ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

ФАКУЛЬТЕТ ЭКОНОМИЧЕСКИХ НАУК

ДЕПАРТАМЕНТ ТЕОРЕТИЧЕСКОЙ ЭКОНОМИКИ

ФИНАЛЬНЫЙ ПРОЕКТ

по дисциплине «Язык SQL»

Выполнила:

студентка группы БЭК1810

Бузаева Софья Михайловна

Москва — 2021

Описание предметной области

Процесс глобализации привел к тому, что покупатели слали более придирчивыми к покупкам. С помощью интернета и развитой сети доставки они могут приобрести любую вещь от любого бренда. Так происходит на основных рынках. Однако существуют области торговли, которым не свойственна массовость из-за их специфичности. Одной из таких областей является ортопедическая обувь. Стереотип о том, что ортопедическая обувь обязана быть удобной и некрасивой (для бабушек) среди людей присутствует до сих пор. К сожалению, есть категория людей, которая не может себе позволить другой обуви из-за особенностей своего тела (инвалиды). Поэтому подобного рода магазины будут нужны всегда. Да, возможно их немного, но они присутствуют на рынке обуви.

Таким образом, область проектируемой базы данных становится ясна. Создаваемая база данных будет опираться на принципы работы реальной ортопедической компании «ОРТОМОДА». Сразу стоит отметить, что база данных будет сильно упрощена по своей структуре и будет содержать далеко не все области, присутствующие в компании. Поскольку включение всего этого раздуло бы базу до невероятных размеров.

Проектируемая база данных будет отражать основные механизмы работы обувной компании со встроенными специфичными элементами из ортопедической области. Как и положено, в компании будет сфера, связанная с закупками, и сфера, связанная с продажами.

<u>Принцип работы БД следующий:</u>

- Обувная компания предоставляет 2 типа услуг: продажа обуви, которую она закупила у других поставщиков, индивидуальный пошив обуви (особенно он свойственен тем, у кого структура ног не позволяет носить стандартные модели), который производит сама. Примечание: особенности работы собственного производства обуви в БД затрагиваться не будут.
- При закупке и продаже обуви будем считать, что отгрузка, доставка, приемка товаров происходит мгновенно и в нашей БД никак не отражена. Также добавим предпосылку, что конкретная модель обуви конкретного размера может храниться только на одном складе. Таким образом, вопрос о списании обуви со склада отпадает (т.е. при покупке ил продаже кол-во пар обуви будет просто списываться из того места, где она хранится – не будет дилемы, откуда конкретно ее списывать).
- Поставщиками компании могут быть как частные лица (физические), так и компании (юридические).
- Компания занимается только розничной продажей.
- Клиенты могут выбирать из каталога любую модель и заказывать либо ее, либо эту же модель, но индивидуального пошива.
- Клиенты компании делятся на 3 типа: госконтрактники, платники и платники с возмещением.
    - Госконтрактник – клиент, который без всякой оплаты заказывает обувь. Оплату же производит государство сразу за всех таких клиентов (госконтрактом). Как правило, такими клиентами являются люди с инвалидностью.
    - Платники – обычные клиенты, которые захотели приобрести обувь в данном магазине.

    - Платник с возмещением – клиент, который покупает обувь на свои деньги, а после при предоставлении чека в ФСС получает денежное возмещение на суму покупки.
- В компании есть сотрудники, которые работают в подразделениях компании. Для упрощения предполагается, что сотрудник любой должности может совершать продажи и закупки. Однако сотрудник обязательно должен быть привязан к филиалу компании. Ограничение на кол-во сотрудников в филиалах нет.
- Подразделения компании могут быть нескольких типов: офис, магазин, склад, пункт выдачи. Опять же для упрощения БД предполагается, что продажи могут осуществлять из любого отделения компании, ровно, как и хранение обуви могут быть в любом из них.
- Номенклатура представляет собой довольно обширный выбор моделей, которые делятся на несколько категорий. Здесь стоит опять сделать уточнение, что в данной БД будут представлены не все разделы обуви, которые присутствуют в ортопедической компании. Я остановилась на следующих:
    - Для кого: женская, мужская и детская обувь
    - Сезон года: зима, лето, весна-осень
    - Цвет (тоже ограничила): черный, коричневый, синий, белый и красный

  Каждая модель обуви имеет свой код и артикул. Артикул для одинаковых моделей один и то же не зависимо от размера пары. Код уникальный.
  Одна модель может быть представлена только одним цветом.
- Оплата продажи может осуществляться любым возможным способом (ограничений нет). Однако есть ограничения на способ доставки: самовывоз, курьер, СДЭК и почта России.
- Состояние заказа отслеживается (его статус). Он может быть один из следующих:
    - доставка
    - выдан на руки
    - в производстве
    - готов
    - на примерке
    - на складе
- Области, связанные с бухгалтерией, будут по максимуму обходиться и упрощаться.

Проанализировав все вышеперечисленное, в проектируемой БД получилось 8 таблиц:
1. filials (подразделения компании)
2. workers (работники)
3. suppliers (поставщики)
4. clients (клиенты)
5. nomenclature (номенклатура - каталог)
6. stock (склад, складские остатки)
7. supplies (закупки
8. sales (продажи)

Преимущества базы данных

Компании подобного рода компилируют много сведений из разных областей ежедневно. Это и продажи, и закупки, склады, доставки и многое другое. Функционировать без единой системы хранения данных просто невозможно. В создание БД позволить поддерживать целостность данных и предоставить быстрый доступ в необходимой информации, которая будет являться

наиболее актуальной (не устаревшей). Также написание предварительно функций и триггеров поможет сотрудникам компании, не знакомым с языков программирования и формирования запросов, быстро и легко получать всю необходимую им информацию и вносить изменения в данные.

Примеры требований к БД

- Выполнение поиска по подразделениям
- Выполнение поиска по сотрудникам
- Выполнение поиска по клиентам
- Выполнение поиска в номенклатуре
- Пересчет кол-ва пар обуви при совершении акта продажи
- Пересчет кол-ва пар обуви при совершении акта закупки

Требование к данным

1. Подразделения (filials)
Каждое подразделение компании необходимо будет идентифицировать однозначно. Также для каждого подразделения должны быть указаны данные о нем.

Предположительные столбцы в таблице:
- id филиала (уникальный номер)
- тип филиала (офис, магазин, склад, пункт выдачи)
- город, где филиал расположен
- телефонный номер
- адрес

Стоит отметить, что здесь присутствует некоторое дублирование данные, а именно, информация о городе, в котором расположен филиал будет содержатся и в поле «адрес», и в поле «город». Причиной такого дублирование стало то, что адреса могу быть указаны совершенно по-разному и работать с таким полем может быть затруднительно. Города же помогу при большой сети магазинов быстрее ориентироваться в местах расположения подразделений.

2. Сотрудники (workers)
Каждого сотрудника необходимо будет идентифицировать однозначно. Также для оформления трудового договора требуются определенные сведения о сотруднике (в проектируемой БД они не будут представлены в полном формате, поскольку это лишь атрибуты, которые увеличивают таблицы, но не привносят особую пользу в работу БД).

Предположительные столбцы в таблице:
- id сотрудника(уникальный номер)
- ФИО
- должность, которую он занимает
- подразделение, в котором он работает
- серия и номер паспорта (их комбинация должна быть уникальной)
- дата рождения

- пол
- телефонный номер

Также накладывается ограничения, что сотрудник не может работать в подразделении, которого нет в БД.

3. Поставщик(suppliers)

Каждого поставщика необходимо будет идентифицировать однозначно. Также для каждого поставщика будут указаны некоторые данные о нем. Данные нужны для лучшего понимания, с кем компания имеет дело.

Предположительные столбцы в таблице:

- id поставщика(уникальный номер)
- название (ФИО, если поставщик является физическим лицом, название компании – если юридическим)
- тип поставщика (физ. лицо или юр. лицо)
- телефонный номер
- адрес

4. Клиенты (clients)

Каждого сотрудника необходимо будет идентифицировать однозначно. Поскольку компания довольно тесно взаимодействует со своими клиентами (в особенности, если это инвалиды или клиенты, которые решили шить обувь индивидуально), для каждого из них заводится специальная карточка, в которой указываются данные клиенты и впоследствии будет записываться его история покупок. В историю могут входит визиты к врачу (в некоторых подразделениях компании расположены врачебные кабинеты, где предоставляются услуги консультации со специалистами в области ортопедии – в нашей БД это никак не отражено), покупки, история общения с менеджерами и прочие взаимодействия клиента с компанией.

Предположительные столбцы в таблице:

- id клиента (уникальный номер)
- ФИО
- тип клиента (госконтрактники, платники и платники с возмещением)
- серия и номер паспорта (их комбинация должна быть уникальной)
- СНИЛС
- дата рождения
- пол
- телефонный номер
- адрес

5. Номенклатура (nomenclature)

Все доступные модели обуви содержатся в номенклатуре, которая предоставляется клиентам для выбора моделей обуви. Каждая модель обуви будет иметь свой уникальный номер (здесь имеется в виду, что для каждой позиции номера будут уникальными; другими словами, одна и та же модель, но с разными размерами будет иметь разные номера). Также модели обуви

будет присвоен артикул, который будет единым для всех размеров обуви одинаковой модели. Это сделано для того, чтобы продавец мог сказать клиенту, какие размеры обуви конкретной модели есть в наличии.

Все модели будут подразделять на разные секции, о которых было написано ранее. В номенклатуре будет также указана цена, по которой товар будет продаваться клиентам, и закупочная цена. В связи с тем, что при создании данной БД акцент делался не на финансовой сфере, закупочная цена модели обуви в данной таблице будет каждый раз обновлять при новой закупке. Т.е. в таблице будет указана последняя закупочная цена по данной модели. Также формула наценки на закупочную цену для получения продажной никак не отражена в БД. Более того, нет ограничений на соотношение цен. Т.е. чисто гипотетически закупочная цена может быть выше продажной. Эта сфера представляет определенный интерес, на которой можно создать БД, но я решила не углубляться в финансовые вопросы и с акцентироваться на общих принципах работы компании.

Предположительные столбцы в таблице:

- id конкретной модели (уникальный номер)
- артикул модели
- название
- секция (для кого) женская, мужская и детская обувь
- сезон года: зима, лето, весна-осень
- цвет: черный, коричневый, синий, белый и красный
- цена для продажи
- закупочная цена
- размер
- бренд (название бренда не обязано совпадать с названием поставщика, особенно это относится в ИП)

6. Склад, складские остатки, сток (stock)

Данная таблица предназначения для хранения информации о кол-ве товаров в подразделениях компании. Об основных ограничениях и предположениях, наложенных на данную таблицу, говорилось выше. Самое важное – одна конкретная позиция в номенклатуре (id модели) хранится в одном подразделении (не имеет значение, где оно расположено).

Предположительные столбцы в таблице:

- id конкретной модели (уникальный номер)
- кол-во товара
- id подразделения

Накладывается ограничение, что на складе могут быть только модели обуви, указанные в номенклатуре и хранится они могут только в подразделениях, информация о которых есть в таблице filials.

7. Закупки, поставки (supplies)

В таблице «закупки» я постаралась отразить наиболее важные характеристики, которые указываются в закупочных документах. Т.е. хоть и сильно упрощенно приблизиться в реальности.

Предположительные поля, которые будут указываться при каждой закупке расписаны ниже. Добавлю к ним несколько уточнений.

Все закупки будут единичными. Т.е. в одно строке таблицы может содержаться только информация об одной конкретной закупке, в которой закупили одну конкретную модель (уникальный id модели обуви). Если у одного поставщика сразу закупается 2 и более модели. То в данную таблицу такая сделка будет вноситься как 2 и более отдельные строчки (по одной для каждой модели обуви).

Дата совершения закупки будет отражать время (дату и время вплоть до секунд), когда данная закупка (сделка) была совершена. Тут не учитывается весь процесс заключения сделки: договор о цене, оформление документов, доставка и прочие накладные расходы, логистика и тд.

Также в данной таблице никак не будет отражаться пункт доставки партии. Таким образом. Будет предполагаться, что товар доставляется сразу в подразделение, в котором хранится данный вид обуви (id конкретной модели). Предположительно, здесь в работу вступил триггер, который при добавлении новой строки в таблицу закупок будет обновлять информацию в таблице stock (пересчитывать кол-во товара на складе).

Предположительные столбцы в таблице:

- id конкретной закупки (уникальный номер)
- дата совершения закупки
- id поставщика
- общая сумма закупки
- цена за одну штуку
- кол-во закупленных пар обуви
- id закупаемой модели обуви
- id работника, который оформил сделку

Ограничения, которые будут наложены на данную таблицу:

- Закупить товар можно только тот, который уже имеется в номенклатуре
- Оформить сделку может только работник, информация о котором содержится в таблицу workers
- Поставщиком может являться только поставщик, информация о котором содержится в таблицу suppliers
- Общая сумма закупки должна равняться кол-во закупленных пар, умноженное на цену за одну пару

8. Продажи (sales)

В данной таблице будет действовать очень похожий принцип, который был задуман для таблицы supplies. Некоторые моменты и принципы работы были описаны в самом начале. Добавлю еще несколько сведений.

Все продажи будут единичными. Т.е. в одно строке таблицы может содержаться только информация об одной конкретной продаже, в которой была продана одна конкретная модель (уникальный id модели обуви). Если один покупатель купил 2 и более модели. То в данную таблицу такая сделка будет вноситься как 2 и более отдельные строчки (по одной для каждой модели обуви). При этом допускается, что один покупатель может купить более одной пары абсолютно одинаковой обуви (вплоть до размера).

Дата совершения продажи будет отражать время (дату и время вплоть до секунд), когда данная продажа была совершена. Тут не учитывается процесс покупки, доставки (если она требуется). Это только отражается в статусе конкретной закупке.

Также не будет учитываться время внесения средств в кассу. Т.е. не важно, когда был оплачен товар: сразу же или по истечении какого-то времени, или может быть это вообще госконтракт. Опять же это сделано для упрощения и неделания возиться с бухгалтерией.

При внесении новой продажи в таблицу автоматически будет пересчитываться кол-ва пар, оставшихся на складе. Продажа не может совершиться, если на складе отсутствует данная пара обуви.

Предположительные столбцы в таблице:

- id конкретной продажи (уникальный номер)
- дата совершения продажи
- id клиента
- id покупаемой модели обуви
- тип продажи (из каталога либо индивидуальный пошив)
- статус заказа (доставка, выдан на руки, в производстве, готов, на примерке, на складе)
- кол-во закупаемых пар
- форма оплаты (наличными, по карте через кассу, рассрочка, госконтракт и др); ограничения на данную строку налагаться не будут, так как одну и ту же форму оплаты можно назвать разными именами
- способ доставки (самовывоз, курьер, СДЭК и почта России)
- id подразделения, которое совершило продажу (здесь стоит вставить, что подразделение, которое продало обувь, и подразделение, в котором обувь хранится, не обязаны совпадать; также отсюда вытекает, что перемещение пар обуви между подразделениями компании никак не будет отражено в данной БД; т.е. все перемещения и сложности доставки до клиенты будут отражены только в поле «статус заказа»)
- id работника, совершившего продажу

Ограничения, которые будут наложены на данную таблицу:

- Продать товар можно только тот, который уже имеется в номенклатуре
- Совершить продажу может только работник, информация о котором содержится в таблицу workers
- Покупателем может являться только тот человек, информация о котором содержится в таблицу clients (т.е. на него должна быть уже карта клиента компании)
- Продать модно только из того подразделения, информация о котором содержится в таблицу filials

Стоит отметить, что в данной таблице не будет отражена сумма покупки и цена за 1 шт. Данную информацию можно будет получить из номенклатуры. Такое решение было принят при непосредственном создании БД (написании таблиц) с целью создания определенного вида запросов (например, по типу CTE) и написанию функций.

Мы не будем касаться раздела, связанного с возвратом купленной продукции и некачественной партии поставки. Предполагается, что возвраты будут отображаться в таблицах «supplies» и «sales». Другими словами, если покупатель захочет вернуть купленную им обувь, эта операция будет внесена в таблицу закупки. И наоборот, если компания решит вернуть поставщику партию, то данная операция будет внесена в таблицу продажи. Причем предполагается, что цена

будет в первом случае продажная, а во втором закупная. Таким образом, оборот будет расти, но прибыль останется одинаковой.

Подытожим все вышесказанное:

Конечно данную БД можно спроектировать более детально, учитывая мелкие особенности (особенности бухгалтерского учета, временные несостыковки, специфику должностей сотрудников, специфику подразделений, расширить сеть магазинов на российский уровень и учитывать сложности коммуникации между ними и многое другое). Таким образом, у данной БД есть огромный потенциал для ее развития и расширения.

В связи с небольшим объемом БД многие типы переменных (в особенности уникальные номера – id) будут иметь небольшой задел.

Пример возможных запросов

- Посмотреть, какой сотрудник продал больше всего товаров за неделю, месяц, под года и тд
- Посмотреть из какого филиала было продано больше всего пар обуви
- Проранжировать модели обуви по популярности
- Проверить наличие обуви на складе
- Посмотреть, какие размеры доступны для данной модели

Проектирование модели БД

Рис 1. Концептуальная модель базы данных

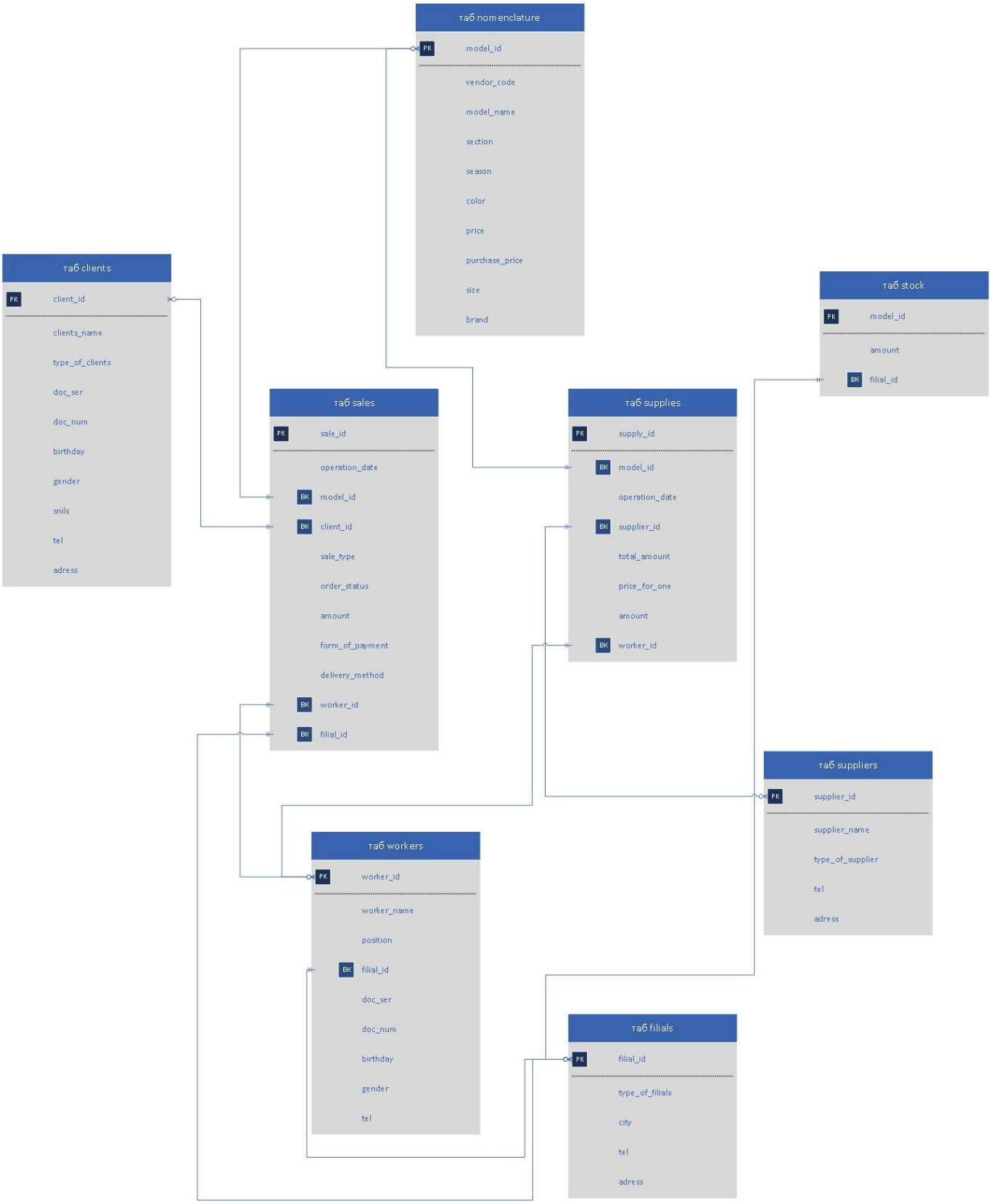

Рис 2. Логическая модель базы данных

Физическая модель базы данных

Таблица «filials»

```
shoes_sale=# \d filials
                                Таблица "public.filials"
     Столбец     |          Тип          | Правило сортировки | Допустимость NULL | По умолчанию
-----------------+-----------------------+--------------------+-------------------+--------------
 filial_id       | character varying(3)  |                    | not null          |
 type_of_filials | character varying(12) |                    | not null          |
 city            | text                  |                    | not null          |
 tel             | text                  |                    | not null          |
 adress          | jsonb                 |                    | not null          |
Индексы:
    "filials_pkey" PRIMARY KEY, btree (filial_id)
Ограничения-проверки:
    "filials_type_of_filials_check" CHECK (type_of_filials::text = ANY (ARRAY['Склад'::character varying, 'Магазин'::characte
r varying, 'Офис'::character varying, 'Пункт выдачи'::character varying]::text[]))
Ссылки извне:
    TABLE "sales" CONSTRAINT "sales_filial_id_fkey" FOREIGN KEY (filial_id) REFERENCES filials(filial_id)
    TABLE "stock" CONSTRAINT "stock_filial_id_fkey" FOREIGN KEY (filial_id) REFERENCES filials(filial_id)
    TABLE "workers" CONSTRAINT "workers_filial_id_fkey" FOREIGN KEY (filial_id) REFERENCES filials(filial_id)
```

Таблица «workers»

```
shoes_sale=# \d workers
                                Таблица "public.workers"
   Столбец   |          Тип          | Правило сортировки | Допустимость NULL | По умолчанию
-------------+-----------------------+--------------------+-------------------+--------------
 worker_id   | character varying(3) |                    | not null          |
 worker_name | text                  |                    | not null          |
 position    | text                  |                    | not null          |
 filial_id   | character varying(3) |                    | not null          |
 doc_ser     | numeric(4,0)          |                    | not null          |
 doc_num     | numeric(6,0)          |                    | not null          |
 birthday    | date                  |                    | not null          |
 gender      | text                  |                    | not null          |
 tel         | text                  |                    | not null          |
Индексы:
    "workers_pkey" PRIMARY KEY, btree (worker_id)
    "unique_passport" UNIQUE CONSTRAINT, btree (doc_ser, doc_num)
Ограничения-проверки:
    "workers_gender_check" CHECK (gender = ANY (ARRAY['male'::text, 'female'::text]))
Ограничения внешнего ключа:
    "workers_filial_id_fkey" FOREIGN KEY (filial_id) REFERENCES filials(filial_id)
Ссылки извне:
    TABLE "sales" CONSTRAINT "sales_worker_id_fkey" FOREIGN KEY (worker_id) REFERENCES workers(worker_id)
    TABLE "supplies" CONSTRAINT "supplies_worker_id_fkey" FOREIGN KEY (worker_id) REFERENCES workers(worker_id)
```

Таблица «suppliers»

```
shoes_sale=# \d suppliers
                                Таблица "public.suppliers"
     Столбец      |          Тип          | Правило сортировки | Допустимость NULL | По умолчанию
------------------+-----------------------+--------------------+-------------------+--------------
 supplier_id      | character varying(3) |                    | not null          |
 supplier_name    | text                  |                    | not null          |
 type_of_supplier | text                  |                    | not null          |
 tel              | text                  |                    | not null          |
 adress           | jsonb                 |                    | not null          |
Индексы:
    "suppliers_pkey" PRIMARY KEY, btree (supplier_id)
Ограничения-проверки:
    "suppliers_type_of_supplier_check" CHECK (type_of_supplier = ANY (ARRAY['физ. лицо'::text, 'юр. лицо'::text]))
Ссылки извне:
    TABLE "supplies" CONSTRAINT "supplies_supplier_id_fkey" FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id)
```

Таблица «clients»

```
shoes_sale=# \d clients
                                Таблица "public.clients"
     Столбец     |         Тип          | Правило сортировки | Допустимость NULL | По умолчанию
-----------------+----------------------+--------------------+-------------------+--------------
 client_id       | character varying(3) |                    | not null          |
 clients_name    | text                 |                    | not null          |
 type_of_clients | text                 |                    | not null          |
 doc_ser         | numeric(4,0)         |                    | not null          |
 doc_num         | numeric(6,0)         |                    | not null          |
 birthday        | date                 |                    | not null          |
 gender          | text                 |                    | not null          |
 snils           | text                 |                    | not null          |
 tel             | text                 |                    | not null          |
 adress          | jsonb                |                    | not null          |
Индексы:
    "clients_pkey" PRIMARY KEY, btree (client_id)
    "unique_passport_client" UNIQUE CONSTRAINT, btree (doc_ser, doc_num)
Ограничения-проверки:
    "clients_gender_check" CHECK (gender = ANY (ARRAY['male'::text, 'female'::text]))
    "clients_type_of_clients_check" CHECK (type_of_clients = ANY (ARRAY['Госконтрактник'::text, 'Платник'::text, 'Платник с в
озмещением'::text]))
Ссылки извне:
    TABLE "sales" CONSTRAINT "sales_client_id_fkey" FOREIGN KEY (client_id) REFERENCES clients(client_id)
```

Таблица «nomenclature»

```
shoes_sale=# \d nomenclature
                             Таблица "public.nomenclature"
    Столбец     |         Тип          | Правило сортировки | Допустимость NULL | По умолчанию
----------------+----------------------+--------------------+-------------------+--------------
 model_id       | character varying(3) |                    | not null          |
 vendor_code    | character varying(4) |                    | not null          |
 model_name     | text                 |                    | not null          |
 section        | text                 |                    | not null          |
 season         | text                 |                    | not null          |
 color          | text                 |                    | not null          |
 price          | numeric(7,2)         |                    | not null          |
 purchase_price | numeric(7,2)         |                    |                   |
 size           | integer              |                    | not null          |
 brand          | text                 |                    | not null          |
Индексы:
    "nomenclature_pkey" PRIMARY KEY, btree (model_id)
    "unique_model" UNIQUE CONSTRAINT, btree (vendor_code, size)
Ограничения-проверки:
    "nomenclature_color_check" CHECK (color = ANY (ARRAY['Черный'::text, 'Синий'::text, 'Красный'::text, 'Коричневый'::text,
'Белый'::text]))
    "nomenclature_season_check" CHECK (season = ANY (ARRAY['Зима'::text, 'Лето'::text, 'Весна-осень'::text]))
    "nomenclature_section_check" CHECK (section = ANY (ARRAY['Обувь женская'::text, 'Обувь мужская'::text, 'Обувь детская'::t
ext]))
Ссылки извне:
    TABLE "sales" CONSTRAINT "sales_model_id_fkey" FOREIGN KEY (model_id) REFERENCES nomenclature(model_id)
    TABLE "stock" CONSTRAINT "stock_model_id_fkey" FOREIGN KEY (model_id) REFERENCES nomenclature(model_id)
    TABLE "supplies" CONSTRAINT "supplies_model_id_fkey" FOREIGN KEY (model_id) REFERENCES nomenclature(model_id)
```

Таблица «stock»

```
shoes_sale=# \d stock
                              Таблица "public.stock"
  Столбец  |         Тип          | Правило сортировки | Допустимость NULL | По умолчанию
-----------+----------------------+--------------------+-------------------+--------------
 model_id  | character varying(3) |                    | not null          |
 amount    | integer              |                    | not null          |
 filial_id | character varying(3) |                    | not null          |
Индексы:
    "stock_pkey" PRIMARY KEY, btree (model_id)
Ограничения-проверки:
    "stock_amount_check" CHECK (amount >= 0)
Ограничения внешнего ключа:
    "stock_filial_id_fkey" FOREIGN KEY (filial_id) REFERENCES filials(filial_id)
    "stock_model_id_fkey" FOREIGN KEY (model_id) REFERENCES nomenclature(model_id)
```

Таблица «supplies»

```
shoes_sale=# \d supplies
                                Таблица "public.supplies"
    Столбец     |            Тип             | Правило сортировки | Допустимость NULL | По умолчанию
----------------+----------------------------+--------------------+-------------------+--------------
 supply_id      | character varying(10)      |                    | not null          |
 operation_date | timestamp without time zone |                    | not null          |
 supplier_id    | character varying(3)       |                    | not null          |
 total_amount   | numeric(9,2)               |                    | not null          |
 price_for_one  | numeric(7,2)               |                    | not null          |
 amount         | integer                    |                    | not null          |
 model_id       | character varying(3)       |                    | not null          |
 worker_id      | character varying(3)       |                    | not null          |
Индексы:
    "supplies_pkey" PRIMARY KEY, btree (supply_id)
Ограничения-проверки:
    "supplies_check" CHECK ((price_for_one * amount::numeric) = total_amount)
Ограничения внешнего ключа:
    "supplies_model_id_fkey" FOREIGN KEY (model_id) REFERENCES nomenclature(model_id)
    "supplies_supplier_id_fkey" FOREIGN KEY (supplier_id) REFERENCES suppliers(supplier_id)
    "supplies_worker_id_fkey" FOREIGN KEY (worker_id) REFERENCES workers(worker_id)
Триггеры:
    update_nomenclature AFTER INSERT ON supplies FOR EACH ROW EXECUTE FUNCTION update_nomenclature()
    update_stock AFTER INSERT ON supplies FOR EACH ROW EXECUTE FUNCTION update_stock()
```

Таблица «sales»

```
shoes_sale=# \d sales
                                Таблица "public.sales"
    Столбец     |            Тип             | Правило сортировки | Допустимость NULL | По умолчанию
----------------+----------------------------+--------------------+-------------------+--------------
 sale_id        | character varying(10)      |                    | not null          |
 operation_date | timestamp without time zone |                    | not null          |
 client_id      | character varying(3)       |                    | not null          |
 model_id       | character varying(3)       |                    | not null          |
 sale_type      | text                       |                    | not null          |
 order_status   | text                       |                    | not null          |
 amount         | integer                    |                    | not null          |
 form_of_payment | text                      |                    | not null          |
 delivery_method | text                      |                    | not null          |
 filial_id      | character varying(3)       |                    | not null          |
 worker_id      | character varying(3)       |                    | not null          |
Индексы:
    "sales_pkey" PRIMARY KEY, btree (sale_id)
Ограничения-проверки:
    "sales_delivery_method_check" CHECK (delivery_method = ANY (ARRAY['Самовывоз'::text, 'Курьер'::text, 'СДЭК'::text, 'Почта
 России'::text]))
    "sales_order_status_check" CHECK (order_status = ANY (ARRAY['В производстве'::text, 'Готов'::text, 'На примерке'::text, '
На складе'::text, 'Доставка'::text, 'Выдан на руки'::text]))
    "sales_sale_type_check" CHECK (sale_type = ANY (ARRAY['Индивидуальный пошив'::text, 'Из каталога'::text]))
Ограничения внешнего ключа:
    "sales_client_id_fkey" FOREIGN KEY (client_id) REFERENCES clients(client_id)
    "sales_filial_id_fkey" FOREIGN KEY (filial_id) REFERENCES filials(filial_id)
    "sales_model_id_fkey" FOREIGN KEY (model_id) REFERENCES nomenclature(model_id)
    "sales_worker_id_fkey" FOREIGN KEY (worker_id) REFERENCES workers(worker_id)
Триггеры:
    update_stock2 AFTER INSERT ON sales FOR EACH ROW EXECUTE FUNCTION update_stock2()
```

Процедуры и триггеры

Как уже упоминалось выше, при закупке и продаже товара необходимо обновлять информацию по наличию товара на складе (в подразделениях компании). Для этого были написаны 2 триггера. Один из них срабатывает, когда в таблицу закупки «supplies» вносится (insert) новая строчка. Другой – когда в таблицу продажи «sales» вносится (insert) новая строчка.

```
-- Триггер 1

CREATE OR REPLACE FUNCTION update_stock() RETURNS trigger AS
$$
DECLARE
delta stock.amount%TYPE;
id_model stock.model_id%TYPE;

BEGIN
IF TG_OP = 'INSERT' THEN
delta = NEW.amount;
id_model = NEW.model_id;
END IF;

UPDATE stock s SET amount = amount + delta
WHERE model_id = id_model;

RETURN NULL;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS update_stock ON supplies;

CREATE TRIGGER update_stock
AFTER INSERT ON supplies
FOR EACH ROW EXECUTE PROCEDURE update_stock();
```

```
-- Триггер 2

CREATE OR REPLACE FUNCTION update_stock2() RETURNS trigger AS
$$
DECLARE
delta stock.amount%TYPE;
id_model stock.model_id%TYPE;

BEGIN
IF TG_OP = 'INSERT' THEN
delta = NEW.amount;
id_model = NEW.model_id;
END IF;

UPDATE stock s SET amount = amount - delta
WHERE model_id = id_model;

RETURN NULL;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS update_stock2 ON sales;

CREATE TRIGGER update_stock2
AFTER INSERT ON sales
FOR EACH ROW EXECUTE PROCEDURE update_stock2();
```

Еще один триггер понадобился для внесения изменений в закупочную цену в таблице номенклатура при внесении строки в таблицу «supplies».

```sql
-- Триггер 3
CREATE OR REPLACE FUNCTION update_nomenclature() RETURNS trigger AS
$$
DECLARE
purchasing_price supplies.price_for_one%TYPE;
id_model supplies.model_id%TYPE;

BEGIN
IF TG_OP = 'INSERT' THEN
purchasing_price = NEW.price_for_one;
id_model = NEW.model_id;
END IF;

UPDATE nomenclature n SET purchase_price = purchasing_price
WHERE model_id = id_model;

RETURN NULL;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS update_nomenclature ON supplies;

CREATE TRIGGER update_nomenclature
AFTER INSERT ON supplies
FOR EACH ROW EXECUTE PROCEDURE update_nomenclature();
```

К данной БД было написано 2 функции:

1. Связана с особенностями таблицы «sales». Поскольку в ней не указана цена, а только кол-во проданных пар, то может потребоваться запрос с целью получения стоимости одной пары обуви проданной модели и общая сумма продажи, если было продано более одной пары обуви.

```sql
-- Функция 1

DROP FUNCTION amount_of_sale ( sale_id varchar(10) );

CREATE OR REPLACE FUNCTION amount_of_sale( sale_id varchar(10),
OUT price_for_one numeric(7, 2), OUT total_amount numeric (9, 2), OUT amount integer)
AS $$
SELECT n.price AS price_for_one,
n.price * s.amount AS total_amount,
s.amount AS amount
FROM nomenclature n JOIN sales s
ON n.model_id = s.model_id
WHERE s.sale_id = sale_id;
$$ LANGUAGE sql;
```

Пример работы функции 1:
На вход функция должна получить id продажи.

```
shoes_sale=# select * from amount_of_sale('2');
 price_for_one | total_amount | amount
---------------+--------------+--------
       7990.00 |     15980.00 |      2
(1 строка)

shoes_sale=# select * from amount_of_sale('1');
 price_for_one | total_amount | amount
---------------+--------------+--------
       9900.00 |      9900.00 |      1
(1 строка)
```

2. Для любой компании очень важно понимать, насколько она является прибыльной. Таким образом, периодически руководство просит отчет о прибыли, полученной компанией за определенной период.

С этой целью была написана функция, которая на вход требует начальную и конечную дату и выдает общую сумму закупок, общую сумму продаж и рассчитывает прибыль как разницу между продажами и закупками за данный период. Границы дат включаются.

```sql
-- Функция 2 (сложн)

DROP FUNCTION profit_per_period( start_date date, finish_date date );

CREATE OR REPLACE FUNCTION profit_per_period2( start_date date, finish_date date,
OUT start_dat date, OUT finish_date date, OUT total_sale_price numeric (11, 2),
OUT total_purchase_price numeric (11, 2), OUT profit numeric (11, 2))
AS $$

SELECT
start_date,
finish_date,

( SELECT sum( n.price * s.amount)
FROM sales s JOIN nomenclature n
ON s.model_id = n.model_id
WHERE date_trunc('day', s.operation_date) >= start_date AND date_trunc('day', s.operation_date) <= finish_date)
AS total_sale_price,

( SELECT sum( total_amount )
FROM supplies
WHERE date_trunc('day', operation_date) >= start_date AND date_trunc('day', operation_date) <= finish_date)
AS total_purchase_price,

( SELECT sum( n.price * s.amount)
FROM sales s JOIN nomenclature n
ON s.model_id = n.model_id
WHERE date_trunc('day', s.operation_date) >= start_date AND
 date_trunc('day', s.operation_date) <= finish_date) - ( SELECT sum( total_amount )
FROM supplies
WHERE date_trunc('day', operation_date) >= start_date AND date_trunc('day', operation_date) <= finish_date) AS profit;

$$ LANGUAGE sql;
```

Пример работы функции 2:

```
shoes_sale=# select * from profit_per_period( '2021-10-01', '2021-10-31');
 start_dat  | finish_date | total_sale_price | total_purchase_price |  profit
------------+-------------+------------------+----------------------+-----------
 2021-10-01 | 2021-10-31  |         59970.00 |            121600.00 | -61630.00
(1 строка)

shoes_sale=# select * from profit_per_period( '2021-09-01', '2021-09-30');
 start_dat  | finish_date | total_sale_price | total_purchase_price |   profit
------------+-------------+------------------+----------------------+------------
 2021-09-01 | 2021-09-30  |         63577.00 |            300500.00 | -236923.00
(1 строка)

shoes_sale=# select * from profit_per_period( '2021-10-01', '2021-10-05');
 start_dat  | finish_date | total_sale_price | total_purchase_price |  profit
------------+-------------+------------------+----------------------+-----------
 2021-10-01 | 2021-10-05  |         12580.00 |             25000.00 | -12420.00
(1 строка)
```

Отрицательная прибыль связана с тем, что у при вставке данных в таблицы я вставила много закупок (на большие сумы), а продаж было немного (точнее они были на меньшие суммы), поскольку покупателями выступают частные лица (розничная торговля).

Комментарии

При вставке данных по продажам и закупкам дата вводилась вручную с целью создания функционирующей БД во времени. Однако изначально предполагалось, что дата и время будут определяться автоматически

Примеры вставок в таблицы «sales» и «supplies»:

```
INSERT INTO sales (sale_id, operation_date, client_id, model_id, sale_type, order_status, amount, form_of_payment, delivery_method, filial_id, worker_id)
VALUES ('100', CURRENT_TIMESTAMP, '001', '001', 'Из каталога', 'Доставка', 2, 'По карте', 'Курьер', '001', '001');

INSERT INTO supplies (supply_id, operation_date, supplier_id, total_amount, price_for_one, amount, model_id, worker_id)
VALUES ('100', CURRENT_TIMESTAMP, '001', 15000, 5000, 3, '001', '001');
```

Также отдельно проверялась возможность продажи пары обуви, которая отсутствует на складе (на нем 0 шт.).

```
-- продажа, которая не может совершиться, тк в наличии нет данной пары обуви
INSERT INTO sales (sale_id, operation_date, client_id, model_id, sale_type, order_status, amount, form_of_payment, delivery_method, filial_id, worker_id)
VALUES ('23', '2021-11-17 15:43:30', '002', '005', 'Из каталога', 'Доставка', 2, 'По карте', 'СДЭК', '004', '007');
```

```
ОШИБКА:  новая строка в отношении "stock" нарушает ограничение-проверку "stock_amount_check"
ПОДРОБНОСТИ:  Ошибочная строка содержит (005, -2, 006).
КОНТЕКСТ:  SQL-оператор: "UPDATE stock s SET amount = amount - delta
WHERE model_id = id_model"
функция PL/pgSQL update_stock2(), строка 12, оператор SQL-оператор
```

Примеры запросов к БД

Запрос 1

Рассчитать сколько продал менеджер с id ‘003’ за определенный период (за октябрь)

```
SELECT sum ( n.price * s.amount ) AS cumulative_sales
FROM sales s JOIN nomenclature n
ON s.model_id = n.model_id
WHERE worker_id = '003'
AND date_trunc('day', s.operation_date) >= '2021-10-01'::date AND date_trunc('day', s.operation_date) <= '2021-10-31'::date;
```

```
 cumulative_sales
------------------
         21580.00
(1 строка)
```

Проверка запроса:

```
shoes_sale=# SELECT * FROM sales
WHERE worker_id = '003'
AND date_trunc('day', operation_date) >= '2021-10-01'::date AND date_trunc('day', operation_date) <= '2021-10-31'::date;
 sale_id |   operation_date    | client_id | model_id |  sale_type  | order_status  | amount | form_of_payment | delivery_method | filial_id | worker_id
---------+---------------------+-----------+----------+-------------+---------------+--------+-----------------+-----------------+-----------+----------
 8       | 2021-10-02 09:17:06 | 004       | 006      | Из каталога | Доставка      |      1 | По карте        | Курьер          | 006       | 003
 9       | 2021-10-05 14:56:16 | 002       | 004      | Из каталога | Доставка      |      1 | По карте        | Самовывоз       | 007       | 003
 14      | 2021-10-19 20:40:50 | 002       | 013      | Из каталога | Выдан на руки |      1 | По карте        | Самовывоз       | 010       | 003
(3 строки)
```

Далее я вручную все сложила и получила указанная выше сумма.

Запрос 2 (оконная функция)

Предоставление сведений о распределении кол-ва выписанных чеков (именно чеков, не штук!) конкретной модели обуви во времени. Для примера была взята модель ‘010’.

```
SELECT
s.operation_date,
extract( 'month' from s.operation_date ) AS month,
extract( 'day' from s.operation_date ) AS day,
count( * ) OVER (
PARTITION BY date_trunc( 'month', s.operation_date )
ORDER BY s.operation_date
) AS count
FROM sales s
WHERE s.model_id = '010';
```

Перед запуском для более наглядной работы запроса предварительно надо добавить дополнительные строчки в таблицу «sales»:

```
INSERT INTO sales (sale_id, operation_date, client_id, model_id, sale_type, order_status, amount,
form_of_payment, delivery_method, filial_id, worker_id)
VALUES ('21', '2021-10-11 14:58:30', '003', '010', 'Из каталога', 'Доставка', 1, 'По карте', 'СДЭК', '006', '005');
INSERT INTO sales (sale_id, operation_date, client_id, model_id, sale_type, order_status, amount,
form_of_payment, delivery_method, filial_id, worker_id)
VALUES ('22', '2021-10-17 15:43:30', '002', '010', 'Из каталога', 'Доставка', 2, 'По карте', 'СДЭК', '004', '007');
```

```
   operation_date    | month | day | count
---------------------+-------+-----+-------
 2021-09-30 15:37:00 |     9 |  30 |     1
 2021-10-06 12:11:30 |    10 |   6 |     1
 2021-10-11 14:58:30 |    10 |  11 |     2
 2021-10-17 15:43:30 |    10 |  17 |     3
 2021-11-04 19:36:57 |    11 |   4 |     1
(5 строк)
```

Запрос 3 (CTE)

Посмотреть какой тип клиентов (госконтрактник, платник, платник с возмещением) больше приносит дохода.

```
WITH cpr AS
( SELECT s.client_id,
sum( n.price * s.amount)
FROM sales s JOIN nomenclature n
ON s.model_id = n.model_id
GROUP BY s.client_id
)
SELECT
c.type_of_clients,
sum( cpr.sum )
FROM cpr JOIN clients c
ON cpr.client_id = c.client_id
GROUP BY c.type_of_clients;
```

```
    type_of_clients    |    sum
-----------------------+-----------
 Платник               | 102960.00
 Госконтрактник        |  45780.00
 Платник с возмещением |  30124.00
(3 строки)
```

Предварительно тестировали подзапросы:

```
SELECT
client_id,
sum( n.price * s.amount)
FROM sales s JOIN nomenclature n
ON s.model_id = n.model_id
GROUP BY s.client_id;
```

```
 client_id |   sum
-----------+----------
 005       | 13590.00
 002       | 50970.00
 007       | 18507.00
 001       | 29700.00
 003       | 22290.00
 004       | 32190.00
 006       | 11617.00
(7 строк)
```

```
SELECT
type_of_clients
FROM clients
GROUP BY type_of_clients;
```

```
    type_of_clients
-----------------------
 Платник
 Госконтрактник
 Платник с возмещением
(3 строки)
```

Запрос 4

Посчитать сумму закупок по месяцам и сумму продаж по месяцам

```
SELECT extract( 'year' from operation_date ) AS year,
extract( 'month' from operation_date ) AS month,
sum( total_amount ) AS total_purchase_price
FROM supplies
GROUP BY date_trunc( 'month', operation_date ), year, month
ORDER BY year, month;
```

```
 year | month | total_purchase_price
------+-------+----------------------
 2021 |     9 |            300500.00
 2021 |    10 |            121600.00
 2021 |    11 |             18000.00
(3 строки)
```

Запрос 5

Посчитать сумму продаж по месяцам

```
SELECT extract( 'year' from s.operation_date ) AS year,
extract( 'month' from s.operation_date ) AS month,
```

```
sum( n.price * s.amount) AS total_sale_price
FROM sales s JOIN nomenclature n
ON s.model_id = n.model_id
GROUP BY date_trunc( 'month', s.operation_date ), year, month
ORDER BY year, month;
```

```
 year | month | total_sale_price
------+-------+------------------
 2021 |     9 |         63577.00
 2021 |    10 |         77670.00
 2021 |    11 |         37617.00
(3 строки)
```

Запрос 6 (CTE)

Посмотреть из какого подразделения больше всего идут продажи.

```
WITH fpr AS
(SELECT s.filial_id,
sum( n.price * s.amount) AS total_sale_price
FROM sales s JOIN nomenclature n
ON s.model_id = n.model_id
WHERE date_trunc('day', s.operation_date) >= '2021-10-01'::date AND date_trunc('day',
s.operation_date) <= '2021-10-31'::date
GROUP BY s.filial_id
)
SELECT
fpr.filial_id,
f.type_of_filials,
fpr.total_sale_price,
f.adress
FROM fpr JOIN filials f
ON fpr.filial_id = f.filial_id
ORDER BY fpr.total_sale_price DESC;
```

```
shoes_sale=# \i /mnt/sql_codes/final_project/select6.sql
 filial_id | type_of_filials | total_sale_price |                                                 adress
-----------+-----------------+------------------+---------------------------------------------------------------------------------------------
 006       | Склад           |         16390.00 | {"адрес": "г. Москва, проспект Вернадского, 8к1"}
 004       | Магазин         |         11800.00 | {"адрес": "г. Москва, Скатертный переулок, 4/2с1"}
 009       | Пункт выдачи    |         11800.00 | {"адрес": "Московская область, г.Одинцово, ул. Комсомольская, 6"}
 003       | Магазин         |          9900.00 | {"адрес": "г. Москва, 2-й Кадашёвский переулок, 5с2"}
 010       | Пункт выдачи    |          9000.00 | {"адрес": "Московская область, г.Красногорск, Советская улица, 2"}
 007       | Склад           |          7990.00 | {"адрес": "г. Москва, Рязанский проспект, 4"}
 008       | Пункт выдачи    |          6200.00 | {"адрес": "Московская область, микрорайон Северный, г.Домодедово, ул. Центральная, 22"}
 011       | Пункт выдачи    |          4590.00 | {"адрес": "Московская область, г. Мытищи,  ул. Разведчика Абеля, 3"}
(8 строк)
```

Запрос экзаменационный

Проранжировать модели по доходу (суммарные продажи)

```
WITH tpr AS
( SELECT n.model_id,
n.model_name,
sum( n.price * s.amount)
FROM sales s JOIN nomenclature n
ON s.model_id = n.model_id
```

```
GROUP BY n.model_id
)

SELECT tpr.model_id,
tpr.model_name,
tpr.sum
FROM tpr
ORDER BY tpr.sum DESC;
```

model_id	model_name	sum
010	детские ортопедические сандалии Ортомода, натуральная кожа 2139-ХМ-019	41300.00
004	Женские полуботинки на диабетическую стопу Ортомода, натуральная кожа 8502-Л-502	23970.00
001	Ботильоны из натуральной кожи, 82374-Х-106	19800.00
008	МУЖСКИЕ БОТИНКИ ОРТОПЕДИЧЕСКИЕ ЗИМНИЕ ОРТОМОДА, НАТУРАЛЬНАЯ КОЖА, МЕХ 9131 Б/ш-м	12400.00
009	детские ортопедические сандалии Ортомода, натуральная кожа 2139-ХМ-019	11800.00
002	Ботильоны из натуральной кожи, 82374-Х-106	9900.00
020	МУЖСКИЕ ПОЛУБОТИНКИ ОРТОПЕДИЧЕСКИЕ ДЕМИСЕЗОННЫЕ ОРТОМОДА, НАТУРАЛЬНАЯ КОЖА 9229	9900.00
006	САБО АНАТОМИЧЕСКИЕ ЛЕТНИЕ, НАТУРАЛЬНАЯ КОЖА 9161	9180.00
015	Зимние ботинки на велькро 82411-Н-927	9000.00
013	Зимние ботинки на велькро 82411-Н-927	9000.00
018	Мужские полуботинки на диабетическую стопу Ортомода, натуральная кожа 8122 в (муж)	6590.00
017	ЖЕНСКИЕ ТУФЛИ ОРТОПЕДИЧЕСКИЕ ЛЕТНИЕ ОРТОМОДА, НАТУРАЛЬНАЯ КОЖА 8226	5717.00
016	ЖЕНСКИЕ ТУФЛИ ОРТОПЕДИЧЕСКИЕ ЛЕТНИЕ ОРТОМОДА, НАТУРАЛЬНАЯ КОЖА 8226	5717.00
007	САБО АНАТОМИЧЕСКИЕ ЛЕТНИЕ, НАТУРАЛЬНАЯ КОЖА 9161	4590.00

(14 строк)