

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Базы данных

Информационная система аптеки

Кондренко К.П., группа 21203

7 апреля 2024 г.

Содержание

1	Задание	3
1.1	Описание предметной области	3
2	Схема базы данных	4
2.1	Описание таблиц	4
2.2	Создание таблиц	6
2.3	Ограничения по поддержанию целостности	10
3	Реализация запросов к базы данных	12

1 Задание

Разработать структуру базы данных для информационной системы аптеки и реализовать приложение в архитектуре клиент-сервер, выполняющее операции внесения данных в базу данных, редактирование данных и запросы.

1.1 Описание предметной области

Аптека продает медикаменты и изготавливает их по рецептам. Лекарства могут быть разных типов:

1. Готовые лекарства: таблетки, мази, настойки.
2. Изготавливаемые аптекой: микстуры, мази, растворы, настойки, порошки.

Различие в типах лекарств отражается в различном наборе атрибутов, их характеризующих. Микстуры и порошки изготавливаются только для внутреннего применения, растворы для наружного, внутреннего применения и для смешивания с другими лекарствами и мази только для наружного применения. Лекарство различны также по способу приготовления и по времени приготовления. Порошки и мази изготавливаются смешиванием различных компонент. При изготовлении растворов и микстур ингредиенты не только смешивают, но и отстаивают с последующей фильтрацией лекарства, что увеличивает время изготовления.

В аптеке существует справочник технологий приготовления различных лекарств. В нем указываются: идентификационный номер технологии, название лекарства и сам способ приготовления. На складе на все медикаменты устанавливается критическая норма, т.е. когда какого-либо вещества на складе меньше критической нормы, то составляются заявки на данные вещества и их в срочном порядке привозят с оптовых складов медикаментов.

Для изготовления аптекой лекарства, больной должен принести рецепт от лечащего врача. В рецепте должно быть указано: ФИО, подпись и печать врача, ФИО, возраст и диагноз пациента, также количество лекарства и способ применения. Больной отдает рецепт регистратору, он принимает заказ и смотрит, есть ли компоненты заказываемого лекарства. Если не все компоненты имеются в наличии, то делает заявки на оптовые склады лекарств и фиксирует ФИО, телефон и адрес необслуженного покупателя, чтобы сообщить ему, когда доставят нужные компоненты. Такой больной пополняет справочник заказов - это те заказы, которые находятся в процессе приготовления, с пометкой, что не все компоненты есть для заказа. Если все компоненты имеются, то они резервируются для лекарства больного. Покупатель выплачивает цену лекарства, ему возвращается рецепт с пометкой о времени изготовления. Больной также пополняет справочник заказов в производстве. В назначенное время больной приходит и по тому же рецепту получает готовое лекарство. Такой больной пополняет список отданных заказов.

Ведется статистика по объемам используемых медикаментов. Через определенный промежуток времени производится инвентаризация склада. Это делается для того, чтобы определить, есть ли лекарства с критической нормой, или вышел срок хранения или недовысвободилось.

2 Схема базы данных

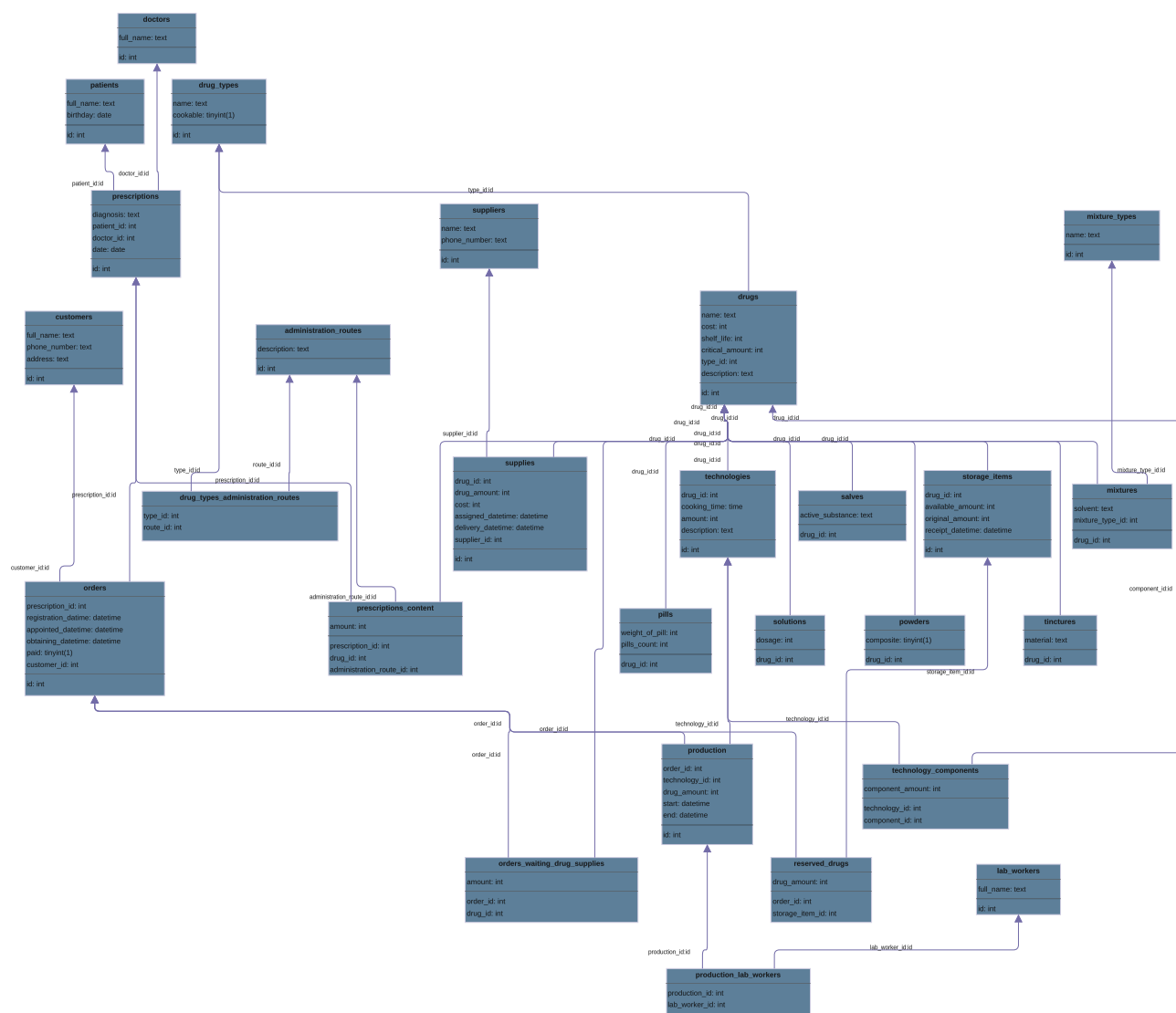


Рис. 1: Графическая схема базы данных

2.1 Описание таблиц

- **administration_routes** — способы применения лекарств (идентификатор способа, описание);
- **drug_types** — типы лекарств (идентификатор типа, название, являются ли приготавливаемыми лекарства данного типа);
- **mixture_types** — типы микстур (идентификатор типа, название);
- **lab_workers** — работники лаборатории аптеки, изготавливающие лекарства (идентификатор работника, ФИО);
- **patients** — пациенты, то есть те, на кого выписывают рецепты (идентификатор пациента, ФИО, дата рождения);
- **doctors** — врачи, которые выписывают рецепты для больных (идентификатор врача, ФИО);

- **customers** — клиенты аптеки (идентификатор клиента, ФИО, номер телефона, адрес);
- **suppliers** — поставщики лекарств в аптеку (идентификатор поставщика, название, номер телефона);
- **drugs** — лекарства (идентификатор лекарства, название, стоимость, срок годности в часах, критическая норма, идентификатор типа — из *drug_types*, описание);
- **mixtures** — микстуры (идентификатор лекарства — из *drugs*, растворитель, идентификатор типа микстуры — из *mixture_types*);
- **pills** — таблетки (идентификатор лекарства — из *drugs*, масса одной таблетки, количество таблеток в упаковке);
- **powders** — порошки (идентификатор лекарства — из *drugs*, составной порошок или нет);
- **salves** — мази (идентификатор лекарства — из *drugs*, действующее вещество);
- **solutions** — растворы (идентификатор лекарства — из *drugs*, концентрация);
- **tinctures** — настойки (идентификатор лекарства — из *drugs*, материал);
- **drug_types_administration_routes** — соответствие между типами лекарств и способами их применения (идентификатор типа, идентификатор способа);
- **prescriptions** — рецепты, выписанные больным врачами (идентификатор рецепта, диагноз, идентификатор пациента, идентификатор врача, дата);
- **orders** — заказы (идентификатор заказа, идентификатор рецепта — из *prescriptions*, дата и время регистрации, назначенные дата и время получения заказа, реальные дата и время получения заказа, оплачен ли заказ, идентификатор клиента — из *customers*);
- **prescriptions_content** — состав рецептов (идентификатор рецепта — из *prescriptions*, идентификатор лекарства — из *drugs*, количество лекарства, способ применения — из *administration_routes*);
- **storage_items** — позиции лекарств на складе (идентификатор позиции, идентификатор лекарства — из *drugs*, доступное количество лекарства в позиции на складе, исходное количество лекарства в позиции на складе; дата и время получения на складе);
- **supplies** — поставки лекарств от поставщиков (идентификатор поставки, идентификатор лекарства — из *drugs*, количество лекарства, общая стоимость, назначенные дата и время поставки, реальные дата и время поставки, идентификатор поставщика — из *suppliers*);
- **technologies** — справочник технологий приготовления лекарств (идентификатор технологии, идентификатор лекарства — из *drugs*, время приготовления, количество приготавливаемого лекарства, инструкция);
- **technology_components** — лекарства, требуемые для приготовления лекарств по технологиям (идентификатор технологии — из *technologies*, идентификатор лекарства, требуемого для технологии — из *drugs*, количество данного лекарства, требуемого для технологии);

- **production** — приготовления лекарств для заказов (идентификатор приготовления, идентификатор заказа — из *orders*, идентификатор технологии приготовления лекарства — из *technologies*, количествоготавливаемого лекарства, дата и время начала готовки, дата и время завершения готовки).
- **orders_waiting_drug_supplies** — поставки каких лекарств нужны для заказов (идентификатор заказа — из *orders*, идентификатор лекарства — из *drugs*, количество лекарства);
- **reserved_drugs** — какие лекарства со склада зарезервированы для заказов (идентификатор заказа — из *orders*, идентификатор позиции склада — из *storage_items*, количество лекарства);
- **production_lab_workers** — какие работники лаборатории участвуют в приготовлениях лекарств (идентификатор приготовления — из таблицы *production*, идентификатор работника лаборатории — *lab_workers*).

2.2 Создание таблиц

SQL-скрипт для создания таблиц базы данных

```
create table if not exists administration_routes
(
    id            int auto_increment
      primary key,
    description text not null
);

create table if not exists drug_types
(
    id            int auto_increment
      primary key,
    name          text      not null,
    cookable      tinyint(1) not null
);

create table if not exists mixture_types
(
    id            int auto_increment
      primary key,
    name text      not null
);

create table if not exists lab_workers
(
    id            int auto_increment
      primary key,
    full_name text not null
);

create table if not exists patients
(
    id            int auto_increment
      primary key,
    full_name text not null,
    birthday      date not null
);

create table if not exists doctors
(
    id            int auto_increment
      primary key,
    full_name text not null
);

create table if not exists customers
(
    id            int auto_increment
```

```

        primary key,
        full_name      text not null,
        phone_number    text not null,
        address         text not null
    );

create table if not exists suppliers
(
    id            int auto_increment
        primary key,
    name          text not null,
    phone_number  text not null
);

create table if not exists drugs
(
    id            int auto_increment
        primary key,
    name          text not null,
    cost          int not null,
    shelf_life    int not null,
    critical_amount int not null,
    type_id       int not null,
    description    text not null,
    constraint drugs_drug_types_id_fk
        foreign key (type_id) references drug_types (id),
    check ('cost' > 0),
    check ('shelf_life' > 0),
    check ('critical_amount' >= 0)
);

create table if not exists mixtures
(
    drug_id       int auto_increment
        primary key,
    solvent       text not null,
    mixture_type_id int not null,
    constraint mixtures_drugs_id_fk
        foreign key (drug_id) references drugs (id),
    constraint mixtures_mixture_types_id_fk
        foreign key (mixture_type_id) references mixture_types (id)
);

create table if not exists pills
(
    drug_id       int not null
        primary key,
    weight_of_pill int not null,
    pills_count   int not null,
    constraint pills_drugs_id_fk
        foreign key (drug_id) references drugs (id),
    check ('weight_of_pill' > 0)
);

create table if not exists powders
(
    drug_id      int not null
        primary key,
    composite    tinyint(1) not null,
    constraint powders_drugs_id_fk
        foreign key (drug_id) references drugs (id)
);

create table if not exists salves
(
    drug_id       int not null
        primary key,
    active_substance text not null,
    constraint salves_drugs_id_fk
        foreign key (drug_id) references drugs (id)
);

create table if not exists solutions
(
    drug_id int not null
        primary key,

```

```

dosage int not null,
constraint solutions_drugs_id_fk
foreign key (drug_id) references drugs (id),
check ((0 <= 'dosage') and ('dosage' <= 100))
);

create table if not exists tinctures
(
drug_id int not null
primary key,
material text not null,
constraint tinctures_drugs_id_fk
foreign key (drug_id) references drugs (id)
);

create table if not exists drug_types_administration_routes
(
type_id int not null,
route_id int not null,
constraint drug_types_administration_routes_administration_routes_id_fk
foreign key (route_id) references administration_routes (id),
constraint drug_types_administration_routes_drug_types_id_fk
foreign key (type_id) references drug_types (id)
);

create table if not exists prescriptions
(
id int auto_increment
primary key,
diagnosis text not null,
patient_id int not null,
doctor_id int not null,
date date not null,
constraint prescriptions_doctors_id_fk
foreign key (doctor_id) references doctors (id),
constraint prescriptions_patients_id_fk
foreign key (patient_id) references patients (id)
);

create table if not exists orders
(
id int auto_increment
primary key,
prescription_id int not null,
registration_datetime datetime not null,
appointed_datetime datetime null,
obtaining_datetime datetime null,
paid tinyint(1) not null,
customer_id int null,
constraint orders_customers_id_fk
foreign key (customer_id) references customers (id),
constraint orders_prescriptions_id_fk
foreign key (prescription_id) references prescriptions (id)
);

create table if not exists prescriptions_content
(
prescription_id int not null,
drug_id int not null,
amount int not null,
administration_route_id int not null,
primary key (prescription_id, drug_id, administration_route_id),
constraint prescriptions_content_administration_routes_id_fk
foreign key (administration_route_id) references administration_routes (id),
constraint prescriptions_content_drugs_id_fk
foreign key (drug_id) references drugs (id),
constraint prescriptions_content_prescriptions_id_fk
foreign key (prescription_id) references prescriptions (id),
check ('amount' > 0)
);

create table if not exists storage_items
(
id int auto_increment
primary key,
drug_id int not null,

```



```

    available_amount int not null,
    original_amount int not null,
    receipt_datetime datetime not null,
    constraint storage_items_drugs_id_fk
        foreign key (drug_id) references drugs (id),
    check ('original_amount' > 0),
    check ('available_amount' >= 0)
);

create table if not exists supplies
(
    id int auto_increment
        primary key,
    drug_id int not null,
    drug_amount int not null,
    cost int not null,
    assigned_datetime datetime not null,
    delivery_datetime datetime null,
    supplier_id int not null,
    constraint supplies_drugs_id_fk
        foreign key (drug_id) references drugs (id),
    constraint supplies_suppliers_id_fk
        foreign key (supplier_id) references suppliers (id),
    check ('drug_amount' > 0),
    check ('cost' >= 0)
);

create table if not exists technologies
(
    id int auto_increment
        primary key,
    drug_id int not null,
    cooking_time time not null,
    amount int not null,
    description text not null,
    constraint technologies_drugs_id_fk
        foreign key (drug_id) references drugs (id),
    check ('amount' > 0)
);

create table if not exists technology_components
(
    technology_id int not null,
    component_id int not null,
    component_amount int not null,
    primary key (component_id, technology_id),
    constraint technology_components_drugs_id_fk
        foreign key (component_id) references drugs (id),
    constraint technology_components_technologies_id_fk
        foreign key (technology_id) references technologies (id),
    constraint positive_component_amount_check
        check ('component_amount' > 0)
);

create table if not exists production
(
    id int auto_increment
        primary key,
    order_id int not null,
    technology_id int not null,
    drug_amount int not null,
    start datetime null,
    end datetime null,
    constraint production_orders_id_fk
        foreign key (order_id) references orders (id),
    constraint production_technologies_id_fk
        foreign key (technology_id) references technologies (id),
    constraint drug
        check ('drug_amount' > 0)
);

create table if not exists orders_waiting_drug_supplies
(
    order_id int not null,
    drug_id int not null,
    amount int not null,

```

```

    primary key (drug_id, order_id),
    constraint orders_waiting_supplies_list_drugs_id_fk
        foreign key (drug_id) references drugs (id),
    constraint orders_waiting_supplies_list_orders_id_fk
        foreign key (order_id) references orders (id),
    check ('amount' > 0)
);

create table if not exists reserved_drugs
(
    order_id          int not null,
    storage_item_id   int not null,
    drug_amount        int not null,
    primary key (order_id, storage_item_id),
    constraint reserved_drugs_orders_id_fk
        foreign key (order_id) references orders (id),
    constraint reserved_drugs_storage_items_id_fk
        foreign key (storage_item_id) references storage_items (id),
    check ('drug_amount' > 0)
);

create table if not exists production_lab_workers
(
    production_id int not null,
    lab_worker_id int not null,
    constraint production_lab_workers_lab_workers_id_fk
        foreign key (lab_worker_id) references lab_workers (id),
    constraint production_lab_workers_production_id_fk
        foreign key (production_id) references production (id)
);

```

2.3 Ограничения по поддержанию целостности

- Любой посетитель аптеки должен иметь рабочий российский номер телефона и корректный российский адрес проживания (столбцы *phone_number* и *address* в таблице **customers**);
- В любом приготовлении должен принимать участие как минимум один рабочий лаборатории;
- Стоимость любого лекарства должна быть выше чем суммарная стоимость компонент для изготовления этого лекарства по всем технологиями (таблицы **technologies** и **technology_components**) его приготовления;
- Для любой технологии должна быть хотя бы одна запись в таблице требуемых для неё компонентов;
- Если для изготовления лекарства существует какая-то технология, то это лекарство должно иметь изготавливаемый тип (столбец *cookable* в таблице **drug_types**).
- Любой поставщик должен иметь рабочий российский номер телефона (столбец *phone_number* в таблице **suppliers**);
- Способ применения лекарства в рецепте (столбец *administration_route_id* в таблице **prescriptions_content**) должен быть допустим для этого лекарства в соответствии с его типом (эта информация хранится в таблице **drug_types_administration_routes**);
- Любой пациент должен иметь дату рождения, которая не больше чем дата добавления пациента в таблицу (столбец *birthday* в таблице *patients*);
- Дата регистрации любого заказа должна быть больше даты выписки рецепта, соответствующего этому заказу (столбец *registration_datetime* в таблице **orders** и столбец *date* в таблице **prescriptions**);

- Если какой-нибудь заказ забрали, то он обязательно оплачен (если в таблице **orders** столбец *obtaining_datetime* не *null*, то столбец *paid* должен быть *True*);
- Если какой-нибудь заказ был забран, то у него должна быть назначенная дата и наоборот (если столбец *obtaining_datetime* не *null*, то столбец *appointed_datetime* не *null* в таблице **orders** и наоборот);
- Дата регистрации любого заказа должна быть не больше назначенной даты его получения, что в свою очередь должно быть не больше реальной даты его получения (столбцы *registration_datetime*, *appointed_datetime* и *obtaining_datetime* в таблице **orders**); Это должно проверяться при изменении полей *appointed_datetime* и *obtaining_datetime* любой записи таблицы *orders*;
- Никакой заказ не должен ждать поставки лекарств, которые для него не требуются, как и для любого заказа не должно изготавливаться лекарств, которые для него не требуются;
- Дата получения поставки на склад должна быть больше чем текущая дата (столбец *receipt_datetime* в таблице **storage_items**);
- Дата начала изготовления лекарства должна быть меньше даты окончания его изготовления (столбцы *start_datetime* и *end_datetime* в таблице **production**);
- Если завершилось приготовление лекарства, то для этой партии лекарства должна добавиться запись на складе;
- Если для изготовления лекарств для какого-то заказа были взяты лекарства со склада, то количество этого лекарства на складе должно уменьшиться;
- Если для какого-то заказа было зарезервировано некоторое лекарство в некотором количестве, то количество доступного лекарства на складе из этой партии должно уменьшиться на соответствующее количество.

3 Реализация запросов к базы данных

1. Получить сведения о покупателях, которые не пришли забрать свой заказ в назначенное им время и общее их число.

Сведения о покупателях

```
select distinct
  customer_id as id,
  customers.full_name,
  customers.phone_number,
  customers.address
from orders
join customers on orders.customer_id = customers.id
where
  orders.appointed_datetime is not null
  and appointed_datetime <= now()
  and (
    orders.obtaining_datetime is null
    or orders.obtaining_datetime <> orders.appointed_datetime
  )
```

Общее число покупателей

```
select
  count(distinct customer_id) as customers_count
from orders
join customers on orders.customer_id = customers.id
where
  orders.appointed_datetime is not null
  and appointed_datetime <= now()
  and (
    orders.obtaining_datetime is null
    or orders.obtaining_datetime <> orders.appointed_datetime
  )
```

2. Получить перечень и общее число покупателей, которые ждут прибытия на склад нужных им медикаментов в целом и по указанной категории медикаментов.

Перечень покупателей (в целом)

```
select distinct
  customers.id,
  customers.full_name
from orders
join orders_waiting_drug_supplies on orders.id = orders_waiting_drug_supplies.
  order_id
join customers on orders.customer_id = customers.id
```

Общее число покупателей (в целом)

```
select
  count(distinct customers.id) as customers_count
from orders
join orders_waiting_drug_supplies on orders.id = orders_waiting_drug_supplies.
  order_id
join customers on orders.customer_id = customers.id
```

Перечень покупателей (по указанной категории)

```
prepare stmt from '
  select distinct
    customers.id,
    customers.full_name
  from orders
    join orders_waiting_drug_supplies on orders.id = orders_waiting_drug_supplies.
      order_id
    join customers on orders.customer_id = customers.id
    join drugs on orders_waiting_drug_supplies.drug_id = drugs.id
  where drugs.type_id = ?
';
```

```

set @type_id = 2;
execute stmt using @type_id;

```

Общее число покупателей (по указанной категории)

```

prepare stmt from '
select
    count(distinct customers.id) as customers_count
from orders
    join orders_waiting_drug_supplies on orders.id = orders_waiting_drug_supplies.
        order_id
    join customers on orders.customer_id = customers.id
    join drugs on orders_waiting_drug_supplies.drug_id = drugs.id
where drugs.type_id = ?
';

set @type_id = 2;
execute stmt using @type_id;

```

3. Получить перечень десяти наиболее часто используемых медикаментов в целом и указанной категории медикаментов.

В целом

```

prepare stmt from '
with
    used_in_cooking_drugs as (
        select
            component_id as drug_id,
            sum(component_amount) as drug_amount
        from production
            join technology_components on production.technology_id =
                technology_components.technology_id
            join drugs on technology_components.component_id = drugs.id
        where start is not null
        group by component_id
    ),
    sold_drugs as (
        select
            drug_id,
            sum(amount) as drug_amount
        from orders
            join prescriptions_content using (prescription_id)
        where obtaining_datetime is not null
        group by drug_id
    ),
    user_drugs as (
        select
            drug_id,
            drugs.name,
            sum(drug_amount) as drug_amount
        from (
            select *
            from used_in_cooking_drugs
            union all
            select *
            from sold_drugs
        ) as _
        join drugs on drug_id = drugs.id

        group by drug_id)

select *
from user_drugs
order by drug_amount desc
limit ?;
';

set @limit = 10;

```

```
execute stmt using @limit;
```

По указанной категории медикаментов

```
prepare stmt from '
with
  used_in_cooking_drugs as (
    select
      component_id as drug_id,
      sum(component_amount) as drug_amount
    from production
      join technology_components on production.technology_id =
        technology_components.technology_id
      join drugs on technology_components.component_id = drugs.id
    where start is not null
    group by component_id
  ),

  sold_drugs as (
    select
      drug_id,
      sum(amount) as drug_amount
    from orders
      join prescriptions_content using (prescription_id)
    where obtaining_datetime is not null
    group by drug_id
  ),

  user_drugs as (
    select
      drug_id,
      drugs.name,
      sum(drug_amount) as drug_amount
    from (
      select *
      from used_in_cooking_drugs
      union all
      select *
      from sold_drugs
    ) as _
      join drugs on drug_id = drugs.id
    where type_id = ?
    group by drug_id)

  select *
  from user_drugs
  order by drug_amount desc
  limit ?;
';

set @limit = 10;
set @type_id = 2;

execute stmt using @type_id, @limit;
```

4. Получить какой объем указанных веществ использован за указанный период.

```
prepare stmt from '
with
  used_in_cooking_drugs as (
    select
      component_id as drug_id,
      sum(component_amount) as drug_amount
    from production
      join technology_components on production.technology_id =
        technology_components.technology_id
      join drugs on technology_components.component_id = drugs.id
    where start between ? and ?
    group by component_id
  ),

  sold_drugs as (
    select
      drug_id,
```

```

        sum(amount) as drug_amount
    from orders
        join prescriptions_content using (prescription_id)
    where obtaining_datetime between ? and ?
    group by drug_id
),

user_drugs as (
    select
        drug_id,
        drugs.name,
        sum(drug_amount) as drug_amount
    from (
        select *
        from used_in_cooking_drugs
        union all
        select *
        from sold_drugs
    ) as _
        join drugs on drug_id = drugs.id
    group by drug_id)

select *
from user_drugs
order by drug_amount desc
';

set @min_datetime = '2023/01/01';
set @max_datetime = '2023/05/01';

execute stmt using @min_datetime, @max_datetime, @min_datetime, @max_datetime;

```

5. Получить перечень и общее число покупателей, заказывавших определенное лекарство или определенные типы лекарств за данный период.

Перечень покупателей, заказавших определённое лекарство за данный период

```

prepare stmt from '
    select distinct
        customers.*
    from prescriptions_content
        join orders using (prescription_id)
        join customers on orders.customer_id = customers.id
    where
        (registration_datetime between ? and ?)
        and (prescriptions_content.drug_id = ?)
';

set @min_registration_datetime = '2023/01/01';
set @max_registration_datetime = '2025/01/01';
set @drug_id = 2;

execute stmt using @min_registration_datetime, @max_registration_datetime, @drug_id;

```

Общее число покупателей, заказавших определённое лекарство за данный период

```

prepare stmt from '
    select
        count(distinct customers.id) as customers_count
    from prescriptions_content
        join orders using (prescription_id)
        join customers on orders.customer_id = customers.id
    where
        (registration_datetime between ? and ?)
        and (prescriptions_content.drug_id = ?)
';

set @min_registration_datetime = '2023/01/01';
set @max_registration_datetime = '2025/01/01';
set @drug_id = 2;

execute stmt using @min_registration_datetime, @max_registration_datetime, @drug_id;

```

Перечень покупателей, заказавших лекарство определённого типа за данный период

```
prepare stmt from '
select distinct
    customers.id as customer_id,
    customers.full_name as customer_full_name
from prescriptions_content
    join drugs on prescriptions_content.drug_id = drugs.id
    join orders using (prescription_id)
    join customers on orders.customer_id = customers.id
where
    (registration_datetime between ? and ?)
    and (drugs.type_id = ?)
';

set @min_registration_datetime = '2023/01/01';
set @max_registration_datetime = '2025/01/01';
set @type_id = 2;

execute stmt using @min_registration_datetime, @max_registration_datetime, @type_id;
```

Общее число покупателей, заказавших лекарство определённого типа за данный период

```
prepare stmt from '
select
count(distinct customers.id) as customers_count
from prescriptions_content
    join drugs on prescriptions_content.drug_id = drugs.id
    join orders using (prescription_id)
    join customers on orders.customer_id = customers.id
where
    (registration_datetime between ? and ?)
    and (drugs.type_id = ?)
';

set @min_registration_datetime = '2023/01/01';
set @max_registration_datetime = '2025/01/01';
set @type_id = 2;

execute stmt using @min_registration_datetime, @max_registration_datetime, @type_id;
```

6. Получить перечень и типы лекарств, достигших своей критической нормы или закончившихся.

Перечень лекарств

```
with
    critical_amount_drugs as (
        select
            drugs.id as drug_id,
            drugs.name as drug_name,
            coalesce(sum(available_amount), 0) as drug_amount,
            critical_amount
        from drugs
        left join storage_items on drugs.id = storage_items.drug_id
        group by
            drugs.id,
            critical_amount
        having
            drug_amount <= critical_amount
    )
select *
from critical_amount_drugs
```

Типы лекарств

```
with
    critical_amount_drugs as (
        select
            drugs.id as drug_id,
            drugs.name as drug_name,
            coalesce(sum(available_amount), 0) as drug_amount,
            critical_amount
        from drugs
```



```

        left join storage_items on drugs.id = storage_items.drug_id
    group by
        drugs.id,
        critical_amount
    having
        drug_amount <= critical_amount
)

select distinct
    type_id,
    drug_types.name
from critical_amount_drugs
join drugs on critical_amount_drugs.drug_id = drugs.id
join db.drug_types on drugs.type_id = drug_types.id
order by
    type_id

```

7. Получить перечень лекарств с минимальным запасом на складе в целом и по указанной категории медикаментов.

В целом

```

with
    drugs_storage_amount as (
        select
            drugs.id as drug_id,
            drugs.name as drug_name,
            coalesce(sum(available_amount), 0) as drug_amount,
            critical_amount
        from drugs
        left join storage_items on drugs.id = storage_items.drug_id
        group by
            drugs.id,
            critical_amount
    ),
    ranked_drugs as (
        select
            drug_id,
            drug_name,
            dense_rank() over (order by drug_amount) as dr,
            drug_amount
        from drugs_storage_amount
        group by drug_id
    )
select
    drug_id,
    drug_name,
    drug_amount
from ranked_drugs
where dr = 1

```

По указанной категории медикаментов

```

with
    drugs_of_type_storage_amount as (
        select
            drugs.id as drug_id,
            drugs.name as drug_name,
            coalesce(sum(available_amount), 0) as drug_amount,
            critical_amount
        from drugs
        left join storage_items on drugs.id = storage_items.drug_id
        where
            type_id = 6
        group by
            drugs.id,
            critical_amount
    ),
    ranked_drugs as (
        select
            drug_id,
            drug_name,

```

```

        dense_rank() over (order by drug_amount) as dr,
        drug_amount
    from drugs_of_type_storage_amount
    group by drug_id
)

select
    drug_id,
    drug_name,
    drug_amount
from ranked_drugs
where dr = 1

```

8. Получить полный перечень и общее число заказов находящихся в производстве.

Полный перечень заказов

```

select distinct
    orders.*
from production
join orders on production.order_id = orders.id

```

Общее число заказов

```

select distinct
    orders.*
from production
join orders on production.order_id = orders.id

```

9. Получить полный перечень и общее число препаратов требующихся для заказов, находящихся в производстве.

Полный перечень препаратов

```

select
    technology_components.component_id,
    drugs.name as component_name,
    sum(production.drug_amount * technology_components.component_amount) as
        component_amount
from production
join technologies on production.technology_id = technologies.id
join technology_components on technologies.id = technology_components.technology_id
join drugs on component_id = drugs.id
group by technology_components.component_id

```

Общее число препаратов

```

select
    sum(production.drug_amount * technology_components.component_amount) as
        component_amount
from production
join technologies on production.technology_id = technologies.id
join technology_components on technologies.id = technology_components.technology_id
join drugs on component_id = drugs.id

```

10. Получить все технологии приготовления лекарств указанных типов, конкретных лекарств, находящихся в справочнике заказов в производстве.

Конкретных лекарств

```

prepare stmt from '
    select
        id as technology_id,
        cooking_time,
        amount,
        description
    from technologies
    where drug_id = ?
';

set @drug_id = 2;

execute stmt using @drug_id;

```

Лекарств данного типа

```
prepare stmt from '
select
    technologies.*
from technologies
    join drugs on technologies.drug_id = drugs.id
where drugs.type_id = ?
';

set @type_id = 2;

execute stmt using @type_id;
```

Лекарств в справочнике заказов в производстве

```
select distinct
    technologies.*
from technologies
    join drugs on technologies.drug_id = drugs.id
    join production on technologies.id = production.technology_id
```

11. Получить сведения о ценах на указанное лекарство в готовом виде, об объеме и ценах на все компоненты, требующиеся для этого лекарства.

```
prepare stmt from '
select
    technologies.id as technology_id,
    drugs.name as component_name,
    drugs.cost as component_cost,
    technology_components.component_amount
from technologies
    join technology_components on technologies.id = technology_components.
        technology_id
    join drugs on drugs.id = technology_components.component_id
where technologies.drug_id = ?
';

set @drug_id = 5;

execute stmt using @drug_id;
```

12. Получить сведения о наиболее часто делающих заказы клиентах на медикаменты определенного типа, на конкретные медикаменты.

На определённый тип лекарств

```
prepare stmt from '
select
    id as customer_id,
    full_name as customer_full_name,
    orders_count
from (
    select
        customers.id,
        customers.full_name,
        count(*) as orders_count,
        dense_rank() over (order by count(*) desc) as dr
    from orders
        join prescriptions_content using (prescription_id)
        join drugs on prescriptions_content.drug_id = drugs.id
        join customers on orders.customer_id = customers.id
    where type_id = ?
    group by customer_id
) all_orders_count_data
where dr = 1
';

set @type_id = 2;

execute stmt using @type_id;
```

На конкретные медикаменты

```
prepare stmt from '
select
    id as customer_id,
    full_name as customer_full_name,
    orders_count
from (
    select
        customers.id,
        customers.full_name,
        count(*) as orders_count,
        dense_rank() over (order by count(*) desc) as dr
    from orders
        join prescriptions_content using (prescription_id)
        join customers on orders.customer_id = customers.id
    where drug_id = ?
    group by customer_id
) all_orders_count_data
where dr = 1
';

set @drug_id = 3;

execute stmt using @drug_id;
```

13. Получить сведения о конкретном лекарстве (его тип, способ приготовления, названия всех компонент, цены, его количество на складе).

Сведения о конкретном лекарстве

```
prepare stmt from '
select
    drugs.name as drug_name,
    drug_types.name as drug_type,
    drugs.cost as drug_cost,
    coalesce(sum(available_amount), 0) as in_storage
from drugs
    join drug_types on drugs.type_id = drug_types.id
    left join storage_items on drugs.id = storage_items.drug_id
where drugs.id = ?
group by drug_id,
    drugs.name,
    drug_types.name,
    drugs.cost
';

set @drug_id = 4;

execute stmt using @drug_id;
```

Перечень технологий приготовления данного лекарства

```
prepare stmt from '
select
    technologies.id as technology_id,
    technologies.description as technology_description,
    technologies.cooking_time,
    technologies.amount as output_amount,
    sum(components.cost * component_amount) as total_components_cost
from drugs
    join technologies on drugs.id = technologies.drug_id
    join technology_components on technologies.id = technology_components.
        technology_id
    join drugs_components on components.id = technology_components.component_id
where drugs.id = ?
group by technologies.id
';

set @drug_id = 2;

execute stmt using @drug_id;
```