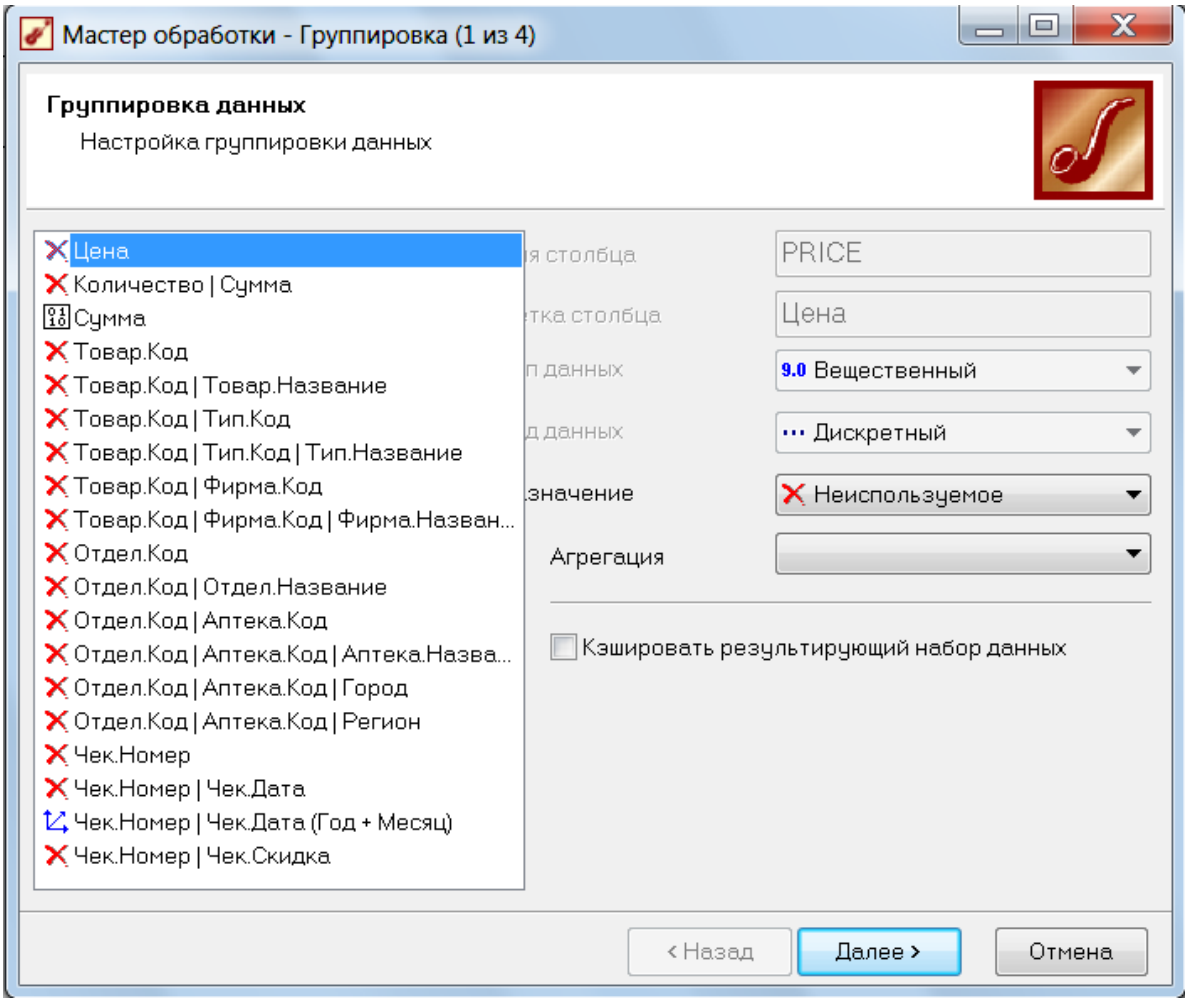
После этого в таблице данных появится новый столбец, соответствующий первому дню заданного периода, т.е., первому дню месяца:

Чек.Номер	Чек.Номер	
	Чек.Дата	Чек.Дата (Год + Месяц)
998	06.01.2016	01.01.2016
3502	19.01.2016	01.01.2016
5300	08.01.2016	01.01.2016
6310	08.01.2016	01.01.2016
7180	29.01.2016	01.01.2016
7804	05.01.2016	01.01.2016
1328	15.01.2016	01.01.2016
2060	03.01.2016	01.01.2016
2517	25.01.2016	01.01.2016
3391	08.01.2016	01.01.2016
6316	28.01.2016	01.01.2016
1021	31.01.2016	01.01.2016
1774	04.01.2016	01.01.2016
3954	21.01.2016	01.01.2016
3991	05.01.2016	01.01.2016

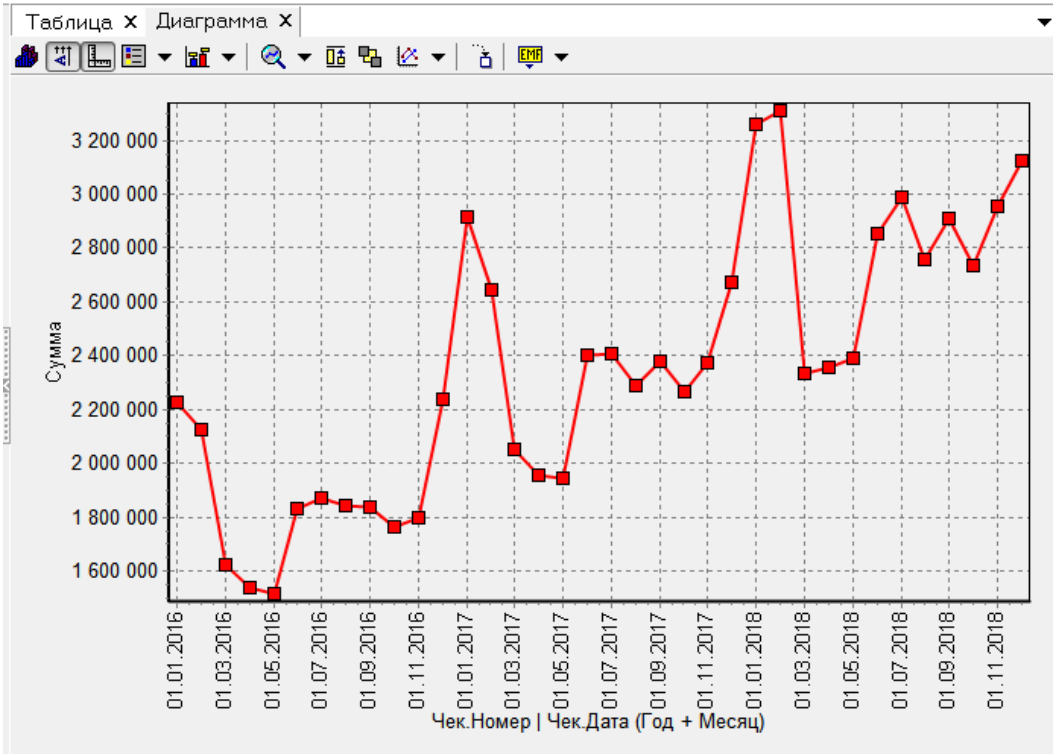
На следующем шаге добавим сценарий группировки. Щелкаем правой кнопкой мыши по сценарию **«Преобразование даты»**, вызываем **«Мастер обработки»** и в группе **«Трансформация данных»** выбираем пункт **«Группировка»**:



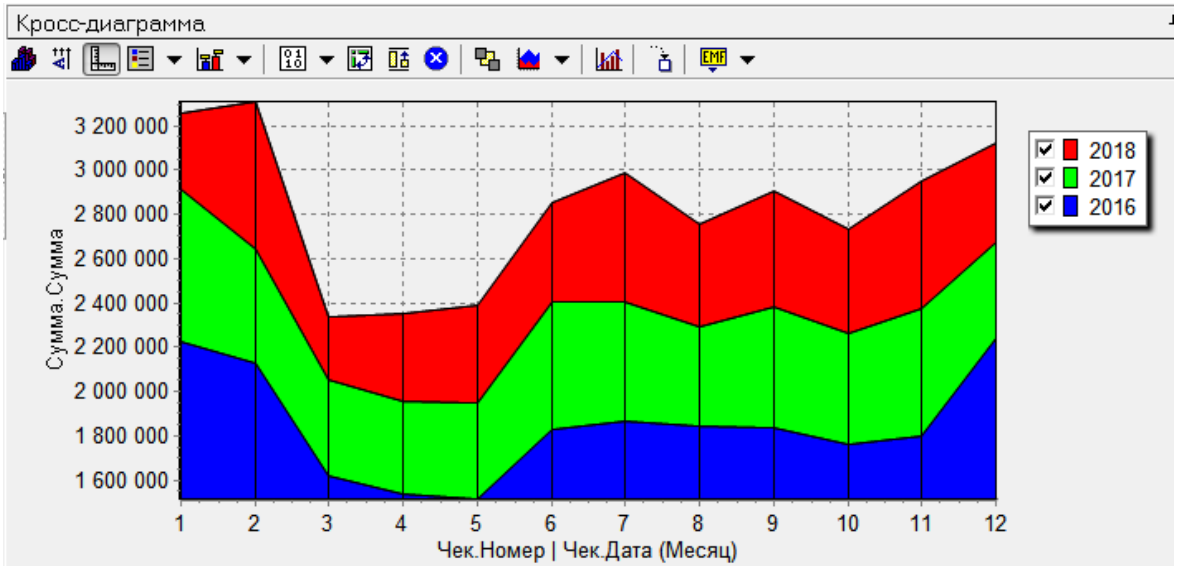
Мы хотим сгруппировать суммы по месяцу и году. Для этого в качестве измерения оставляем только столбец **«Дата (Год+Месяц)»,** а в качестве факта - **«Сумму».** Все остальные столбцы отмечаем как **«Неиспользуемые»:**



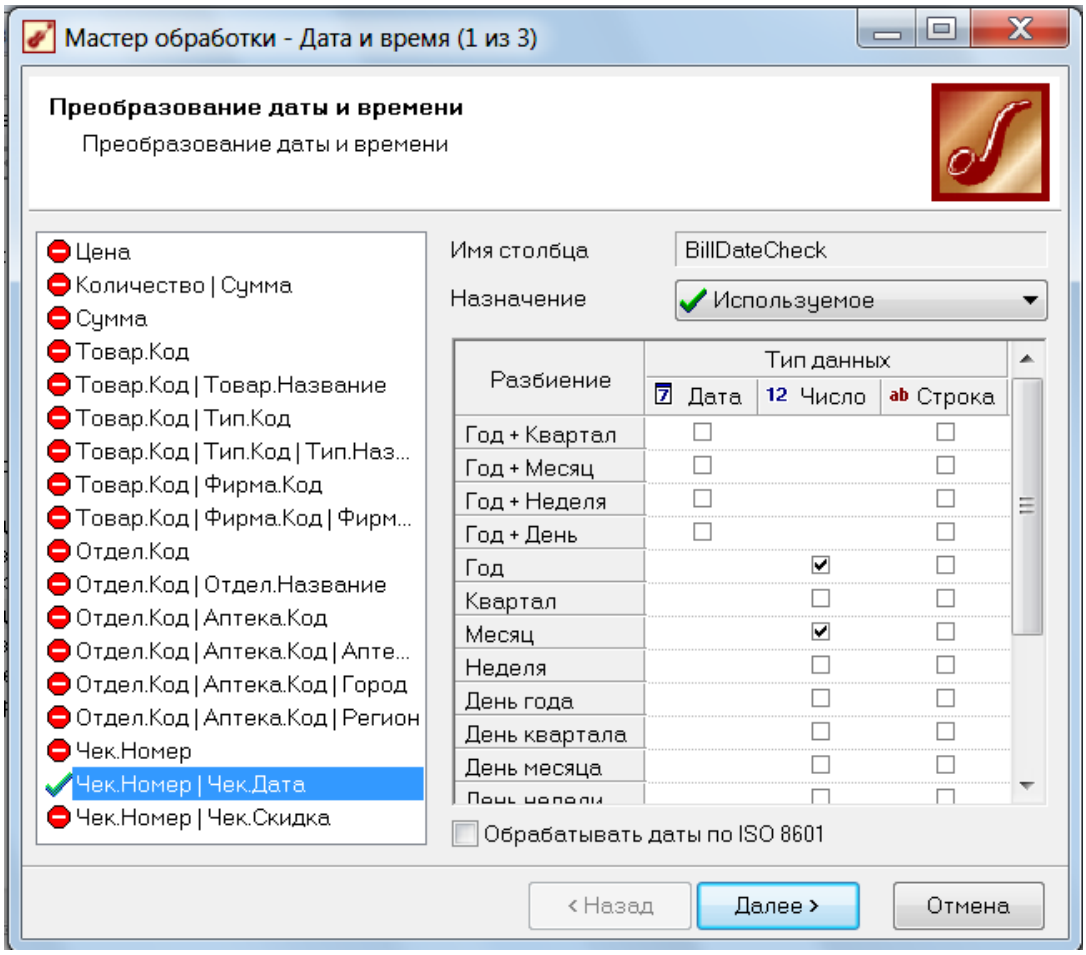
На следующем шаге будут произведены вычисления для группировки. Далее в списке визуализаторов можно выбрать **«Диаграмму».** Построим, например, диаграмму «Линии с точками»:



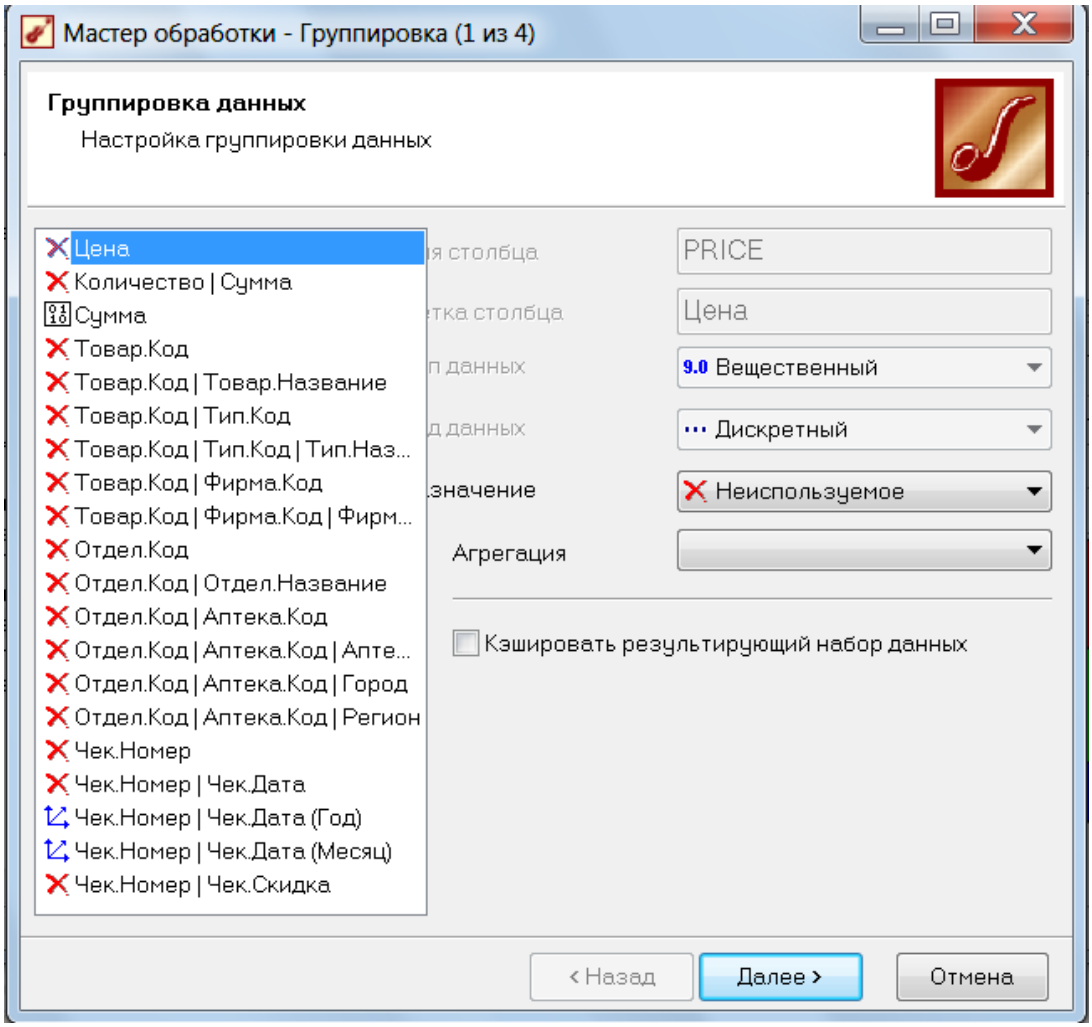
На диаграмме хорошо видно, что пик продажи лекарств приходится на зимние месяцы. **Вопрос:** а можно построить диаграмму, на которой продажи за 3 года отображаются в виде 3 отдельных графиков?



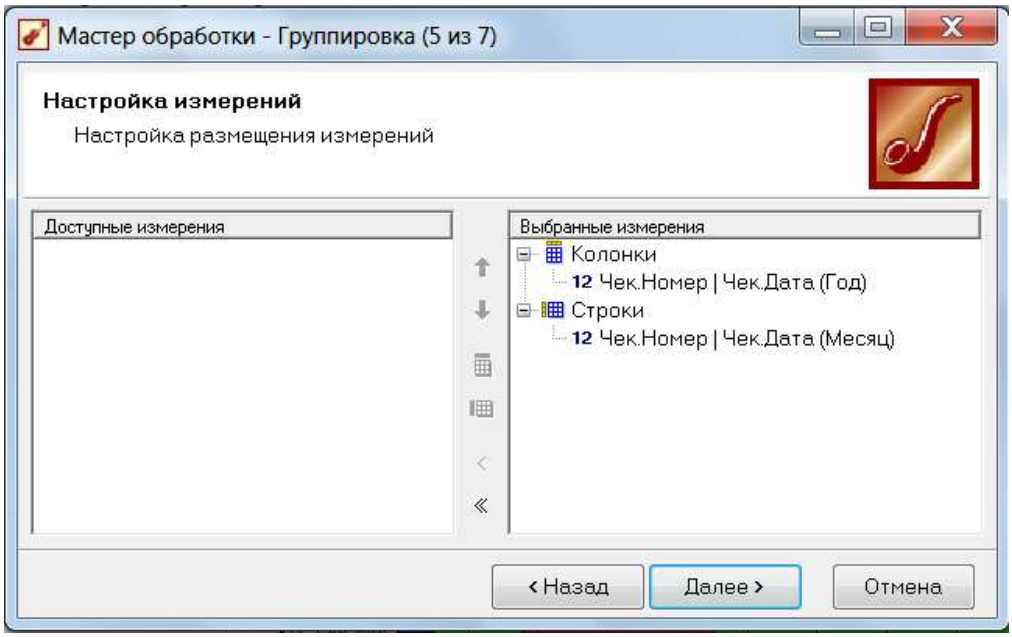
Для этого нужно в сценарии **«Преобразование даты и времени»** выбрать «Год» и «Месяц» отдельно;



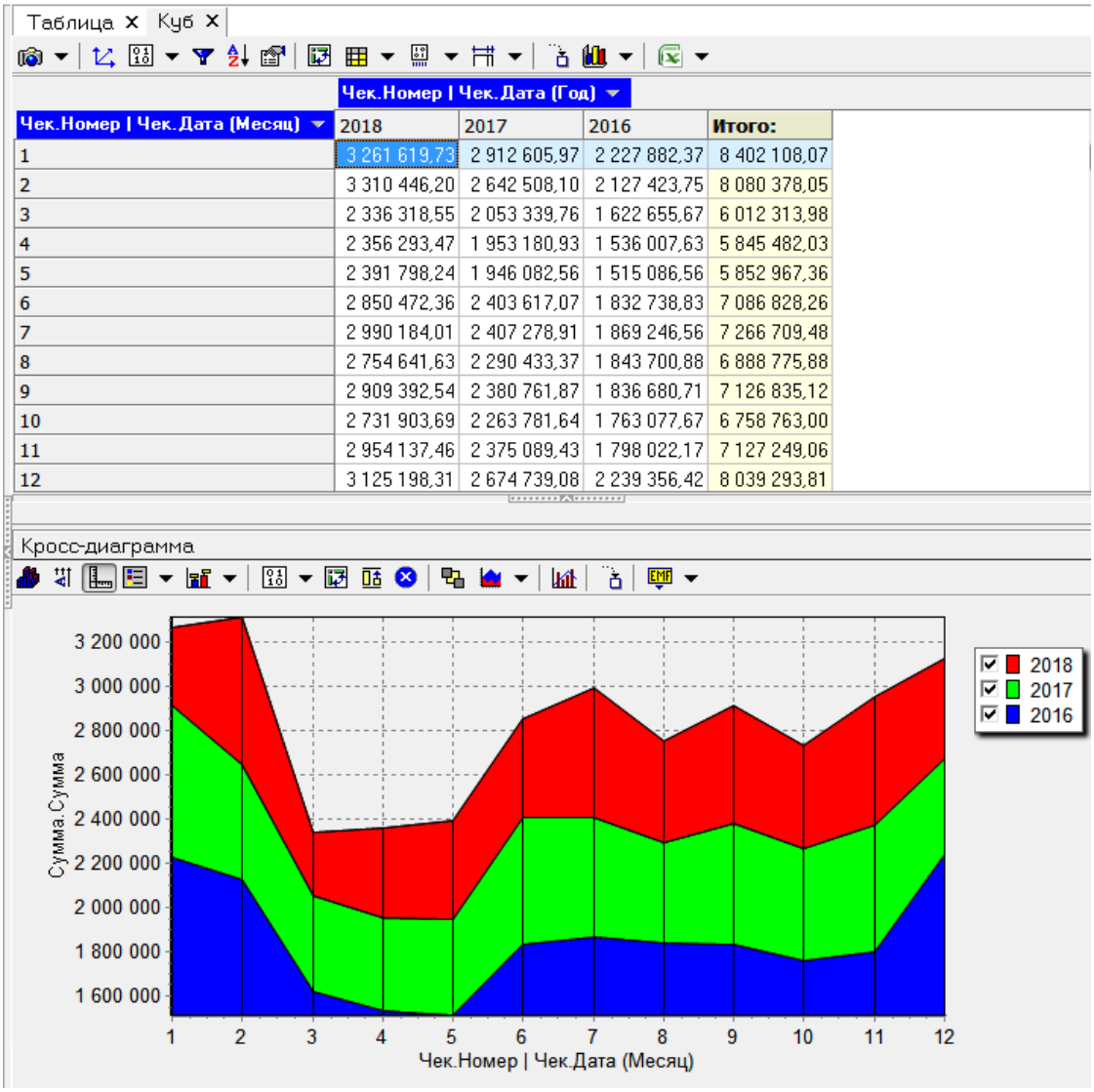
затем в сценарии **«Группировка»** задать измерения «Дата(Год)» и «Дата(Месяц)» и факт «Сумма»;



в качестве визуализатора следует выбрать **«Куб»,** настроить колонки и строки куба.

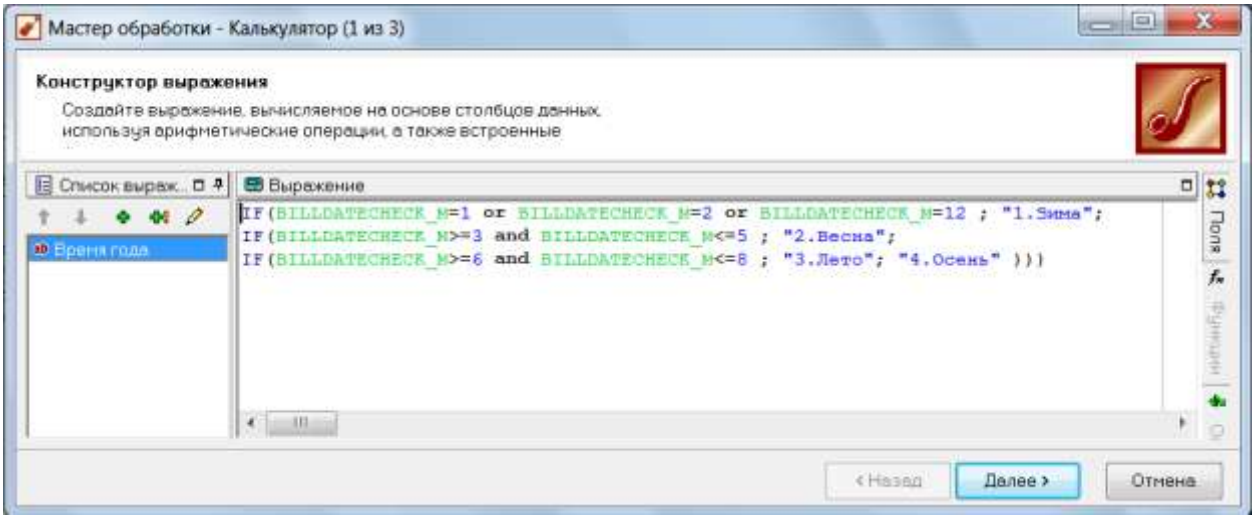


Получится следующий куб, в котором годы нужно отсортировать по убыванию, и можно строить кросс-диаграмму:



А что делать в том случае, если нас интересует объем продаж по сезонам (зима, весна, лето, осень)? Такое преобразование даты в программе не предусмотрено, но его легко можно реализовать самим.

Снова будем использовать обработчик **«Калькулятор».**

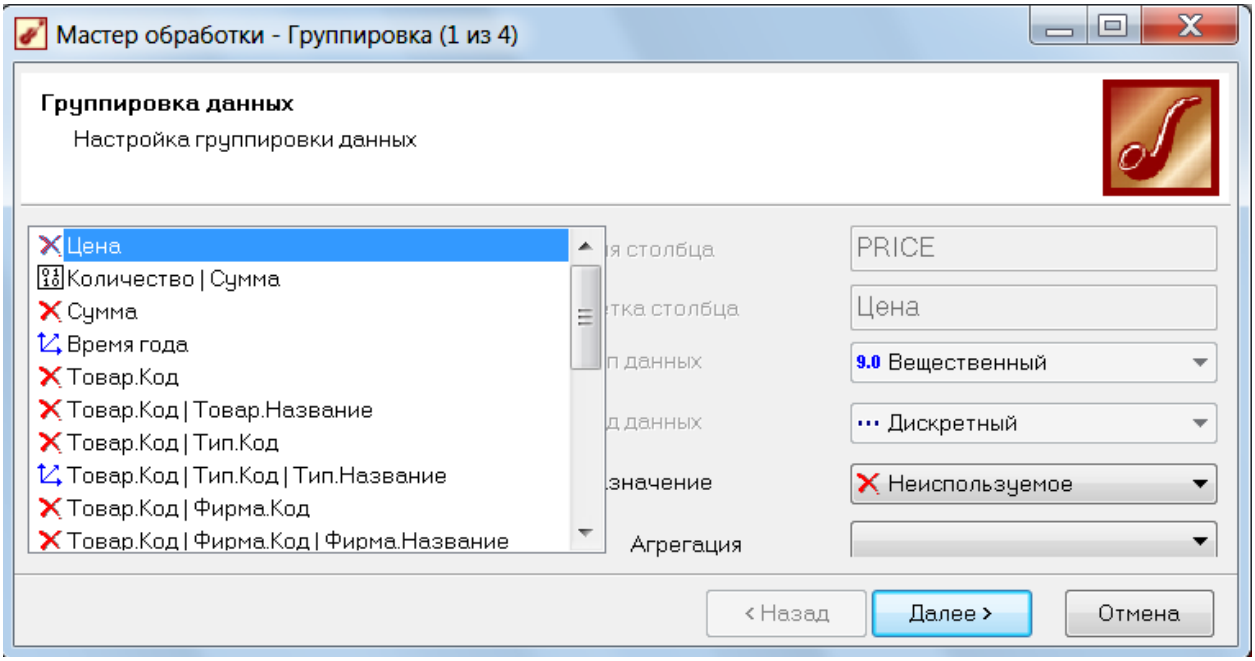


Назовем и сам обработчик, и вычисляемое выражение **«Время года»**. Задаем формулу, в которой в зависимости от номера месяца вычисляется сезон: 1.зима, 2.весна, 3.лето, 4.осень. Сезоны пришлось перенумеровать, потому что иначе они на диаграмме будут упорядочены по алфавиту, а это не соответствует их реальному порядку.

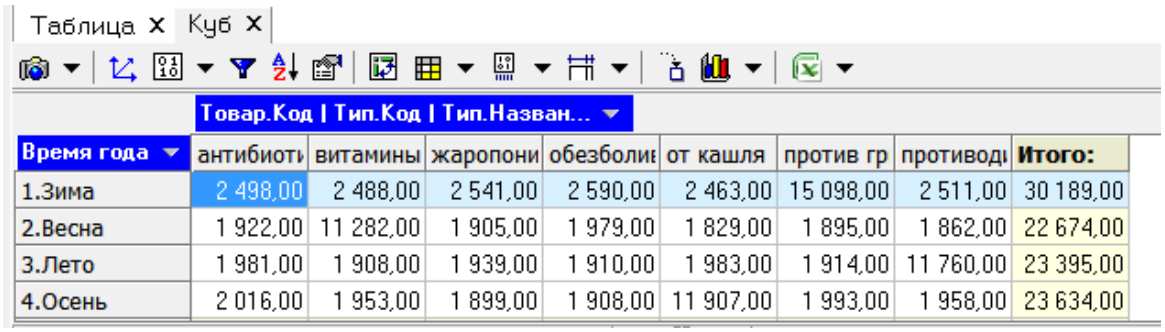
Функция IF представляет собой компактный вариант условного оператора ЕСЛИ … ТО … ИНАЧЕ … и имеет формат:

```
IF(условие, значение_если_истина, значение_если_ложь)
```

Теперь осталось выполнить группировку данных. Оставим здесь измерения **Время года** и **Тип.Наименование** и факт **Количество.Сумма.**



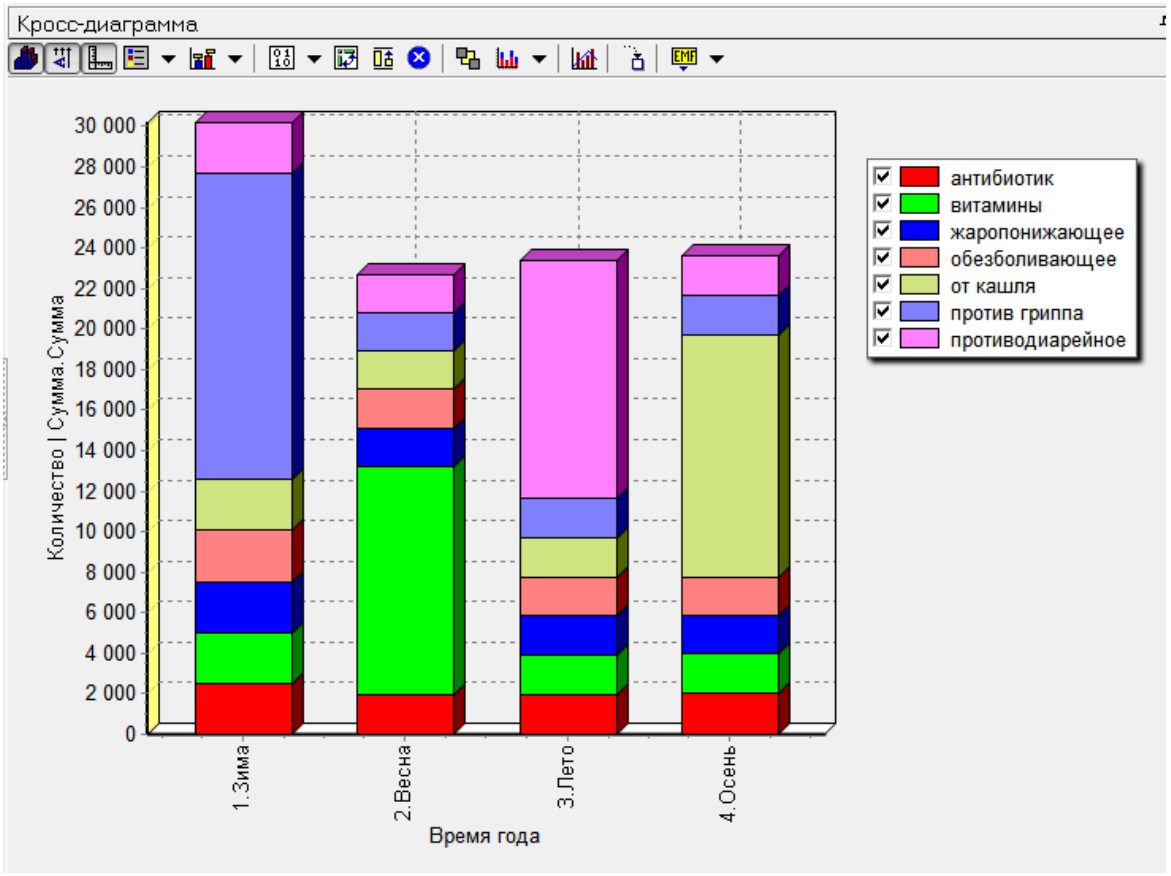
Зададим визуализатор «Куб»:



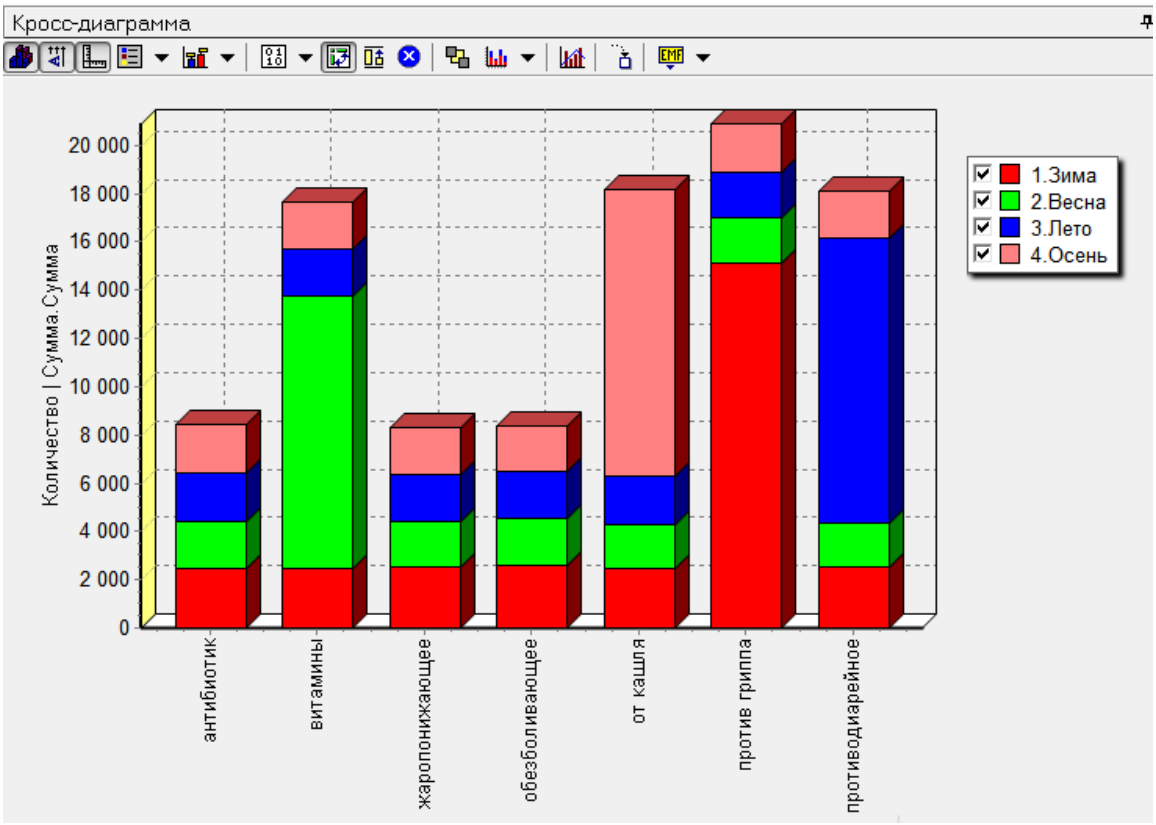
Время года	антибиоти	витамины	жаропони	обезболи	от кашля	против гр	противоди	Итого:
1.Зима	2 498,00	2 488,00	2 541,00	2 590,00	2 463,00	15 098,00	2 511,00	30 189,00
2.Весна	1 922,00	11 282,00	1 905,00	1 979,00	1 829,00	1 895,00	1 862,00	22 674,00
3.Лето	1 981,00	1 908,00	1 939,00	1 910,00	1 983,00	1 914,00	11 760,00	23 395,00
4.Осень	2 016,00	1 953,00	1 899,00	1 908,00	11 907,00	1 993,00	1 958,00	23 634,00

а затем построим диаграмму. Обратите внимание, что наиболее наглядным в данной ситуации является тип диаграммы «Столбчатая с накоплениями».

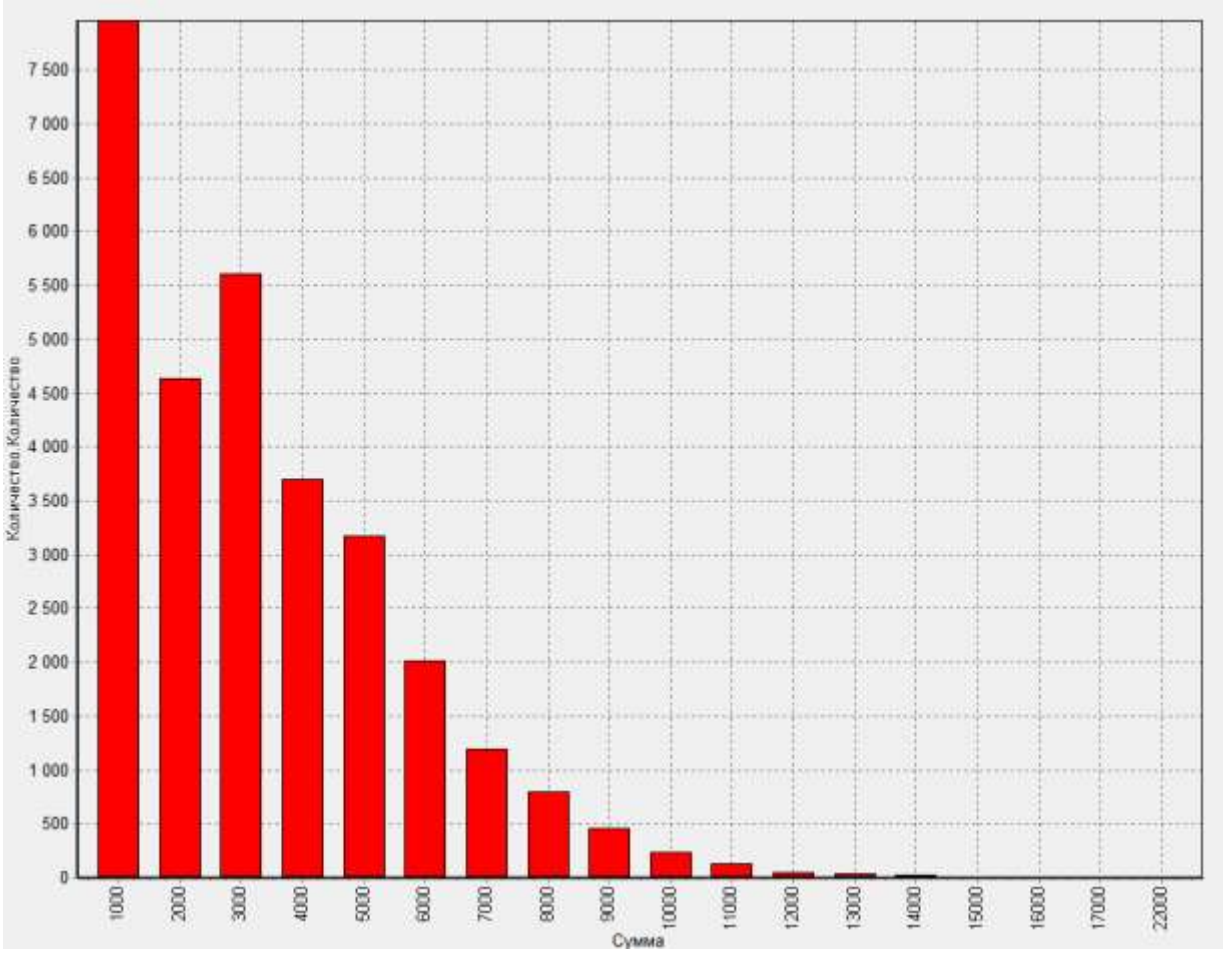
Найдите на диаграмме, какие типы лекарств являются самыми продаваемыми в разные сезоны года.



А вот так эта диаграмма выглядит, если ее транспонировать:



Рассмотрим пример более сложного преобразования данных. Пусть мы хотим проанализировать суммы чеков с помощью примерно такой диаграммы:



Столбцы в этой диаграмме означают, что у нас есть

- примерно 8 тысяч чеков с суммой до 1 тыс. руб.
- примерно 4.5 тысяч чеков с суммой от 1 до 2 тыс. руб.,
- примерно 5.5 тысяч чеков с суммой от 2 до 3 тыс. руб.,
- и т.п.

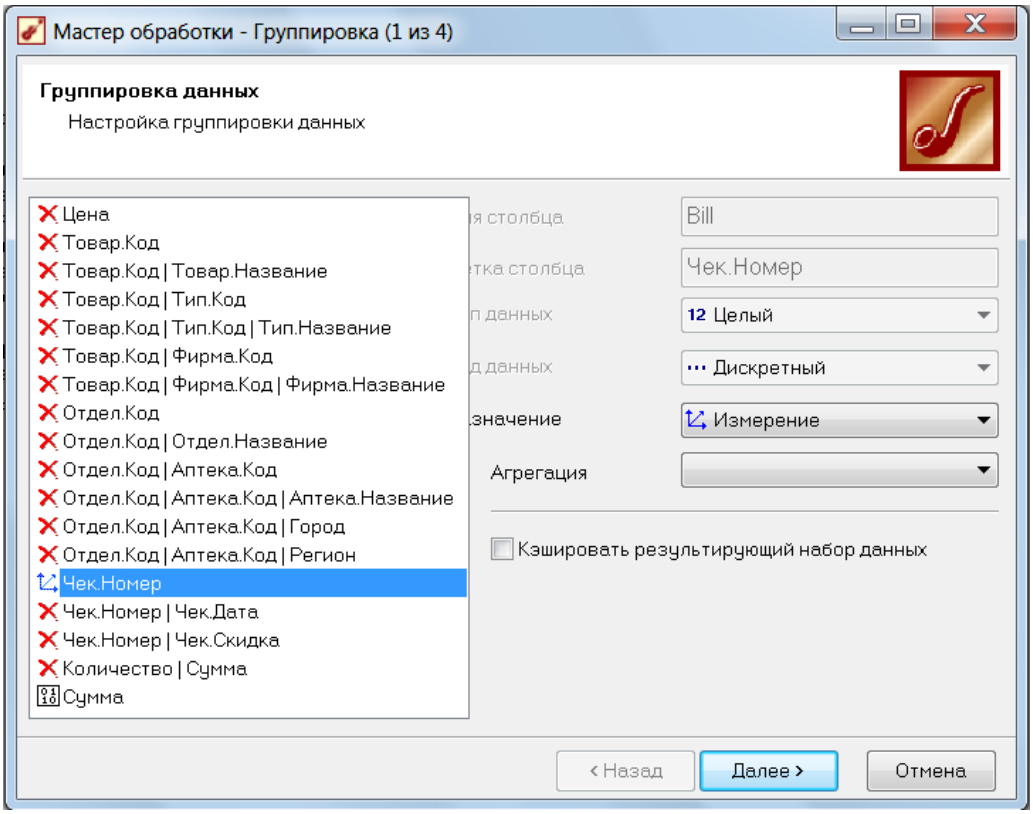
Здесь по горизонтальной оси отложены верхние границы интервалов для суммы чеков, а значения по вертикальной оси означают количество чеков, попавших в каждый интервал.

Как видим, для построения такой диаграммы нам не нужны точные суммы чеков, а лишь информация, в какой интервал попадает каждая сумма.

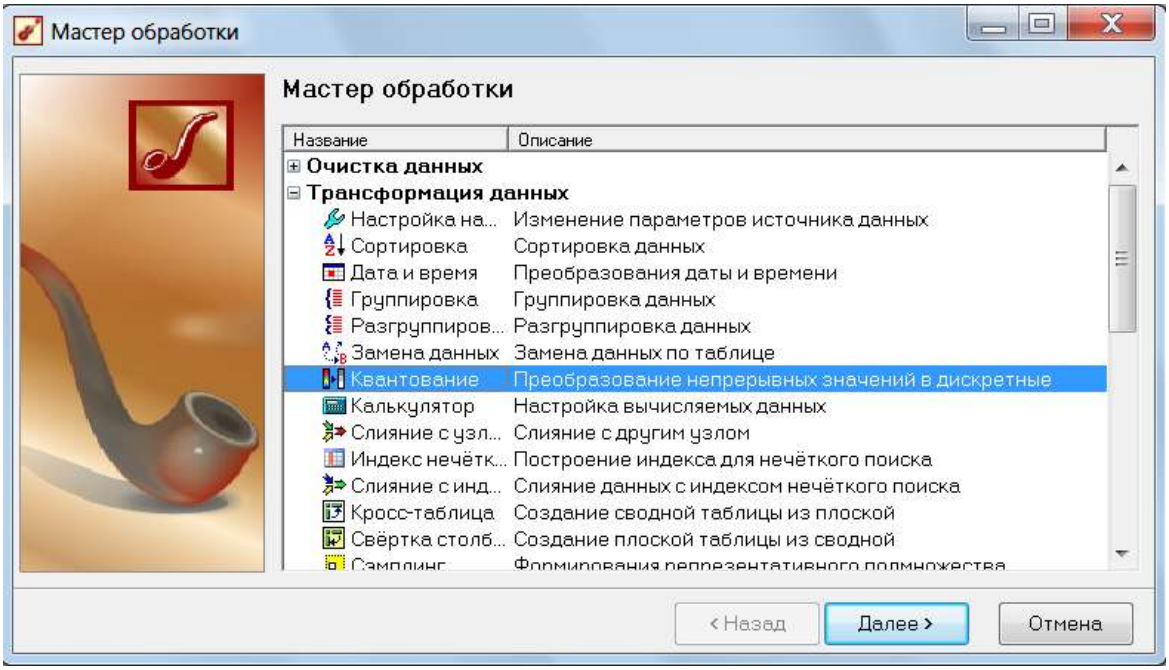
Для построения такой диаграммы попробуем применить обработку **«Квантование»**. Процесс квантования или дискретизации – это замена

точных «непрерывных» значений номерами или метками интервалов, в которые эти значения попадают.

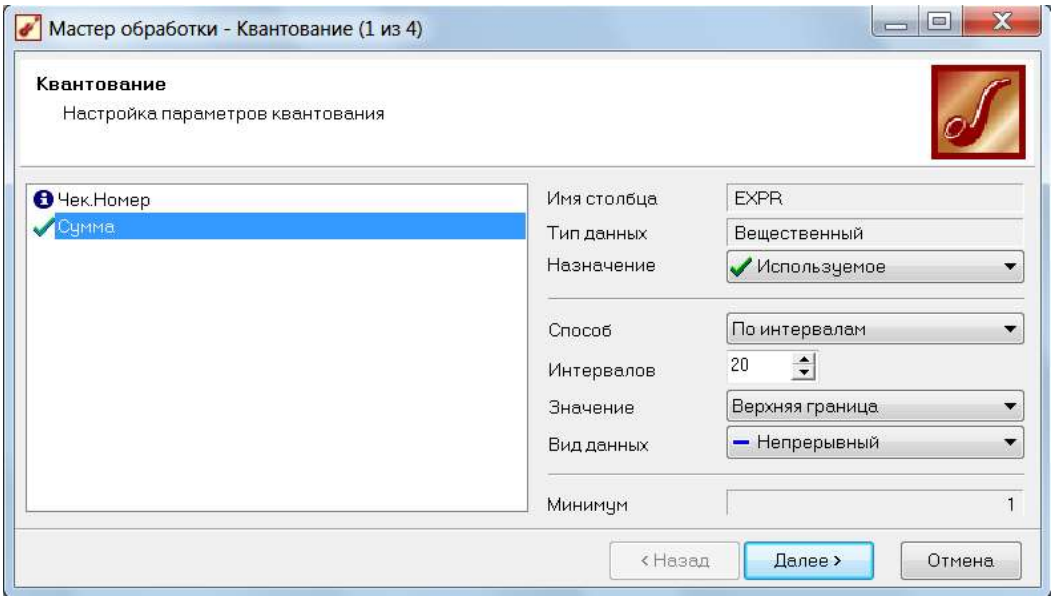
На первом этапе вычислим суммы чеков с помощью обработки «**Группировка**»:



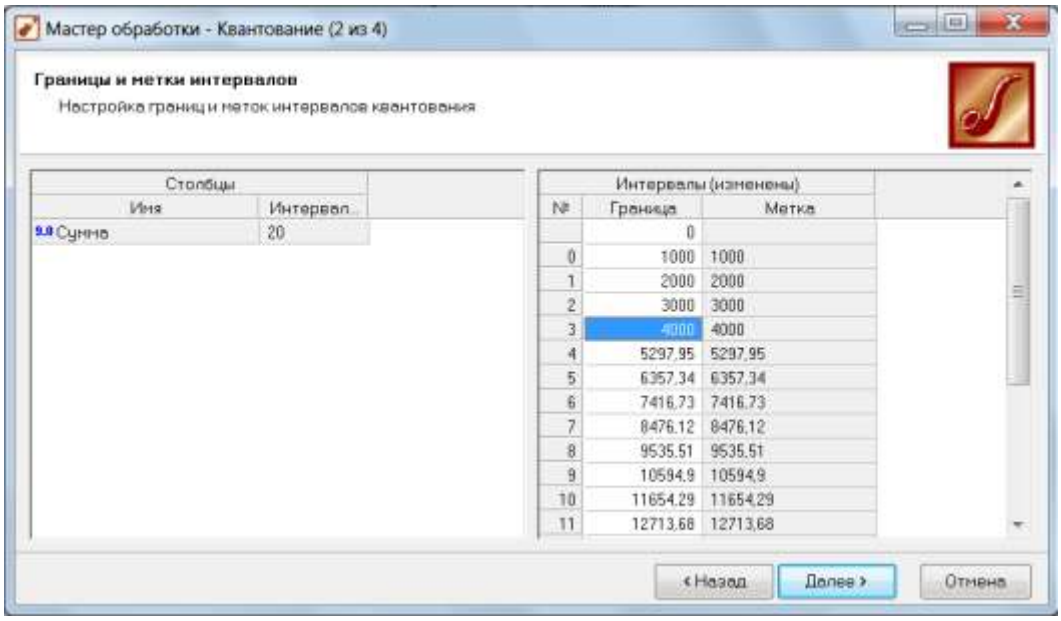
На втором шаге выберем обработку **«Квантование»** из группы **«Трансформация данных»**:



Будем производить квантование по столбцу «**Сумма**», зададим способ **«По интервалам»**, **20** интервалов квантования, в качестве метки интервала зададим его **верхнюю границу**:

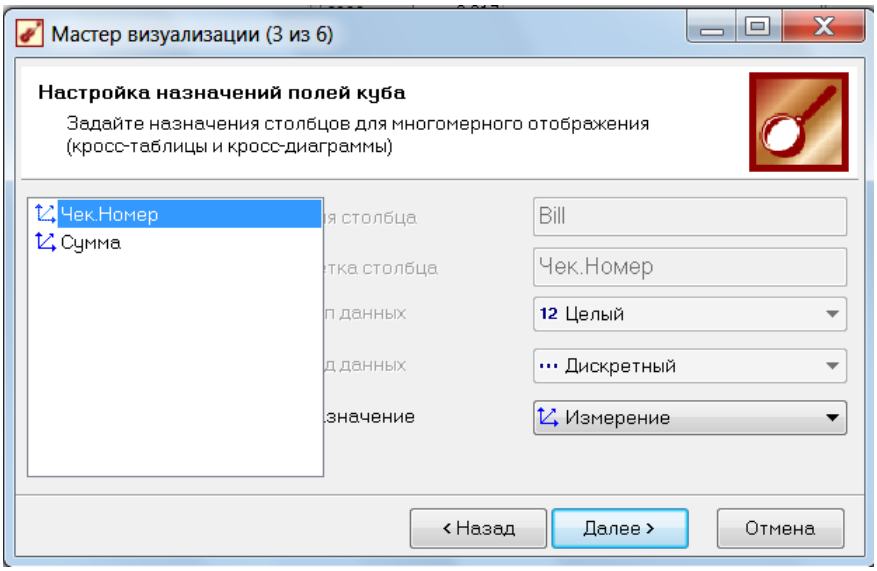


На следующем этапе мастера можно изменить границы интервалов квантования. Округлим их до целых тысяч. Метки интервалов при этом изменятся автоматически. Для последнего интервала назначим границу «с запасом», 22000. Интервалы не обязательно должны быть равными!

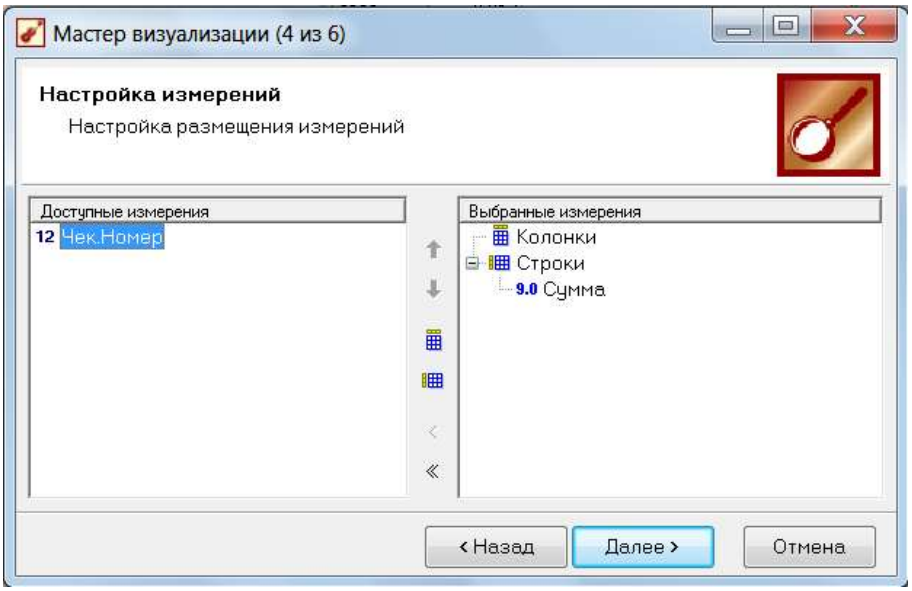


Для вывода результатов квантования настроим визуализатор «Куб». Его можно настроить на 3-м этапе мастера обработки, либо с помощью мастера визуализации.

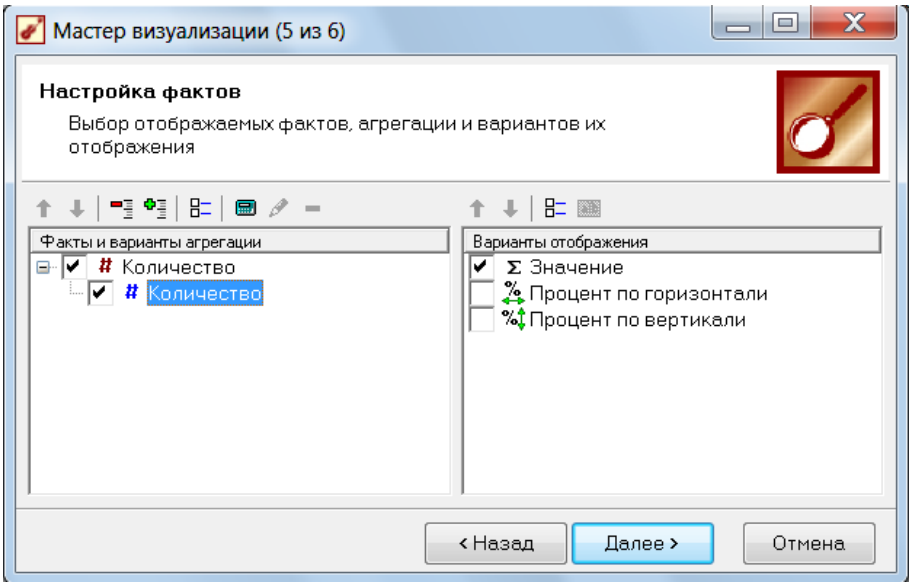
Оба столбца в нашей таблице являются измерениями, явных фактов здесь нет:



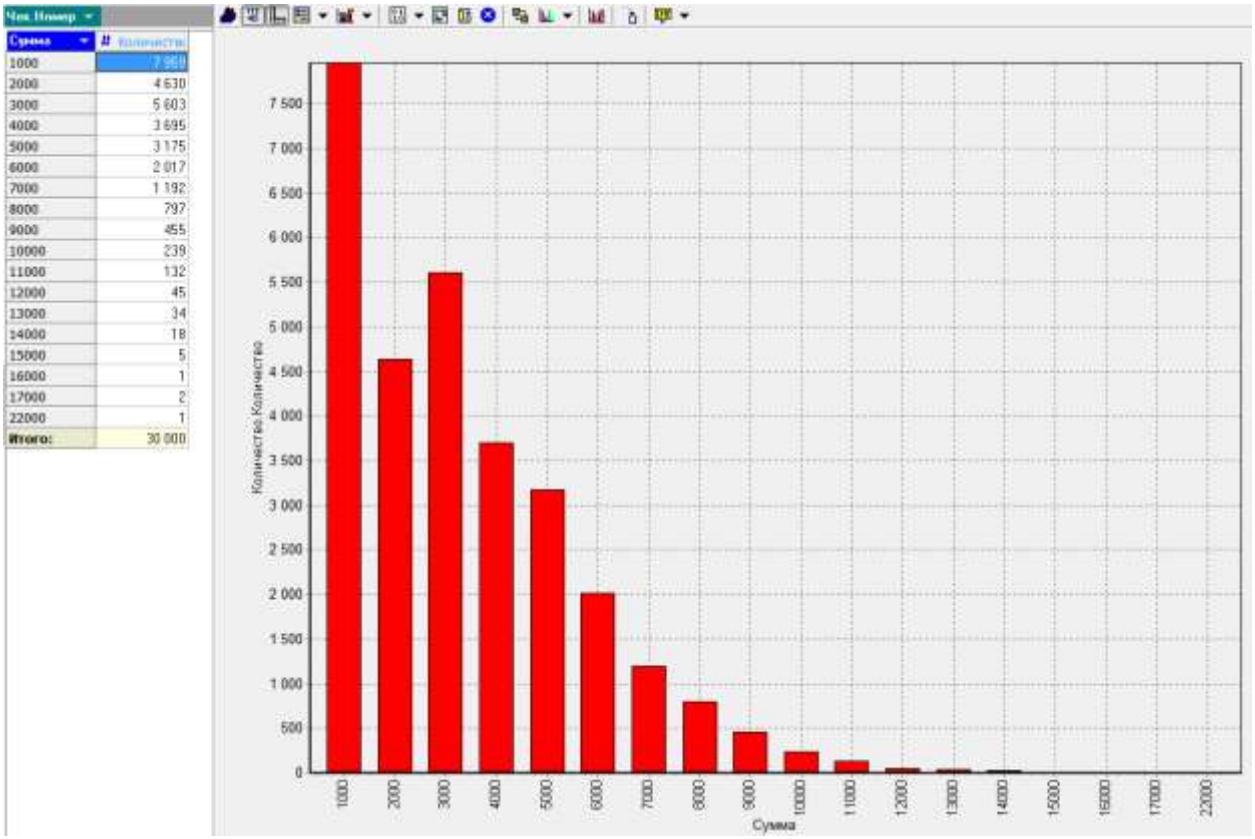
В качестве строк таблицы выберем измерение «Сумма». Напомним, что в этом измерении сейчас заданы метки интервалов.



Агрегирующая функция будет подсчитывать количество строк в группах:



Наконец, по полученному кубу построим диаграмму:



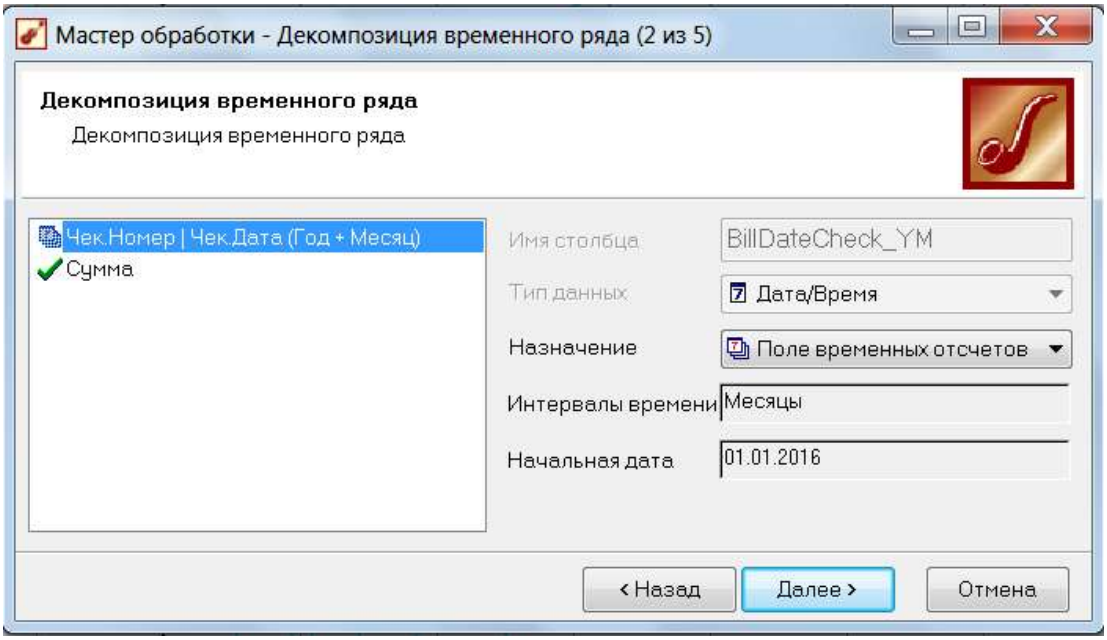
Задание 6в. Примените к вашим данным разнообразные преобразования и визуализаторы. Используйте, как минимум, визуализаторы: **Таблица, Куб, Диаграмма, Детализация** и преобразования: **Калькулятор, Преобразование даты, Фильтр, Группировка, Квантование**. (8 баллов).

Этап 7. Анализ данных в Deductor Studio

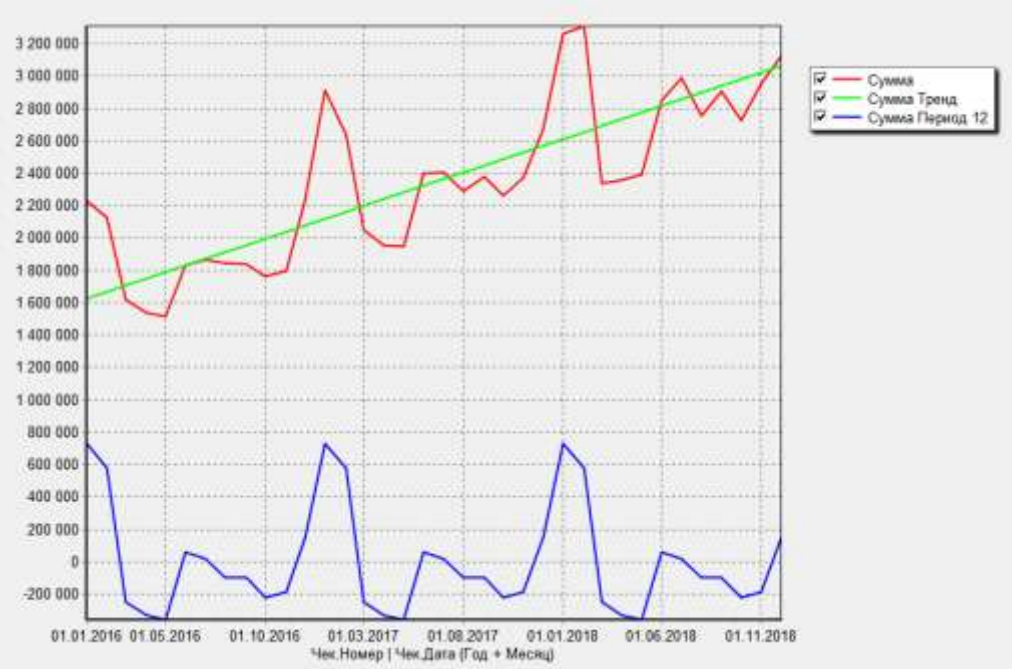
Работа в **Deductor Studio** не ограничивается манипуляциями с OLAP-кубами. Самые интересные и сложные возможности заключаются в применении методов интеллектуального анализа данных (Data mining). Рассмотрим несколько простых примеров.

Пример 1: Выделение сезонной компоненты и тренда из временного ряда.

Мы сгенерировали данные таким образом, что в них явно прослеживается тенденция и существуют сезонные колебания. Для отдельного отображения тренда и сезонной компоненты в секции мастера обработки «**Data mining**» есть обработка «**Декомпозиция временного ряда**». За основу возьмем узел, в котором из даты уже выделены Год+Месяц и произведена группировка. В соответствующем окне настройки есть всего лишь два столбца. Столбец даты имеет тип **«Поле временных отсчетов»:**



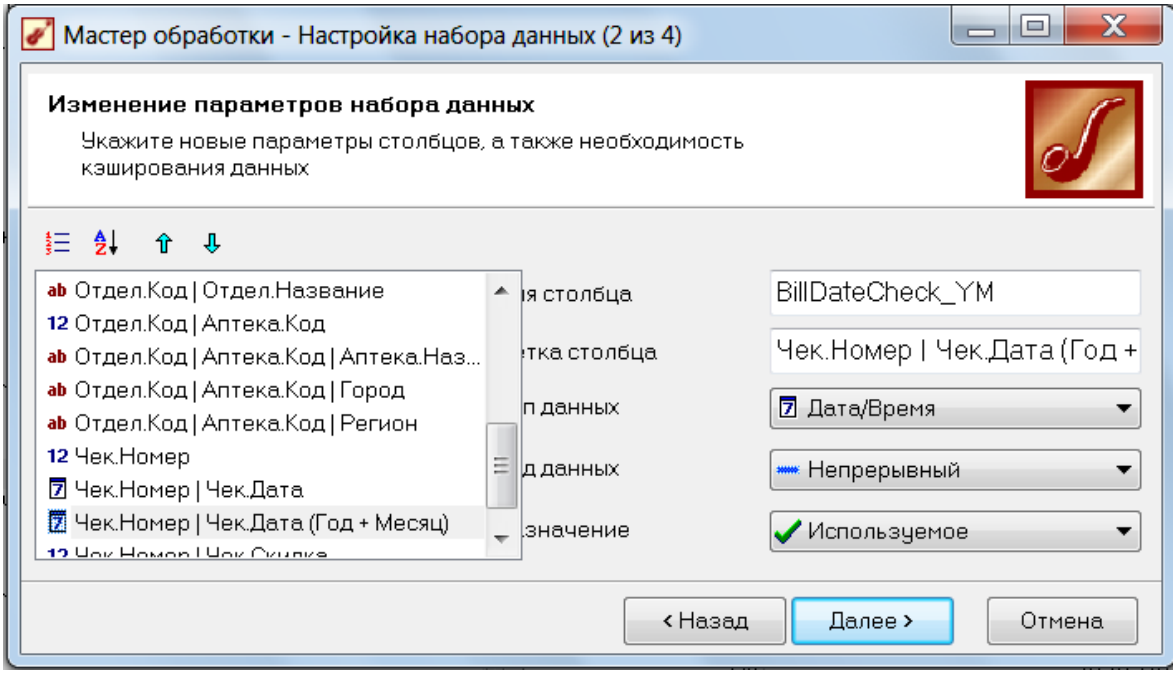
Получим следующий график, на котором выделены растущий тренд (линия зеленого цвета) и сезонная компонента (синяя ломаная).



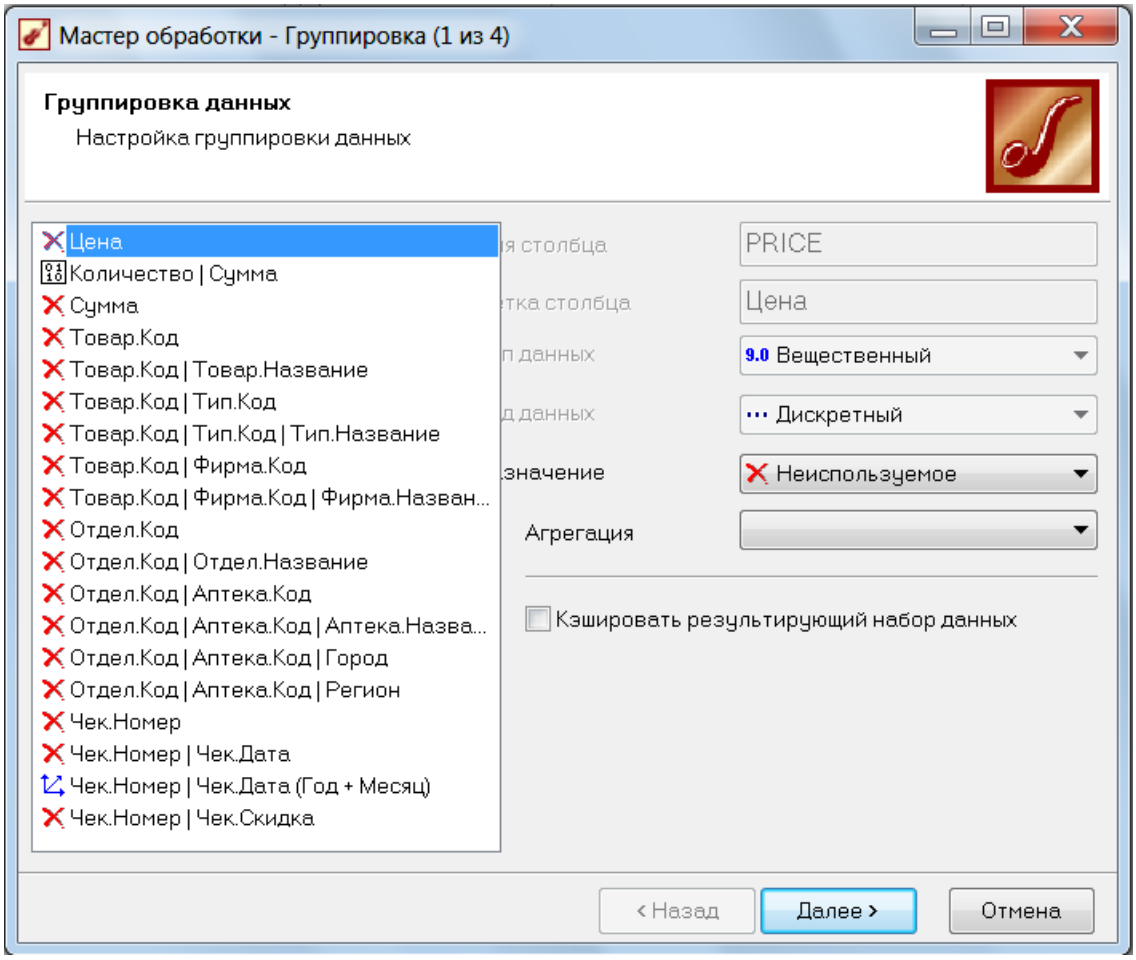
Пример 2: Построим прогноз продаж лекарств типа «жаропонижающее» с горизонтом прогноза 2 месяца. За основу возьмем узел, в котором из даты уже выделены Год+Месяц.

1 шаг. Создадим обработку «**Фильтр**» с условием фильтрования Тип.Наименование=”жаропонижающее”.

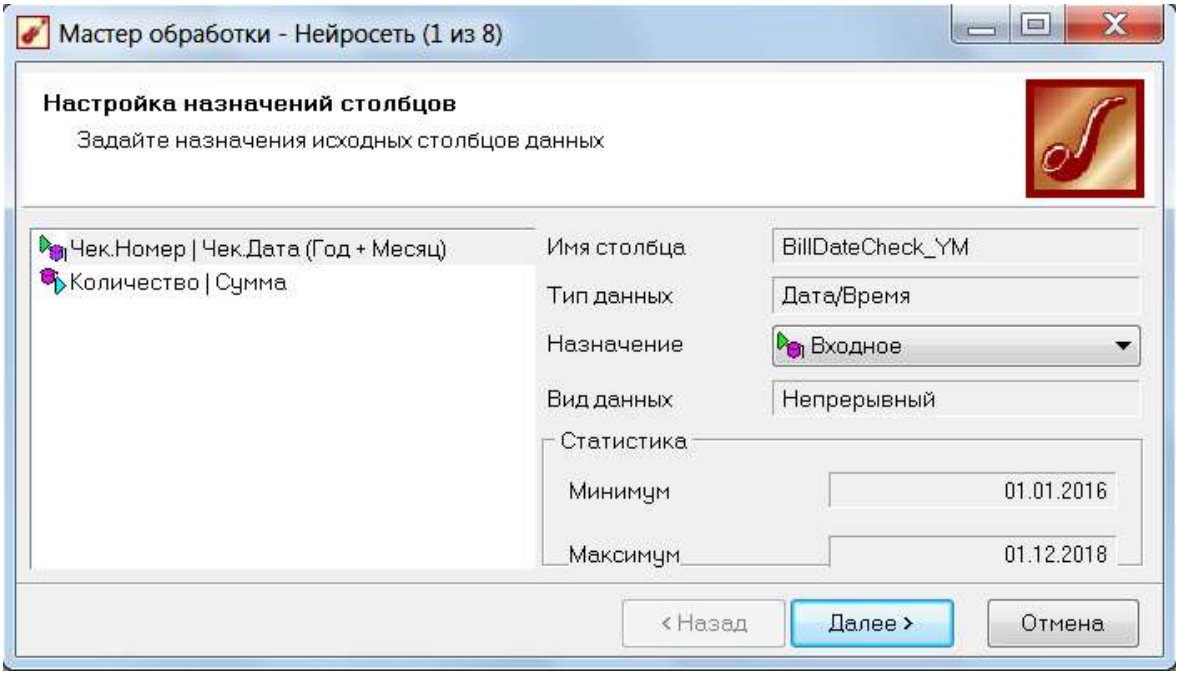
Примечание. Если столбец **Чек.Дата(Год+Месяц)** имеет Вид данных **«Дискретный»**, то следует сначала создать обработку «**Настройка набора данных**» и для столбца **Чек.Дата(Год+Месяц)** назначить Вид данных «**Непрерывный**».



2 шаг. Создадим группировку по измерению **Чек.Дата(Год+Месяц)** с фактами **Количество**.

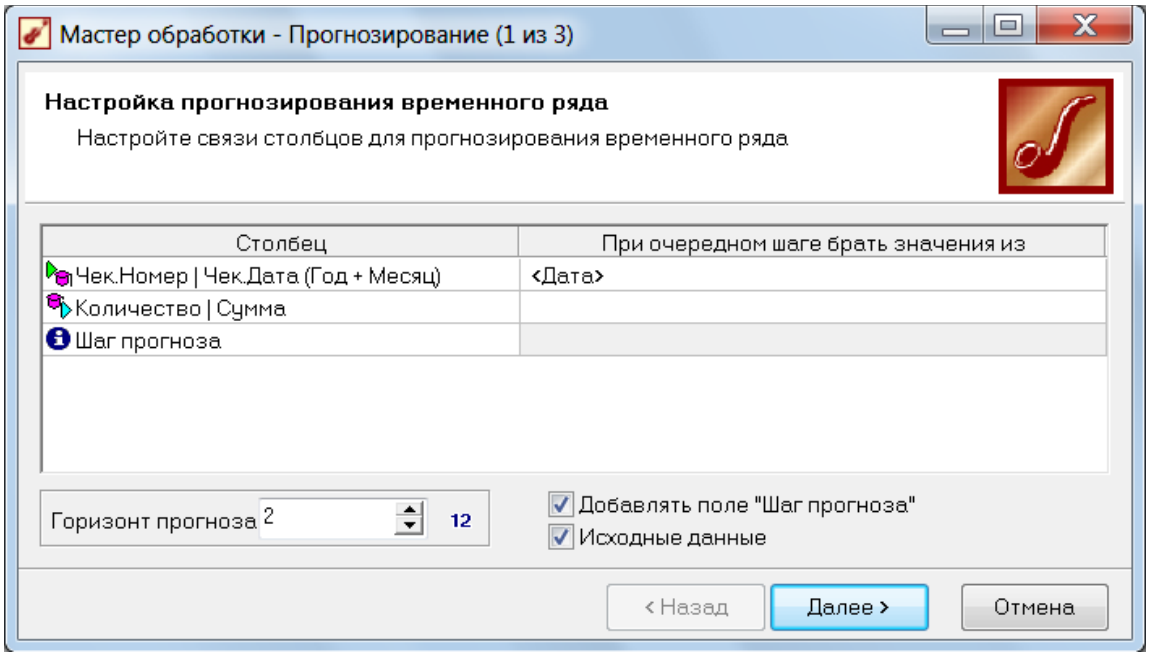


3 шаг. Создадим обработку «**Нейронная сеть**» со следующими входным и выходным столбцами:

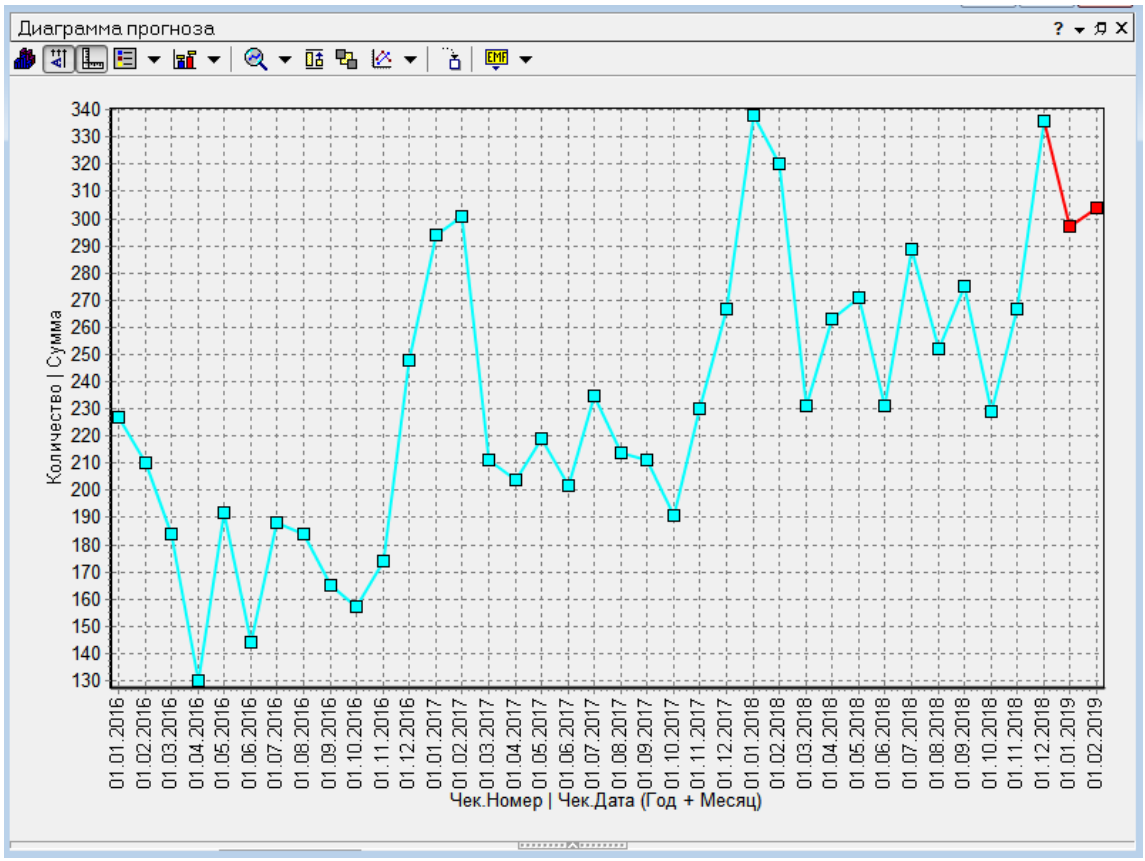


Остальные параметры в этой обработке оставим без изменений.

4 шаг. Наконец, создадим обработку **«Прогнозирование»** (из секции **Data mining**) с горизонтом прогноза в 2 месяца:



Получим следующий прогноз.



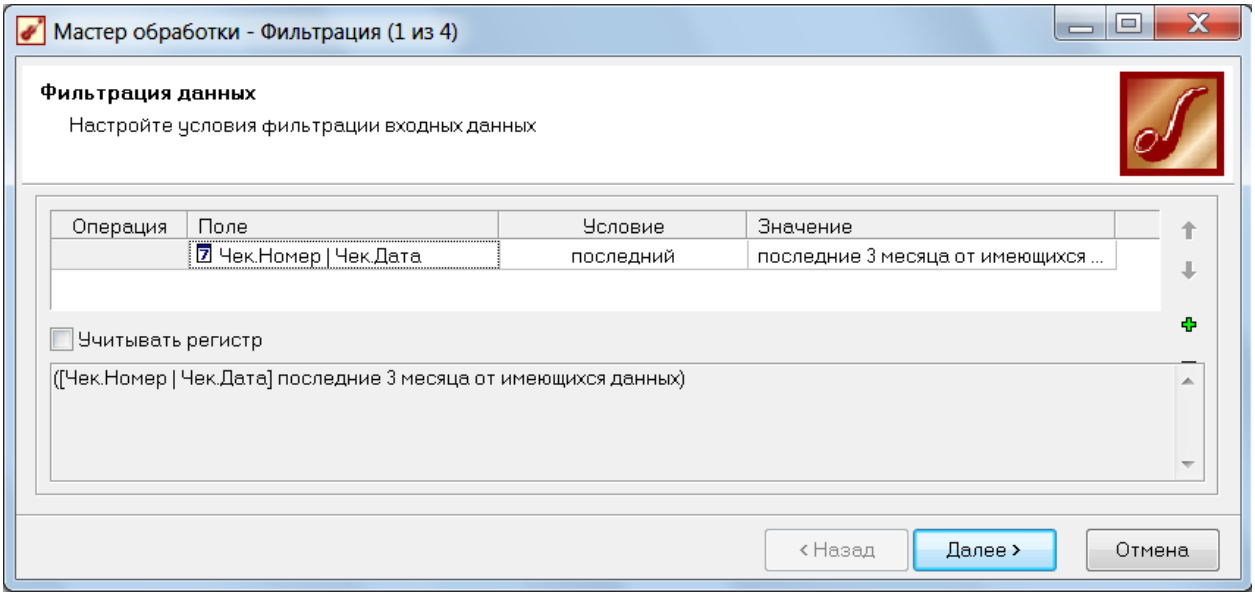
Пример 3: ABC-анализ [1]. В аналитической отчетности очень полезными часто оказываются ABC и XYZ анализ – распределение объектов, например товаров, клиентов, поставщиков, по доходности и стабильности продаж.

Результатом ABC- анализа является группировка объектов по трем категориям:

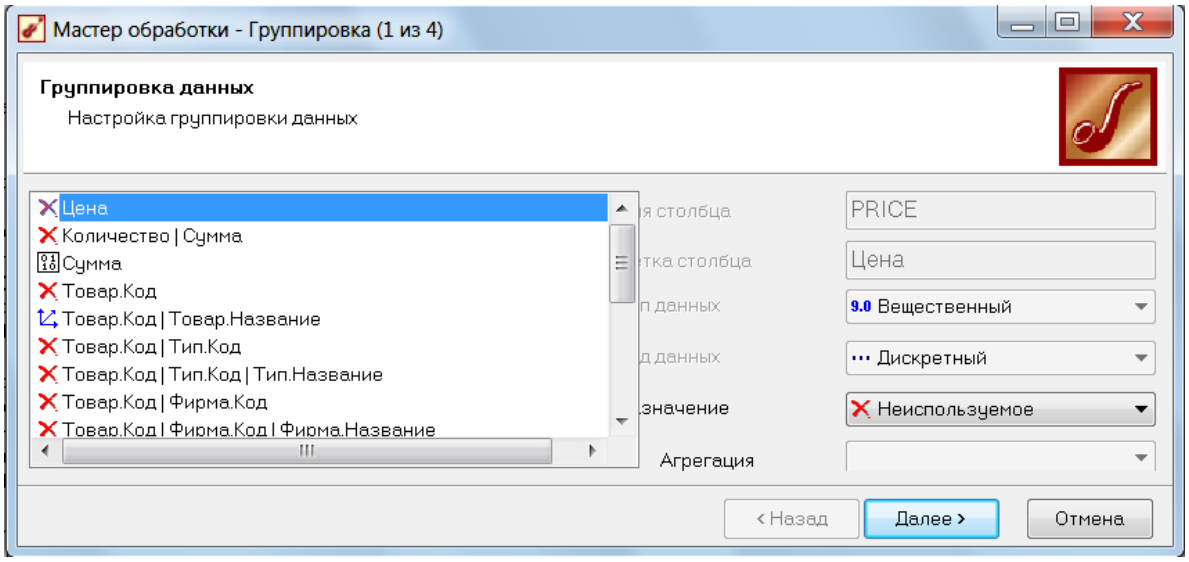
- категория A - наиболее ценные объекты, сумма долей с накопительным итогом которых составляет первые 75 % общей суммы;
- категория B - промежуточные объекты, следующие за группой A, сумма долей с накопительным итогом которых составляет от 75 до 90 % общей суммы;
- категория C - оставшиеся наименее ценные объекты, сумма долей с накопительным итогом которых составляет от 90 до 100 % обшей суммы.

Итак, будем анализировать объем продаж товаров по их названиям, за последние 3 месяца. За основу возьмем узел, в котором рассчитана сумма продаж в качестве отдельного столбца.

1 шаг. Создадим обработку «**Фильтр**» - отфильтруем данные за последние 3 месяца:

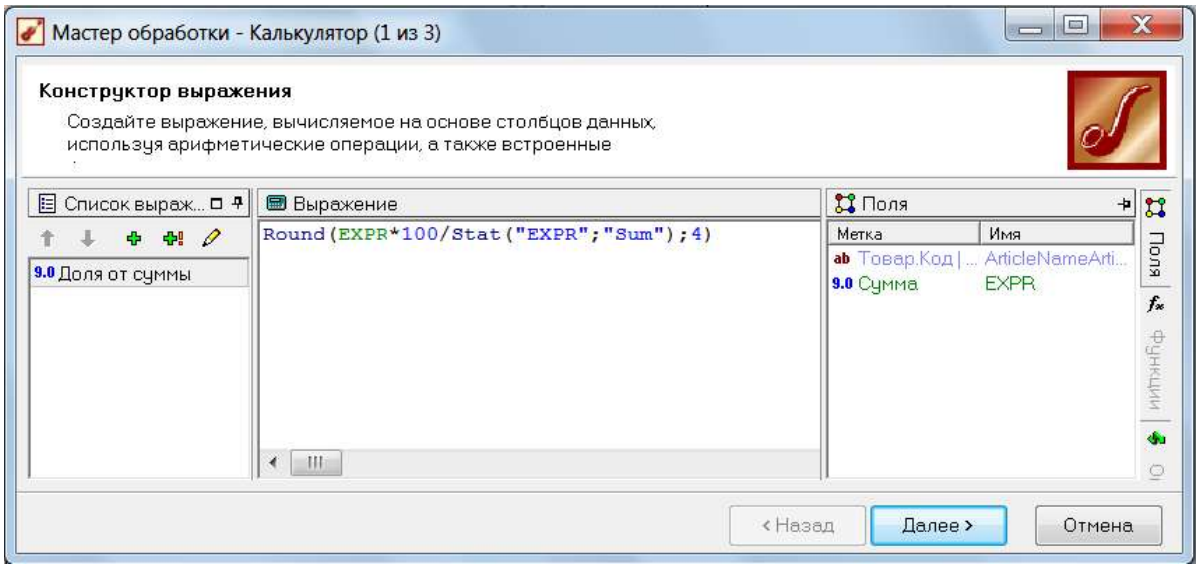


2 шаг. Выполним группировку по нужным нам столбцам:



3 шаг. Создадим обработку «**Сортировка**» по убыванию суммы.

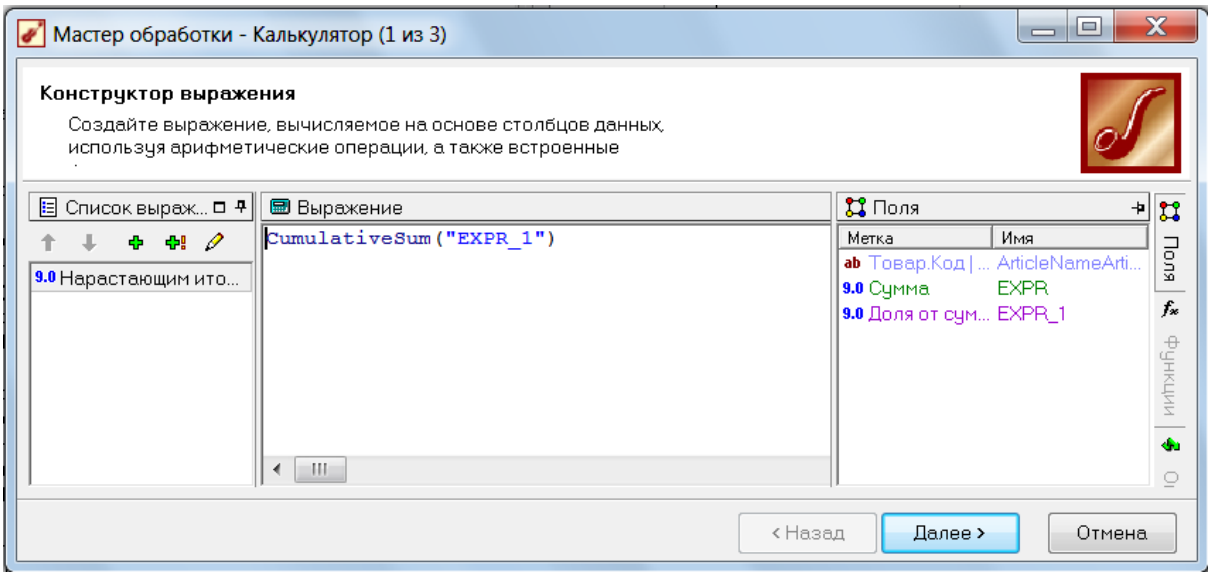
4 шаг. Создадим обработку «**Калькулятор**», который подсчитывает долю текущего товара от общей суммы.



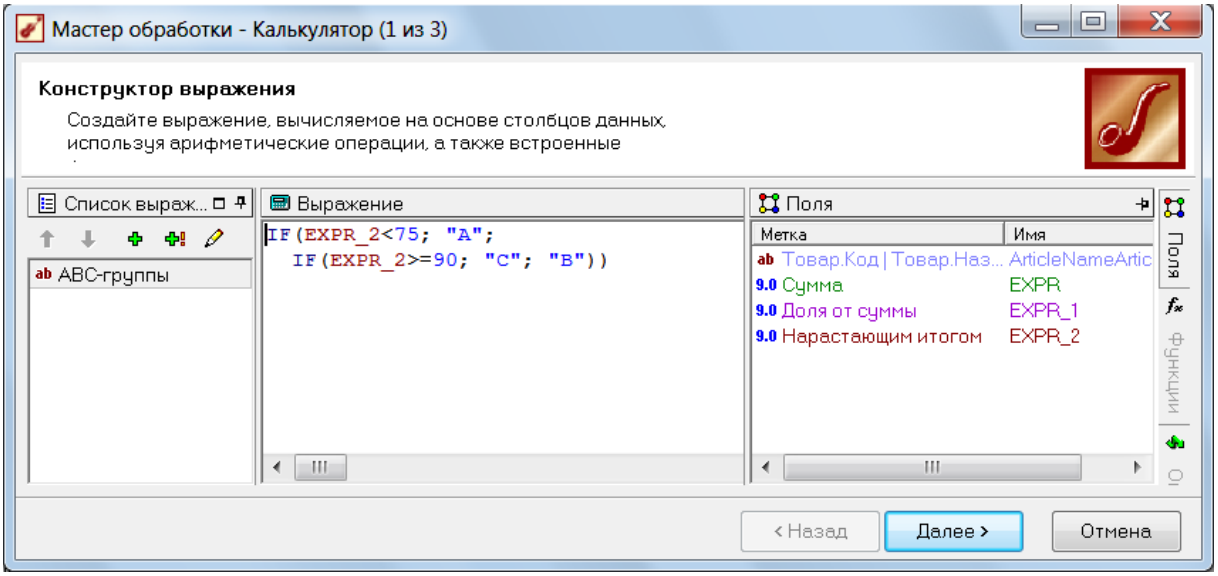
Здесь функция **ROUND**(*число, количество_знаков*) округляет *число* до *количества_знаков* после запятой.

Функция **STAT** вычисляет итоговое значение для всего столбца, имя которого задано в первом аргументе функции. Второй аргумент означает тип итогового значения – количество, сумму, среднее и т.п.

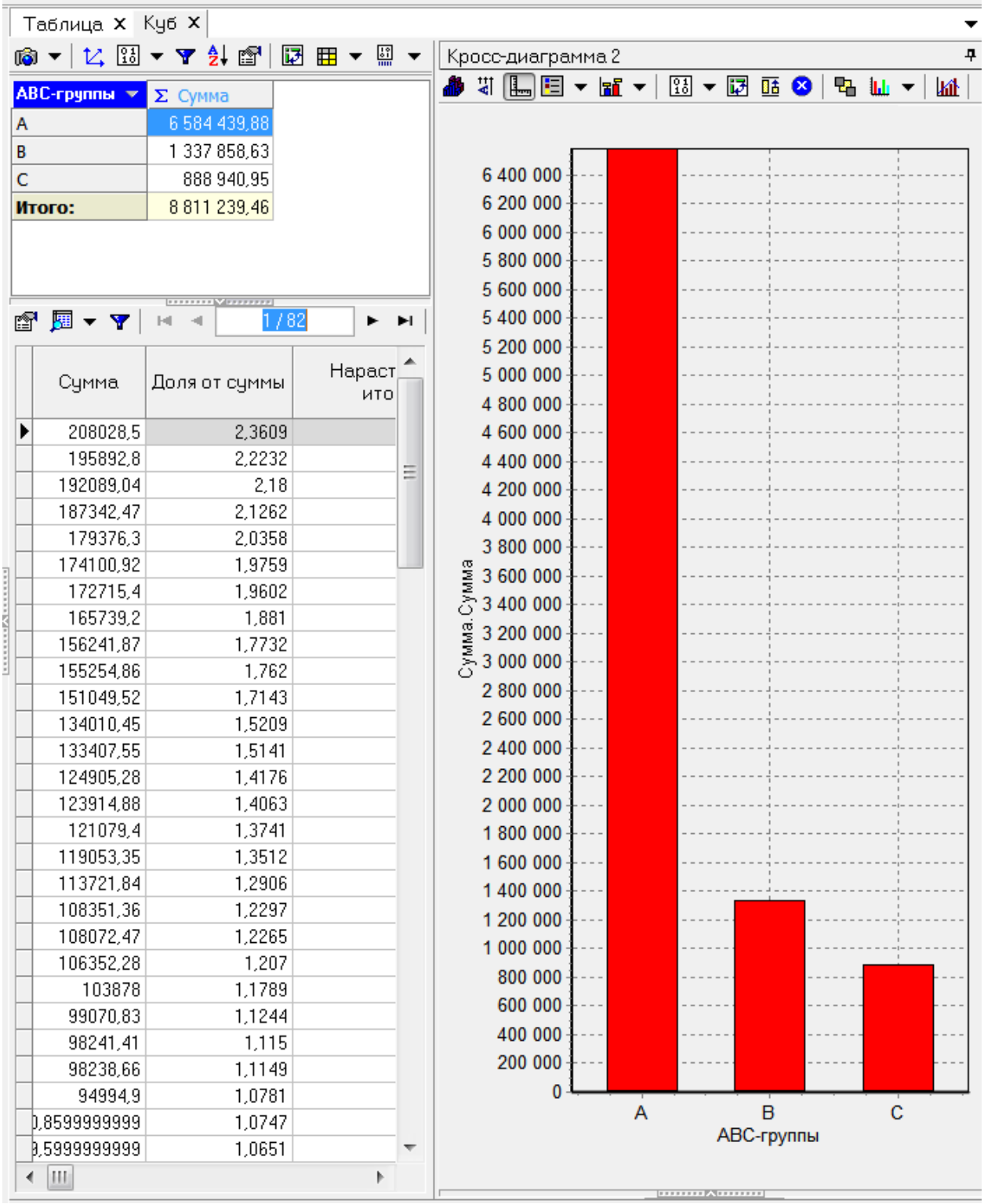
5 шаг. Создадим обработку «**Калькулятор**», который подсчитывает долю текущего товара от общей суммы нарастающим итогом с помощью функции **CumulativeSum**.



6 шаг. Наконец, создадим обработку «**Калькулятор**», который распределяет товары по группам.



Далее выводим полученные данные в виде куба, у которого строками являются ABC-группы. Параллельно выводим диаграмму, показывающую размеры групп, и таблицу с детальной информацией. Теперь мы можем оценить, какие товары приносят нам больше выручки из общего объема, а какие, наоборот, являются невыгодными с этой точки зрения. Точно так же можно оценивать и другие объекты – покупателей, отделы, фирмы и т.п.



Другие примеры практического применения методов Data Mining можно найти в Части 2 книги: **Паклин Н.Б., Орешков В.И. Бизнес-аналитика: от данных к знаниям. - СПб.: Питер. - 2013. - 704 с.**

Задание 7. Примените к вашим данным все рассмотренные выше методы анализа данных. (10 баллов).

Литература

1. Паклин Н.Б., Орешков В.И. **Бизнес-аналитика: от данных к знаниям**. - СПб.: Питер. - 2013. - 704 с.
2. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. **Технологии анализа данных. Data Mining, Visual Mining, Text Mining, OLAP**. - СПб.: БХВ-Петербург. - 2007. - 384 с.
3. Елманова Н., Федоров А. **Введение в OLAP-технологии Microsoft**. - М.: Диалог-МИФИ. - 2002. - 272 с.

Интернет-источники

1. Аналитическая платформа **Loginom**: https://loginom.ru/
2. **Loginom** — руководство пользователя: https://help.loginom.ru/userguide/
3. **Deductor Academic**: https://basegroup.ru/deductor/description