«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ «КПІ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Курсова робота

з курсу «Бази даних»

тема: «АІС аптеки»

Виконала:студентка 3 курсу

групи КА-71

Павлюк В.

Прийняв: Мухін В. Є.

Київ – 2020р.

Вступ

Актуальність. У наш час величезна кількість фірм використовують персональні комп'ютери для збереження і обробки будь-якого виду інформації. Ця інформація міститься в базах даних (БД). БД відіграють важливу роль в світі технологій, що розвивається. Все, з чим ми щодня взаємодіємо в житті, зафіксовано в який-небудь базі. Робота з БД є найважливішим навиком в роботі з комп'ютером.

Мета. Створення АІС Аптеки, що відповідає всім вимогам, вказаним у завданні.

Завдання. Спроектувати БД аптеки та написати всі необхідні запити для роботи з нею.

Практичне значення. Набуття досвіду проектування БД на основі наданої інформації про предметну область та поліпшення навичок у написанні функціональних запитів мовою MySQL.

Програмне забезпечення. При виконанні роботи було використано середовище розробки MySQL Workbench 6.3; операційну систему Windows 10; базу даних MySQL.

1. Постановка задачі

Постановка задачі полягає у проектуванні БД згідно текстового опису предметної області (аптеки):

Аптека продає медикаменти і виготовляє їх за рецептами. Ліки можуть бути різних типів:

1) готові ліки: таблетки, мазі, настоянки;
2) виготовлені аптекою: мікстури, мазі, розчини, настойки, порошки.

Різниця в типах ліків відбивається в різному наборі атрибутів, що їх характеризують. Мікстури і порошки виготовляються тільки для внутрішнього застосування, розчини для зовнішнього, внутрішнього застосування і для змішування з іншими ліками та мазі тільки для зовнішнього застосування. Ліки різні також за способом приготування і за часом приготування. Порошки і мазі виготовляються змішуванням різних компонент. При виготовленні розчинів і мікстур інгредієнти не тільки змішують, але і відстоюють з подальшою фільтрацією лік, що збільшує час виготовлення.

В аптеці існує довідник технологій приготування різних ліків. У ньому зазначаються: ідентифікаційний номер технології, назву ліків і сам спосіб приготування. На складі на всі медикаменти встановлюється критична норма, тобто коли будь-які речовини на складі менше критичної норми, то складаються заявки на дану речовину і її в терміновому порядку привозять з оптових складів медикаментів.

Для виготовлення аптекою лік, хворий повинен принести рецепт від лікаря. У рецепті повинно бути вказано: ПІБ, підпис і печатка лікаря, ПІБ, вік та діагноз пацієнта, також кількість ліків і спосіб застосування. Хворий віддає рецепт реєстратору, він приймає замовлення і дивиться, чи є компоненти з яких складаються ліки. Якщо не всі компоненти є в наявності, то робить заявки на оптові склади ліків і фіксує ПІБ, телефон та адресу не обслугованого покупця, щоб повідомити йому, коли прийдуть потрібні компоненти. Такий хворий поповнює довідник замовлень - це ті замовлення, які знаходяться в процесі приготування, з позначкою, що не всі компоненти є для замовлення. Якщо всі компоненти є, то вони резервуються для ліків хворого. Покупець сплачує ціну ліків, йому повертається рецепт з позначкою про час виготовлення. Дані про хворого заноситься у довідник замовлень у виробництві. У призначений час хворий приходить і за тим же рецептом отримує готові ліки. Інформація про хворого поповнює список виданих замовлень.

Ведеться статистика за обсягами використовуваних медикаментів. Через певний проміжок часу проводиться інвентаризація складу. Це робиться для того, щоб визначити, чи є ліки з критичною нормою, або вийшов строк зберігання або недостача.

Види запитів в інформаційній системі:

1. Отримати відомості про покупців, які не прийшли забрати своє замовлення в призначений їм час і загальне їх число.

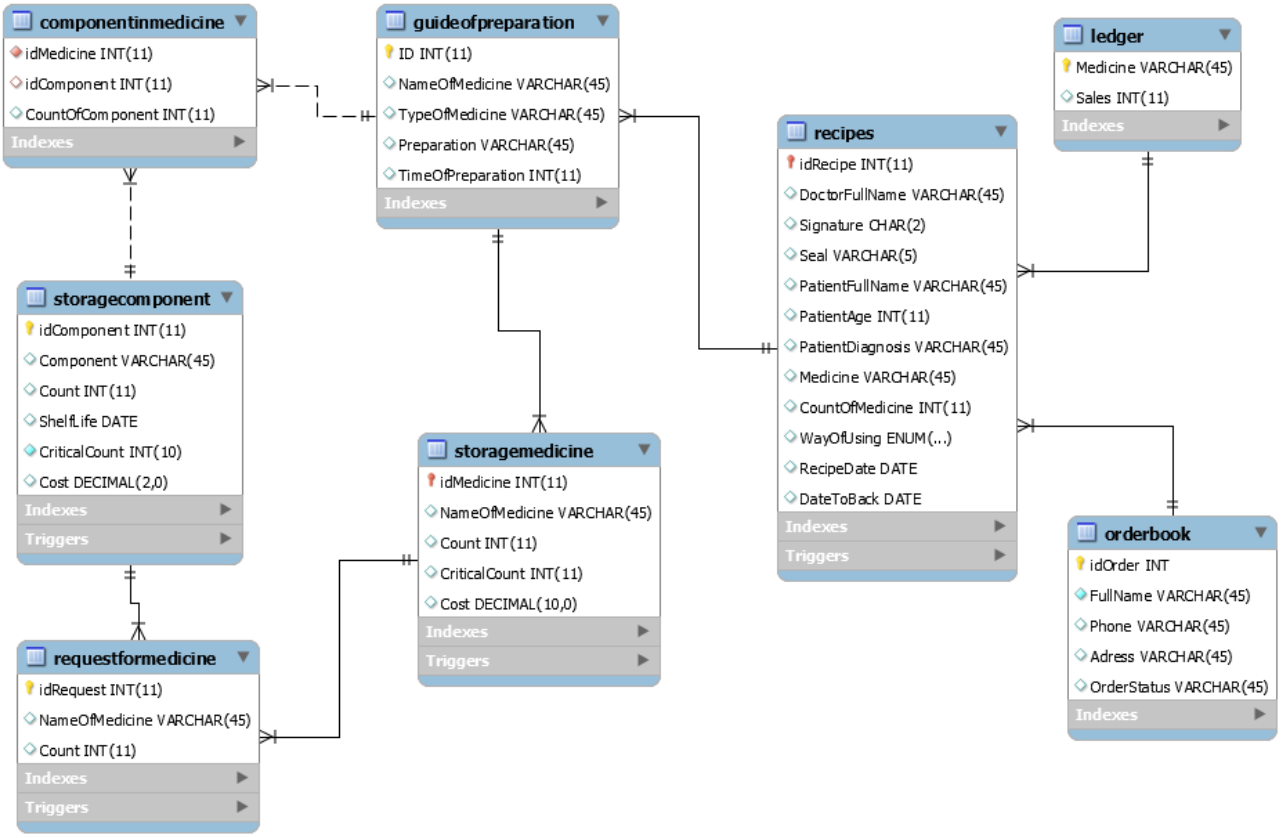
2. Отримати перелік і загальне число покупців, які чекають прибуття на склад потрібних їм медикаментів в цілому і по вказаній категорії медикаментів.
3. Отримати перелік десяти найбільш часто використовуваних медикаментів в цілому і зазначеної категорії медикаментів.
4. Отримати який обсяг зазначених речовин використаний за вказаний період.
5. Отримати перелік і загальне число покупців, які замовляли певні ліки або певні типи ліків за даний період.
6. Отримати перелік і типи ліків, які досягли своєї критичної норми або закінчилися.
7. Отримати перелік ліків з мінімальним запасом на складі в цілому і по вказаній категорії медикаментів.
8. Отримати повний перелік і загальне число замовлень знаходяться у виробництві.
9. Отримати повний перелік і загальне число препаратів потрібних для замовлень, що знаходяться у виробництві.
10. Отримати всі технології приготування ліків зазначених типів, конкретних ліків, ліків, що знаходяться в довіднику замовлень у виробництві.
11. Отримати відомості про ціни на вказані ліки в готовому вигляді, про обсяг і ціни на всі компоненти, потрібні для цих ліків.
12. Отримати відомості замовлення клієнтів, які найчастіші виготовляються, на медикаменти певного типу, на конкретні медикаменти.
13. Отримати відомості про конкретні ліки (їх тип, спосіб приготування, назви всіх компонент, ціни, кількість компонент на складі).

2. Архітектура та інформаційне забезпечення БД

2.1 Аналіз функціонування і організаційні основи підприємства

Завдання полягає у проектуванні БД аптеки. Проаналізувавши вимоги для обслуговування БД, було створено відповідні таблиці для збереження даних про склади, способи приготування ліків та їх види, покупців тощо.

2.2 Проектування структури бази даних



2.3 Життєві цикли бази даних

Попереднє планування

Попереднє планування складав аналіз наявних даних та функціональних вимог. В подальшому йшло формування структури даних та відповідних зв‘язків.

Перевірка здійсненності

У наявності є доступ до бази даних MySQL та відповідного програмного забезпечення Workbench 6.3, що дозволяє реалізувати мету.

Визначення вимог

Вимоги до бази даних були визначені за допомогою опису ПО. База даних описує всі необхідні елементи, вказані в описі.

Реалізація

Реалізація включала в себе написання тригерів та процедур на SQL.

3. Реалізація програмної взаємодії з БД

3.1 **Інструкція користувача**

Користувачеві необхідно мати доступ до програмного забезпечення MySQL Workbench. Тоді він зможе переглядати інформацію, додавати, видаляти та оновлювати дані. Інструкції щодо записів дивіться у таблиці нижче.

Таблиця 3.1.1

Назва запиту	**Параметри**	**Опис**
get_info_done_order	–	Отримати відомості про покупців, які не прийшли забрати своє замовлення в призначений їм час і загальне їх число.
1)get_info_wait_order 2)get_info_wait_for_order	1) – 2) Тип медикаментів	Отримати перелік і загальне число покупців, які чекають прибуття на склад потрібних їм медикаментів в цілому і по вказаній категорії медикаментів.
top_medicine	Тип медикаментів	Отримати перелік десяти найбільш часто використовуваних медикаментів в цілому і зазначеної категорії медикаментів.
count_of_medicine	Тип медикаментів, початкова дата періоду та кінцева дата.	Отримати який обсяг зазначених речовин використаний за вказаний період.
1)patients_list_medicine 2) patients_list_type	1)Назва ліків та межі періоду. 2)Тип ліків та межі періоду.	Отримати перелік і загальне число покупців, які замовляли певні ліки або певні типи ліків за даний період.
1)component_ended 2) medicine_ended	1) – 2) –	Отримати перелік і типи ліків, які досягли своєї критичної норми або закінчилися.
1)min_count_component 2)min_count_med 3)min_count_med_category	1) – 2) – 3) Тип ліків	Отримати перелік ліків з мінімальним запасом на складі в цілому і по вказаній категорії медикаментів.
order_status	Статус замовлення	Отримати повний перелік і

		загальне число замовлень знаходяться у виробництві.
request_med_inprogress	–	Отримати повний перелік і загальне число препаратів потрібних для замовлень, що знаходяться у виробництві.
1)technology_type 2) technology_name 3) technology_inprogress	1) Тип ліків 2) Назва ліків 3) –	Отримати всі технології приготування ліків зазначених типів, конкретних ліків, ліків, що знаходяться в довіднику замовлень у виробництві.
med_price	Назва ліків	Отримати відомості про ціни на вказані ліки в готовому вигляді, про обсяг і ціни на всі компоненти, потрібні для цих ліків.
1)top_orders 2)order_of_type 3)order_of_med	1) – 2) Тип ліків 2) Назва ліків	Отримати відомості замовлення клієнтів, які найчастіші виготовляються, на медикаменти певного типу, на конкретні медикаменти.
info_med	Назва ліків	Отримати відомості про конкретні ліки (їх тип, спосіб приготування, назви всіх компонент, ціни, кількість компонент на складі).
clean_up	–	Видалити записи рецептів, що не відповідають вимогам.
upd_orderbook	–	Оновлення статусів замовлень.
check_storage	–	Списання прострочених ліків.

3.2 Реалізація механізмів БД

3.2.1 SQL-запити.

```
CALL get_info_done_order();
CALL get_info_wait_order();
CALL get_info_wait_for_order('ointment');
CALL top_medicine('ointment');
CALL count_of_medicine('Nimesulid','2005-04-20', '2029-04-20');
CALL patients_list_medicine('Fastum', '2020-03-20' , '2020-04-20');
CALL patients_list_type('tablets', '2020-03-20' , '2020-04-20');
CALL component_ended ();
CALL medicine_ended ();
CALL min_count_component();
CALL min_count_med();
CALL min_count_med_category('tablets');
CALL order_status('InProgress');
CALL request_med_inprogress();
CALL technology_type('ointment');
CALL technology_name('Korvalol');
CALL technology_inprogress();
CALL med_price('Korvalol');
CALL med_price('Barvoval');
CALL top_orders();
CALL order_of_type('ointment');
CALL order_of_med('Korvalol');
CALL info_med('Barvoval');

CALL upd_orderbook();
CALL clean_up();
CALL check_storage();
INSERT INTO recipes VALUES (9, 'Joe Lotrulio', 'JL', 'true', 'Azize Aslanbey', 72,
    'osteoporos', 'Fastum', 1, 'external', '20.04.20','03.04.20');
DELETE FROM requestformedicine WHERE idRequest BETWEEN 21 AND 29;
```

3.2.2 Процедури і функції

Task1

```
CREATE PROCEDURE get_info_done_order()
BEGIN
  SELECT FullName,Phone,Adress FROM orderbook WHERE OrderStatus = 'Done'
      UNION
  SELECT 'Count Of Patients:', '', COUNT(*) as 'Count of DONE' FROM orderbook WHERE
      OrderStatus = 'Done';
END$$
```

Task2

```
CREATE PROCEDURE get_info_wait_order()
```

```
BEGIN
   SELECT FullName,Phone,Adress FROM orderbook WHERE OrderStatus = 'WaitForComponent'
   UNION
   SELECT 'Count of patients:', '', COUNT(*) FROM orderbook WHERE OrderStatus =
        'WaitForComponent';
END$$
CREATE PROCEDURE get_info_wait_for_order(IN type_med VARCHAR(45))
BEGIN
   SELECT FullName,Phone,Adress FROM (orderbook o JOIN recipes r ON o.idOrder =
        r.idRecipe) JOIN guideofpreparation g on g.NameOfMedicine = r.Medicine
   WHERE OrderStatus = 'WaitForComponent' AND TypeOfMedicine = type_med
   UNION
   SELECT 'Count of patients waiting', type_med, COUNT(*) FROM (orderbook o JOIN recipes r
        ON o.idOrder = r.idRecipe) JOIN guideofpreparation g on g.NameOfMedicine = r.Medicine
   WHERE OrderStatus = 'WaitForComponent' AND TypeOfMedicine = type_med;
END$$
```

Task3

```
CREATE PROCEDURE top_medicine(IN category VARCHAR(45))
BEGIN
        SELECT Medicine, Sales from ledger, GuideOfPreparation
   WHERE GuideOfPreparation.TypeOfMedicine = category AND
        GuideOfPreparation.NameOfMedicine = ledger.Medicine
   order by Sales DESC limit 10;

   SELECT * from ledger order by Sales DESC limit 10;
END//
```

Task4

```
CREATE PROCEDURE count_of_medicine (IN comp VARCHAR(45), IN date1 DATE, IN date2
        DATE)
BEGIN
        DECLARE id_comp INT;
   SELECT idComponent INTO id_comp FROM StorageComponent WHERE Component = comp;

        SELECT sum(c.CountOfComponent * r.CountOfMedicine) as 'COUNT'
   FROM (componentinmedicine c JOIN guideofpreparation g on c.idMedicine = g.ID) JOIN recipes
        r ON r.Medicine = g.NameOfMedicine
   WHERE (r.RecipeDate BETWEEN date1 AND date2) AND (c.idComponent = id_comp);

END//
```

Task5

```
CREATE PROCEDURE patients_list_medicine (IN med VARCHAR(45), IN date1 DATE, IN date2
        DATE)
BEGIN
        SELECT PatientFullName, RecipeDate, Medicine FROM recipes WHERE (RecipeDate
        BETWEEN date1 AND date2) AND (Medicine = med)
   UNION
```

```
    SELECT 'Total number:', '', COUNT(*) FROM recipes WHERE (RecipeDate BETWEEN date1
        AND date2) AND (Medicine = med);
END//

CREATE PROCEDURE patients_list_type (IN med_type VARCHAR(45), IN date1 DATE, IN
        date2 DATE)
BEGIN
        SELECT PatientFullName, RecipeDate, TypeOfMedicine FROM (recipes r JOIN
        guideofpreparation g ON r.Medicine = g.NameOfMedicine) WHERE (r.RecipeDate
        BETWEEN date1 AND date2) AND (TypeOfMedicine = med_type)
    UNION
    SELECT 'Total number:', '', COUNT(*) FROM recipes r JOIN guideofpreparation g ON
        r.Medicine = g.NameOfMedicine WHERE (r.RecipeDate BETWEEN date1 AND date2)
        AND (TypeOfMedicine = med_type);
END//
```

Task6

```
CREATE PROCEDURE component_ended ()
BEGIN
        SELECT Component, Count, CriticalCount FROM storagecomponent WHERE
        (Count<=CriticalCount)
    GROUP BY Component
    HAVING COUNT(*)>0;
END//

DELIMITER //
CREATE PROCEDURE medicine_ended ()
BEGIN
        SELECT s.NameOfMedicine, TypeOfMedicine, Count, CriticalCount FROM
        storagemedicine s JOIN guideofpreparation g ON s.NameOfMedicine = g.NameOfMedicine
    WHERE (Count<=CriticalCount)
    GROUP BY s.NameOfMedicine
    HAVING COUNT(*)>0;
END//
```

Task7

```
DELIMITER //
CREATE PROCEDURE min_count_component()
BEGIN
        SELECT Component, Count FROM storagecomponent WHERE (Count = (SELECT
        MIN(Count) FROM storagecomponent));
END//

DELIMITER //
CREATE PROCEDURE min_count_med()
BEGIN
        SELECT NameOfMedicine, Count FROM storagemedicine WHERE (Count = (SELECT
        MIN(Count) FROM storagemedicine));
END//
```

```
DELIMITER //
CREATE PROCEDURE min_count_med_category(IN med_type VARCHAR(45))
BEGIN
        DECLARE min INT;
   SET min = (SELECT MIN(Count) FROM storagemedicine s JOIN guideofpreparation g ON
        s.NameOfMedicine = g.NameOfMedicine WHERE  (TypeOfMedicine = med_type));
        SELECT s.NameOfMedicine, TypeOfMedicine, Count FROM storagemedicine s JOIN
        guideofpreparation g ON s.NameOfMedicine = g.NameOfMedicine
   WHERE (Count = min) AND (TypeOfMedicine = med_type);
END//
```

Task8

```
DELIMITER //
CREATE PROCEDURE order_status(IN or_status VARCHAR(45))
BEGIN
        SELECT FullName, Phone, Adress FROM orderbook WHERE OrderStatus = or_status
        UNION
        SELECT 'Total Number:', '', COUNT(*) FROM orderbook WHERE OrderStatus = or_status;
END//
```

Task9

```
CREATE PROCEDURE request_med_inprogress()
BEGIN
        SELECT NameOfMedicine, OrderStatus FROM requestformedicine r JOIN orderbook o ON
        r.idOrder = o.idOrder
   WHERE OrderStatus = 'InProgress'
   UNION
   SELECT 'Total Number:', COUNT(*) FROM requestformedicine r JOIN orderbook o ON
        r.idOrder = o.idOrder
   WHERE OrderStatus = 'InProgress';
END//
```

Task10

```
CREATE PROCEDURE technology_type(IN med_type VARCHAR(45))
BEGIN
        SELECT NameOfMedicine, TypeOfMedicine, Preparation, Component, CountOfComponent
   FROM guideofpreparation g JOIN (componentinmedicine i JOIN storagecomponent s ON
        i.idComponent = s.idComponent) ON g.ID=i.idMedicine
   WHERE TypeOfMedicine = med_type
   UNION
   SELECT NameOfMedicine, TypeOfMedicine, Preparation, '', ''
   FROM guideofpreparation WHERE Preparation = 'Done' AND TypeOfMedicine = med_type;
END//

DELIMITER //
CREATE PROCEDURE technology_name(IN med_name VARCHAR(45))
BEGIN
        SELECT NameOfMedicine, TypeOfMedicine, Preparation, Component, CountOfComponent
   FROM guideofpreparation g JOIN (componentinmedicine i JOIN storagecomponent s ON
        i.idComponent = s.idComponent) ON g.ID=i.idMedicine
```

```
   WHERE NameOfMedicine = med_name
   UNION
   SELECT NameOfMedicine, TypeOfMedicine, Preparation, '', ''
   FROM guideofpreparation WHERE Preparation = 'Done' AND NameOfMedicine = med_name;
END//

DELIMITER //
CREATE PROCEDURE technology_inprogress()
BEGIN
        SELECT NameOfMedicine, TypeOfMedicine, Preparation, Component,
        CountOfComponent, OrderStatus
   FROM (recipes r JOIN orderbook o ON r.PatientFullName = o.FullName) JOIN
   (guideofpreparation g JOIN (componentinmedicine i JOIN storagecomponent s ON
        i.idComponent = s.idComponent) ON g.ID=i.idMedicine)
   ON r.Medicine = g.NameOfMedicine
   WHERE OrderStatus = 'InProgress'
        UNION
        SELECT NameOfMedicine, TypeOfMedicine, Preparation, '', '', OrderStatus
   FROM (recipes r JOIN orderbook o ON r.idRecipe = o.idOrder) JOIN guideofpreparation g ON
        r.Medicine = g.NameOfMedicine
   WHERE OrderStatus = 'InProgress' AND Preparation = 'Done';
END//
```

Task11

```
CREATE PROCEDURE med_price(IN med VARCHAR(45))
BEGIN
        IF med = (SELECT NameOfMedicine FROM storagemedicine WHERE NameOfMedicine =
        med) THEN
                SELECT NameOfMedicine, Cost FROM storagemedicine WHERE
        NameOfMedicine = med;
        ELSE
                SELECT Component, CountOfComponent, Cost
      FROM guideofpreparation g JOIN (storagecomponent s JOIN componentinmedicine i ON
        s.idComponent = i.idComponent) ON g.ID = i.idMedicine
      WHERE NameOfMedicine = med
      UNION
      SELECT CONCAT('**Total of ',NameOfMedicine),  SUM(CountOfComponent),
        SUM(Cost*CountOfComponent)
      FROM guideofpreparation g JOIN (storagecomponent s JOIN componentinmedicine i ON
        s.idComponent = i.idComponent) ON g.ID = i.idMedicine
      WHERE NameOfMedicine = med;
        END IF;
END//
```

Task12

```
CREATE PROCEDURE top_orders()
BEGIN
        DECLARE max, min, count INT;
   SELECT COUNT(FullName) INTO max FROM orderbook GROUP BY FullName ORDER BY
        COUNT(FullName) DESC limit 1;
```

```
    SELECT COUNT(FullName) INTO min FROM orderbook GROUP BY FullName ORDER BY
        COUNT(FullName) limit 1;
    SET count = (min + max) / 2;

        SELECT *, COUNT(FullName) AS 'Count Of Orders' FROM orderbook
    GROUP BY FullName HAVING COUNT(FullName) >= count
    ORDER BY COUNT(FullName) DESC;
END//

DELIMITER //
CREATE PROCEDURE order_of_type(IN med_type VARCHAR(45))
BEGIN
        SELECT idOrder, FullName, Phone, Adress, Medicine, OrderStatus FROM orderbook o
        JOIN (recipes r JOIN guideofpreparation g ON r.Medicine = g.NameOfMedicine) ON
        o.idOrder=r.idRecipe
    WHERE TypeOfMedicine = med_type;
END//

DELIMITER //
CREATE PROCEDURE order_of_med(IN med VARCHAR(45))
BEGIN
        SELECT idOrder, FullName, Phone, Adress, Medicine, OrderStatus FROM orderbook o
        JOIN recipes r ON o.idOrder=r.idRecipe
    WHERE Medicine = med;
END//
```

Task13

```
CREATE PROCEDURE info_med(IN med VARCHAR(45))
BEGIN
        SELECT NameOfMedicine as 'Name', TypeOfMedicine as 'Type', Preparation, Component,
        (Cost*CountOfComponent) as Cost, Count
    FROM guideofpreparation g JOIN (storagecomponent s JOIN componentinmedicine i ON
        s.idComponent = i.idComponent)
    ON g.ID = i.idMedicine WHERE NameOfMedicine = med
    UNION
    SELECT s.NameOfMedicine, TypeOfMedicine, Preparation, '-', Cost, Count
    FROM guideofpreparation g JOIN storagemedicine s ON g.NameOfMedicine =
        s.NameOfMedicine
    WHERE s.NameOfMedicine = med
    UNION
    SELECT 'Total:', ",", ", SUM(Cost*CountOfComponent), SUM(Count)
        FROM guideofpreparation g JOIN (storagecomponent s JOIN componentinmedicine i ON
        s.idComponent = i.idComponent)
    ON g.ID = i.idMedicine WHERE NameOfMedicine = med;
END//
```

Addition

```
DELIMITER $$
CREATE PROCEDURE clean_up()
```

```
BEGIN
        DELETE FROM recipes WHERE Seal <> 'true';
    DELETE FROM recipes WHERE Medicine IN(SELECT NameOfMedicine FROM
        guideofpreparation WHERE TypeOfMedicine = 'ointment') AND
        (WayOfUsing<>'external');
        DELETE FROM recipes WHERE Medicine IN(SELECT NameOfMedicine
        FROM guideofpreparation WHERE TypeOfMedicine = 'mixture' OR
        TypeOfMedicine = 'powders') AND (WayOfUsing<>'internal');
    DELETE FROM recipes WHERE Medicine IN(SELECT NameOfMedicine FROM
        guideofpreparation WHERE TypeOfMedicine = 'solutions') AND
        (WayOfUsing<>'external')AND(WayOfUsing<>'internal')AND(WayOfUsing<
        >'mix');
END$$

CREATE PROCEDURE upd_orderbook()
BEGIN
        UPDATE orderbook, recipes SET orderbook.OrderStatus = 'Done' WHERE
        (recipes.DateToBack < current_date()) AND
        (orderbook.OrderStatus<>'Completed') AND (orderbook.idOrder =
        recipes.idRecipe);
    UPDATE orderbook, recipes SET orderbook.OrderStatus = 'InProgress' WHERE
        (recipes.DateToBack > current_date()) AND
        (orderbook.OrderStatus<>'Done')AND
        (orderbook.OrderStatus<>'WaitForComponent') AND (orderbook.idOrder =
        recipes.idRecipe);
END$$

CREATE PROCEDURE check_storage()
BEGIN
        UPDATE storagemedicine SET Count = 0 WHERE ShelfLife < current_date();
    UPDATE storagecomponent SET Count = 0 WHERE ShelfLife < current_date();
END$$
```

3.2.3 Тригери

```
CREATE TRIGGER sale_medicine AFTER UPDATE ON storagemedicine
FOR EACH ROW
BEGIN
        IF new.Count <= new.CriticalCount THEN
                INSERT INTO requestformedicine(NameOfMedicine, Count)
        VALUES(new.NameOfMedicine, new.CriticalCount*5);
        END IF;
END//
```

```
DELIMITER //
CREATE TRIGGER sale_component AFTER UPDATE ON storagecomponent
FOR EACH ROW
BEGIN
        IF new.Count <= new.CriticalCount THEN
                INSERT INTO requestformedicine(NameOfMedicine, Count)
        VALUES(new.Component, new.CriticalCount*3);
        END IF;
END//

DELIMITER //
CREATE TRIGGER supply_medicine AFTER DELETE ON requestformedicine
FOR EACH ROW
BEGIN
        UPDATE storagemedicine SET Count = Count + old.Count WHERE
        NameOfMedicine = old.NameOfMedicine;
    UPDATE storagecomponent SET Count = Count + old.Count WHERE Component
        = old.NameOfMedicine;
    UPDATE storagemedicine SET ShelfLife = DATE_ADD(ShelfLife, INTERVAL 5
        YEAR) WHERE NameOfMedicine = old.NameOfMedicine;
    UPDATE storagecomponent SET ShelfLife = DATE_ADD(ShelfLife, INTERVAL
        5 YEAR) WHERE Component = old.NameOfMedicine;

END//

DELIMITER //
CREATE TRIGGER add_to_ledger AFTER insert ON recipes
FOR EACH ROW
        BEGIN
    IF new.Medicine = (SELECT Medicine FROM ledger WHERE ledger.Medicine =
        new.Medicine) THEN
                UPDATE ledger
       SET Sales = Sales + new.CountOfMedicine
       WHERE Medicine = new.Medicine;

        ELSE
                INSERT INTO ledger VALUES (new.Medicine, new.CountOfMedicine);
    END IF;
END//

DELIMITER //
CREATE TRIGGER write_off BEFORE insert ON recipes
FOR EACH ROW
```

```
    BEGIN
IF new.Seal='true' THEN
IF new.Medicine = (SELECT NameOfMedicine FROM storagemedicine WHERE
    NameOfMedicine = new.Medicine) THEN
        IF new.CountOfMedicine <= (SELECT Count FROM storagemedicine
    WHERE NameOfMedicine = new.Medicine) THEN
            SET new.DateToBack =
    DATE_ADD(new.RecipeDate,INTERVAL (SELECT TimeOfPreparation
    FROM guideofpreparation WHERE NameOfMedicine=new.Medicine) DAY);
            UPDATE storagemedicine SET Count = Count -
    new.CountOfMedicine WHERE NameOfMedicine = new.Medicine;
     INSERT INTO orderbook(idOrder, FullName) VALUES(new.idRecipe,
    new.PatientFullName);
        ELSE
            SET new.DateToBack =
    DATE_ADD(new.RecipeDate,INTERVAL (SELECT TimeOfPreparation
    FROM guideofpreparation WHERE NameOfMedicine=new.Medicine) + 20
    DAY);
            INSERT INTO orderbook(idOrder, FullName, OrderStatus)
    VALUES(new.idRecipe, new.PatientFullName,'WaitForComponent');
            INSERT INTO requestformedicine(NameOfMedicine, Count)
    VALUES(new.Medicine, new.CountOfMedicine);
        END IF;
    ELSE
        IF    0 > ANY(SELECT Count -
    CountOfComponent*new.CountOfMedicine FROM
            storagecomponent s JOIN componentinmedicine c ON
    s.idComponent=c.idComponent
     JOIN guideofpreparation g ON g.ID=c.idMedicine WHERE
    g.NameOfMedicine=new.Medicine AND Count -
    CountOfComponent*new.CountOfMedicine<0) THEN

            SET new.DateToBack = DATE_ADD(new.RecipeDate,
    INTERVAL (SELECT TimeOfPreparation FROM guideofpreparation WHERE
    NameOfMedicine=new.Medicine) + 25 DAY);
            INSERT INTO orderbook(idOrder, FullName, OrderStatus)
    VALUES(new.idRecipe, new.PatientFullName,'WaitForComponent');
            INSERT INTO requestformedicine(NameOfMedicine, Count)
    SELECT Component, CountOfComponent*new.CountOfMedicine
     FROM (storagecomponent s JOIN componentinmedicine c ON s.idComponent
    = c.idComponent) JOIN guideofpreparation g ON g.ID=c.idMedicine
     WHERE Count - CountOfComponent*new.CountOfMedicine <0 AND
    NameOfMedicine = new.Medicine;
```

```
        ELSE
          SET new.DateToBack = DATE_ADD(new.RecipeDate,INTERVAL (SELECT
        TimeOfPreparation FROM guideofpreparation WHERE
        NameOfMedicine=new.Medicine) DAY);
          UPDATE storagecomponent, componentinmedicine, guideofpreparation
          SET storagecomponent.Count  = storagecomponent.Count -
        componentinmedicine.CountOfComponent*new.CountOfMedicine
          WHERE guideofpreparation.NameOfMedicine=new.Medicine AND
        componentinmedicine.idMedicine=guideofpreparation.ID AND
        storagecomponent.idComponent=componentinmedicine.idComponent;
                        INSERT INTO orderbook(idOrder, FullName)
        VALUES(new.idRecipe, new.PatientFullName);
      END IF;
        END IF;
    END IF;
END//
```

3.3 Вимоги до апаратних і програмних засобів

В користувача повинно бути інстальовано MySQL Workbench та MySQL Server.

3.4 Випробування розроблених програм

FullName	Phone	Adress
Hazar Shadoglu	156256	Kasak st. 1
Jules Kob	626162	Why st. 12
Count Of Patients:		2

рис. 1 запит №1

FullName	Phone	Adress
John Snow	124569	Doup st. 14
Count of patients:		1

FullName	Phone	Adress
Count of patients waiting	ointment	0

рис. 2 запит №2

Medicine	Sales
Barvoval	5
Korvalol	3
Ibuprofen	2
Kashtan	2
Septolete	2
Angilex	1
Fastum	1
Velon	1

рис. 3 запит №3

	COUNT
▶	32

рис. 4 запит №4

	PatientFullName	RecipeDate	Medicine
▶	Fillip Morris	2020-04-09	Fastum
	Total number:		1

	PatientFullName	RecipeDate	TypeOfMedicine
▶	Janifer Lopez	2020-04-07	tablets
	Total number:		1

рис. 5 запит №5

	NameOfMedicine	TypeOfMedicine	Count	CriticalCount
▶	Septolete	tablets	2	2

рис. 6 запит №6

	Component	Count
▶	Arbasolid	6

	NameOfMedicine	TypeOfMedicine	Count
▶	Septolete	tablets	2

рис. 7 запит №7

	FullName	Phone	Adress
▶	Ellen Ellite	465132	Cherkasy st. 13
	Total Number:		1

рис. 8 запит №8

	NameOfMedicine	OrderStatus
▶	Korvalol	InProgress
	Total Number:	1

рис. 9 запит №9

	NameOfMedicine	TypeOfMedicine	Preparation	Component	CountOfComponent
▶	Velon	ointment	Filtration	Peppermint oil	2
	Velon	ointment	Filtration	Arbasolid	5
	Velon	ointment	Filtration	Nimesulid	4
	Fastum	ointment	Done		
	Romashka	ointment	Done		

	NameOfMedicine	TypeOfMedicine	Preparation	Component	CountOfComponent
▶	Korvalol	tincture	Done		

	NameOfMedicine	TypeOfMedicine	Preparation	Component	CountOfComponent	OrderStatus
▶	Kashtan	solutions	Filtration	Chamomile	2	InProgress
	Kashtan	solutions	Filtration	Cetirizine	5	InProgress
	Kashtan	solutions	Filtration	Peppermint oil	2	InProgress
	Kashtan	solutions	Filtration	Fir oil	3	InProgress

рис. 10 запит №10

	NameOfMedicine	Cost
▶	Korvalol	75

	Component	CountOfComponent	Cost
▶	Nimesulid	5	5
	Arbasolid	2	4
	Cetirizine	4	7
	Fir oil	6	3
	**Total of Barvoval	17	79

рис. 11 запит №11

	idOrder	FullName	Phone	Adress	OrderStatus	Count Of Orders
▶	1	Jay Pritchet	502040	Canon st. 7	Completed	1
	3	John Snow	124569	Doup st. 14	WhiteForOrder	1
	2	Caroline Fox	105869	Karantine st....	Completed	1
	7	Jules Kob	626162	Why st. 12	Done	1
	8	Ellen Ellite	465132	Cherkasy st....	InProgress	1
	4	Miran Oslanbey	258933	Treory st. 78b	Completed	1
	5	Hazar Shadoglu	156256	Kasak st. 1	Done	1

	idOrder	FullName	Phone	Adress	Medicine	OrderStatus
▶	5	Hazar Shadoglu	156256	Kasak st. 1	Fastum	Done
	4	Miran Oslanbey	258933	Treory st. 78b	Velon	Completed

	idOrder	FullName	Phone	Adress	Medicine	OrderStatus
▶	1	Jay Pritchet	502040	Canon st. 7	Korvalol	Completed

рис. 12 запит №12

	Name	Type	Preparation	Component	Cost	Count
▶	Barvoval	tincture	Filtration	Nimesulid	25	24
	Barvoval	tincture	Filtration	Arbasolid	8	6
	Barvoval	tincture	Filtration	Cetirizine	28	16
	Barvoval	tincture	Filtration	Fir oil	18	35
	Total:				79	81

рис. 13 запит №13

Додаткове:

idOrder	FullName	Phone	Adress	OrderStatus
1	Jay Pritchet	502040	Canon st. 7	Done
2	Caroline Fox	105869	Karantine st....	Completed
3	John Snow	124569	Doup st. 14	InProgress
4	Miran Oslanbey	258933	Treory st. 78b	Done
5	Hazar Shadoglu	156256	Kasak st. 1	Done
7	Jules Kob	626162	Why st. 12	Done
8	Ellen Ellite	465132	Cherkasy st....	InProgress
NULL	NULL	NULL	NULL	NULL

рис. 14 запит оновлення статусу замовлення

idRecipe	DoctorFullName	Signature	Seal	PatientFullName	PatientAge	PatientDiagnosis	Medicine	CountOfMedicine	WayOfUsing	RecipeDate	DateToBack
1	Jay Pritchet	JP	true	Kevin Spice	36	heart disease	Korvalol	3	internal	2020-04-12	2020-04-12
2	Caroline Fox	CF	true	Stan Le	70	lung cancer	Ibuprofen	2	internal	2020-05-18	2020-05-18
3	John Snow	JS	true	Dru Baltimor	25	depression	Barvoval	5	internal	2020-04-30	2020-06-06
4	Miran Oslanbey	MO	true	Frank Sinatra	80	type 2 diabetes	Velon	1	external	2020-03-06	2020-03-21
5	Hazar Shadoglu	HS	true	Fillip Morris	45	osteoporosis	Fastum	1	external	2020-04-09	2020-04-09
7	Jules Kob	JK	true	Janifer Lopez	37	oral disease	Septolete	2	external	2020-04-07	2020-04-07
8	Ellen Ellite	EE	true	Margo Fabio	90	arthritis	Kashtan	2	internal	2020-05-06	2020-05-21

рис. 15 запит видалення записів, що не відповідають вимогам

idMedicine	NameOfMedicine	Count	CriticalCount	Cost	ShelfLife
1	Ibuprofen	0	3	35	2020-05-12
2	Septolete	2	2	20	2020-05-20
3	Fastum	4	2	65	2025-02-12
4	L-Tiroxin	6	3	13	2025-06-11
5	Korvalol	5	3	75	2023-07-15
6	Romashka	8	2	51	2028-05-05
7	Analgin	3	2	23	2028-07-07

idComponent	Component	Count	ShelfLife	CriticalCount	Cost
1	Nimesulid	24	2020-12-22	3	5
2	Arbasolid	0	2015-10-22	3	4
3	Cetirizine	0	2004-08-25	3	7
4	Fir oil	0	2006-12-23	3	3
5	Peppermint oil	20	2026-11-23	3	3
6	Wild fruit carrots	0	2008-08-25	3	6
7	Alcohol	0	2007-07-24	3	6
8	Chamomile	20	2029-05-24	3	3
9	Hop cones	0	2014-07-24	3	6
10	Omeprazole	0	2016-09-23	4	5
11	Moxifloxacin	20	2028-02-25	3	6
12	Macrogol	10	2030-03-25	3	3
13	Lactose	0	2014-05-22	3	11
14	Magnesium stea...	0	2017-07-22	3	3
15	Microcrystalline ...	0	2016-05-23	3	1
16	Pentoxifylline	0	2014-12-25	3	2
17	Hypromellose	10	2030-02-25	3	6
18	Dimethicone	0	2006-06-26	3	7
19	Amlodipine	0	2009-11-25	3	5
20	Calcium dihydrate	0	2012-12-22	3	4

рис. 16 запит списання зі складів прострочених ліків

idRecipe	DoctorFullName	Signature	Seal	PatientFullName	PatientAge	PatientDiagnosis	Medicine	CountOfMedicine	WayOfUsing	RecipeDate	DateToBack
4	Miran Oslanbey	MO	true	Frank Sinatra	80	type 2 diabetes	Velon	1	external	2020-03-06	2020-03-21
5	Hazar Shadoglu	HS	true	Fillip Morris	45	osteoporosis	Fastum	1	external	2020-04-09	2020-04-09
7	Jules Kob	JK	true	Janifer Lopez	37	oral disease	Septolete	2	external	2020-04-07	2020-04-07
8	Ellen Ellite	EE	true	Margo Fabio	90	arthritis	Kashtan	2	internal	2020-05-06	2020-05-21
9	Joe Lotrulio	JL	true	Azize Aslanbey	72	osteoporos	Fastum	8	external	2020-04-20	2020-05-10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

рис. 17 робота тригера для таблиці recipes

Medicine	Sales
Angilex	1
Barvoval	5
Fastum	9
Ibuprofen	2
Kashtan	2
Korvalol	3
Septolete	2
Velon	1
NULL	NULL

рис. 18 робота тригера для таблиці ledger

idRequest	NameOfMedicine	Count
18	Wild fruit carrots	9
19	Alcohol	9
20	Hop cones	9
21	Omeprazole	12
22	Lactose	9
23	Magnesium stea...	9
24	Microcrystalline ...	9
25	Pentoxifylline	9
26	Dimethicone	9
27	Amlodipine	9
28	Calcium dihydrate	9
29	Fastum	8
NULL	NULL	NULL

рис. 19 робота тригера для таблиці requestformedicine

idMedicine	NameOfMedicine	Count	CriticalCount	Cost	ShelfLife
1	Ibuprofen	0	3	35	2020-05-12
2	Septolete	2	2	20	2020-05-20
3	Fastum	3	2	65	2025-02-12
4	L-Tiroxin	6	3	13	2025-06-11
5	Korvalol	5	3	75	2023-07-15
6	Romashka	8	2	51	2028-05-05

рис. 20 робота тригера для таблиці storagemedicine

idComponent	Component	Count	ShelfLife	CriticalCount	Cost
1	Nimesulid	49	2020-12-22	3	5
2	Arbasolid	19	2020-10-22	3	4
3	Cetirizine	9	2009-08-25	3	7
4	Fir oil	9	2020-12-23	3	3
5	Peppermint oil	20	2026-11-23	3	3
6	Wild fruit carrots	9	2020-08-25	3	6
7	Alcohol	9	2020-07-24	3	6
8	Chamomile	20	2029-05-24	3	3
9	Hop cones	9	2019-07-24	3	6
10	Omeprazole	12	2021-09-23	4	5
11	Moxifloxacin	20	2028-02-25	3	6
12	Macrogol	10	2030-03-25	3	3

рис. 21 робота тригера для таблиці storagecomponent

3.5 Опис тестової бази данных

Скриншоти таблиць

idRecipe	DoctorFullName	Signature	Seal	PatientFullName	PatientAge	PatientDiagnosis	Medicine	CountOfMedicine	WayOfUsing	RecipeDate	DateToBack
1	Jay Pritchet	JP	true	Kevin Spice	36	heart disease	Korvalol	3	internal	2020-04-12	2020-04-12
2	Caroline Fox	CF	true	Stan Le	70	lung cancer	Ibuprofen	2	internal	2020-05-18	2020-05-18
3	John Snow	JS	true	Dru Baltimor	25	depression	Barvoval	5	internal	2020-04-30	2020-06-06
4	Miran Oslanbey	MO	true	Frank Sinatra	80	type 2 diabetes	Velon	1	external	2020-03-06	2020-03-21
5	Hazar Shadoglu	HS	true	Fillip Morris	45	osteoporosis	Fastum	1	external	2020-04-09	2020-04-09
6	Stefan Salvatore	SS	false	Jack Black	19	asthma	Angilex	1	external	2020-05-05	2020-05-13
7	Jules Kob	JK	true	Janifer Lopez	37	oral disease	Septolete	2	external	2020-04-07	2020-04-07
8	Ellen Ellite	EE	true	Margo Fabio	90	arthritis	Kashtan	2	internal	2020-05-06	2020-05-21

рис. 1 таблиця recipes

idOrder	FullName	Phone	Adress	OrderStatus
1	Jay Pritchet	502040	Canon st. 7	Completed
2	Caroline Fox	105869	Karantine st....	Completed
3	John Snow	124569	Doup st. 14	InProgress
4	Miran Oslan...	258933	Treory st. 78b	Completed
5	Hazar Shado...	156256	Kasak st. 1	Done
7	Jules Kob	626162	Why st. 12	Done
8	Ellen Ellite	465132	Cherkasy st....	InProgress
NULL	NULL	NULL	NULL	NULL

рис.2 таблиця orderbook

ID	NameOfMedicine	TypeOfMedicine	Preparation	TimeOfPreparation
1	Ibuprofen	tablets	Done	0
2	Septolete	tablets	Done	0
3	Fastum	ointment	Done	0
4	L-Tiroxin	tablets	Done	0
5	Barvoval	tincture	Filtration	12
6	Korvalol	tincture	Done	0
7	Velon	ointment	Filtration	15
8	Elokom	solutions	Filtration	12
9	Romashka	ointment	Done	0
10	Kashtan	solutions	Filtration	15
11	Coldrex	powders	Mix	7
12	Miranex	tinctures	Filtration	18
13	Angilex	powders	Mix	8
14	Lorosan	powders	Mix	6
15	Analgin	mixture	Done	1
16	Nimesil	powders	Mix	10

рис.3 таблиця guideofpreparation

Medicine	Sales
Angilex	1
Barvoval	5
Fastum	1
Ibuprofen	2
Kashtan	2
Korvalol	3
Septolete	2
Venom	1

рис.4 таблиця ledger

	idRequest	NameOfMedicine	Count
▶	8	Korvalol	15
	9	Ibuprofen	15
	10	Nimesulid	25
	11	Arbasolid	10
	12	Cetirizine	20
	13	Septolete	10
*	NULL	NULL	NULL

рис.5 таблиця requestformedicine

idComponent	Component	Count	ShelfLife	CriticalCount	Cost
1	Nimesulid	24	2020-12-22	3	5
2	Arbasolid	6	2015-10-22	3	4
3	Cetirizine	6	2004-08-25	3	7
4	Fir oil	29	2006-12-23	3	3
5	Peppermint oil	16	2026-11-23	3	3
6	Wild fruit carrots	20	2008-08-25	3	6
7	Alcohol	20	2007-07-24	3	6
8	Chamomile	16	2029-05-24	3	3
9	Hop cones	20	2014-07-24	3	6
10	Omeprazole	20	2016-09-23	4	5
11	Moxifloxacin	20	2028-02-25	3	6
12	Macrogol	10	2030-03-25	3	3
13	Lactose	10	2014-05-22	3	11
14	Magnesium stea...	10	2017-07-22	3	3
15	Microcrystalline ...	10	2016-05-23	3	1
16	Pentoxifylline	10	2014-12-25	3	2
17	Hypromellose	10	2030-02-25	3	6
18	Dimethicone	10	2006-06-26	3	7
19	Amlodipine	10	2009-11-25	3	5
20	Calcium dihydrate	10	2012-12-22	3	4

рис.6 таблиця storagecomponent

idMedicine	NameOfMedicine	Count	CriticalCount	Cost	ShelfLife
1	Ibuprofen	3	3	35	2020-05-12
2	Septolete	0	2	20	2020-05-20
3	Fastum	3	2	65	2025-02-12
4	L-Tiroxin	6	3	13	2025-06-11
5	Korvalol	2	3	75	2023-07-15
6	Romashka	8	2	51	2028-05-05
7	Analgin	3	2	23	2028-07-07

рис.7 таблиця storagemedicine

idMedicine	idComponent	CountOfComponent
5	1	5
5	2	2
5	3	4
5	4	6
7	5	2
7	2	5
7	1	4
8	3	3
8	7	1
10	8	2
10	3	5
10	5	2
10	4	3
11	1	4
11	2	5
12	3	2
12	5	5
12	9	2
13	6	3
13	1	3
14	2	3
14	3	1
14	4	1
14	5	2
16	6	1
16	8	1

рис.8 таблиця componentinmedicine

4. Висновки

Було виконано проектування та розроблення бази даних для АІС Аптеки, реалізовано різні запити, що потрібні для користування БД, за допомогою тригерів та процедур. В результаті значно покращились навички користування SQL та побудови моделі БД на основі даних вимог.

5. Література

1. https://www.mysql.com/
2. “Learning SQL” By Alan Beaulieu
3. “SQL: The Ultimate Beginners Guide: Learn SQL Today” By Steve Tale

6. Додатки

6.1 Лістинг ПЗ:

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

-- -----------------------------------------------------
-- Schema pharmacy
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema pharmacy
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `pharmacy` DEFAULT CHARACTER SET utf8 ;
USE `pharmacy` ;

-- -----------------------------------------------------
-- Table `pharmacy`.`componentinmedicine`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `pharmacy`.`componentinmedicine` ;

CREATE TABLE IF NOT EXISTS `pharmacy`.`componentinmedicine` (
  `idMedicine` INT(11) NOT NULL,
  `idComponent` INT(11) NULL DEFAULT NULL,
  `CountOfComponent` INT(5) NULL DEFAULT NULL)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `pharmacy`.`guideofpreparation`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `pharmacy`.`guideofpreparation` ;

CREATE TABLE IF NOT EXISTS `pharmacy`.`guideofpreparation` (
  `ID` INT(11) NOT NULL,
  `NameOfMedicine` VARCHAR(45) NULL DEFAULT NULL,
  `TypeOfMedicine` VARCHAR(45) NULL DEFAULT NULL,
  `Preparation` VARCHAR(45) NULL DEFAULT NULL,
  `TimeOfPreparation` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`ID`))
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `pharmacy`.`ledger`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `pharmacy`.`ledger` ;

CREATE TABLE IF NOT EXISTS `pharmacy`.`ledger` (
  `Medicine` VARCHAR(45) NOT NULL,
  `Sales` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`Medicine`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `pharmacy`.`medicinestypes`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `pharmacy`.`medicinestypes` ;

CREATE TABLE IF NOT EXISTS `pharmacy`.`medicinestypes` (
  `TypeOfMedicine` VARCHAR(45) NULL DEFAULT NULL,
  `WayOfPreparation` VARCHAR(45) NULL DEFAULT NULL,
  `PreparationTime` INT(11) NULL DEFAULT NULL,
  `WayOfUsing` VARCHAR(45) NULL DEFAULT NULL)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `pharmacy`.`orderbook`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `pharmacy`.`orderbook` ;

CREATE TABLE IF NOT EXISTS `pharmacy`.`orderbook` (
  `idOrder` INT(11) NOT NULL,
  `FullName` VARCHAR(45) NOT NULL,
  `Phone` VARCHAR(45) NULL DEFAULT NULL,
  `Adress` VARCHAR(45) NULL DEFAULT NULL,
  `OrderStatus` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`idOrder`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

CREATE UNIQUE INDEX `FullName_UNIQUE` ON `pharmacy`.`orderbook`
(`FullName` ASC);


-- -----------------------------------------------------
```

```
-- Table `pharmacy`.`recipes`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `pharmacy`.`recipes` ;

CREATE TABLE IF NOT EXISTS `pharmacy`.`recipes` (
  `idRecipe` INT(11) NOT NULL,
  `DoctorFullName` VARCHAR(45) NULL DEFAULT NULL,
  `Signature` CHAR(2) NULL DEFAULT NULL,
  `Seal` VARCHAR(5) NULL DEFAULT NULL,
  `PatientFullName` VARCHAR(45) NULL DEFAULT NULL,
  `PatientAge` INT(11) NULL DEFAULT NULL,
  `PatientDiagnosis` VARCHAR(45) NULL DEFAULT NULL,
  `Medicine` VARCHAR(45) NULL DEFAULT NULL,
  `CountOfMedicine` INT(11) NULL DEFAULT NULL,
  `WayOfUsing` ENUM('internal', 'external', 'mix') NULL DEFAULT NULL,
  `RecipeDate` DATE NULL DEFAULT NULL,
  `DateToBack` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`idRecipe`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `pharmacy`.`requestformedicine`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `pharmacy`.`requestformedicine` ;

CREATE TABLE IF NOT EXISTS `pharmacy`.`requestformedicine` (
  `idRequest` INT(11) NOT NULL AUTO_INCREMENT,
  `NameOfMedicine` VARCHAR(45) NULL DEFAULT NULL,
  `Count` INT(11) NULL DEFAULT NULL,
  PRIMARY KEY (`idRequest`))
ENGINE = InnoDB
AUTO_INCREMENT = 30
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `pharmacy`.`storagecomponent`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `pharmacy`.`storagecomponent` ;

CREATE TABLE IF NOT EXISTS `pharmacy`.`storagecomponent` (
  `idComponent` INT(11) NOT NULL,
  `Component` VARCHAR(45) NULL DEFAULT NULL,
  `Count` INT(11) NULL DEFAULT NULL,
  `ShelfLife` DATE NULL DEFAULT NULL,
  `CriticalCount` INT(10) UNSIGNED NOT NULL,
  `Cost` DECIMAL(2,0) NULL DEFAULT NULL,
  PRIMARY KEY (`idComponent`))
```

```
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


-- -----------------------------------------------------
-- Table `pharmacy`.`storagemedicine`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `pharmacy`.`storagemedicine` ;

CREATE TABLE IF NOT EXISTS `pharmacy`.`storagemedicine` (
  `idMedicine` INT(11) NOT NULL,
  `NameOfMedicine` VARCHAR(45) NULL DEFAULT NULL,
  `Count` INT(11) NULL DEFAULT NULL,
  `CriticalCount` INT(11) NULL DEFAULT NULL,
  `Cost` DECIMAL(10,0) NULL DEFAULT NULL,
  `ShelfLife` DATE NULL DEFAULT NULL,
  PRIMARY KEY (`idMedicine`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

USE `pharmacy` ;

-- -----------------------------------------------------
-- procedure check_storage
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`check_storage`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `check_storage`()
BEGIN
    UPDATE storagemedicine SET Count = 0 WHERE ShelfLife < current_date();
   UPDATE storagecomponent SET Count = 0 WHERE ShelfLife < current_date();
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure clean_up
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`clean_up`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `clean_up`()
BEGIN
```

```
    DELETE FROM recipes WHERE Seal <> 'true';
   DELETE FROM recipes WHERE Medicine IN(SELECT NameOfMedicine FROM
guideofpreparation WHERE TypeOfMedicine = 'ointment') AND
(WayOfUsing<>'external');
    DELETE FROM recipes WHERE Medicine IN(SELECT NameOfMedicine FROM
guideofpreparation WHERE TypeOfMedicine = 'mixture' OR TypeOfMedicine =
'powders') AND (WayOfUsing<>'internal');
   DELETE FROM recipes WHERE Medicine IN(SELECT NameOfMedicine FROM
guideofpreparation WHERE TypeOfMedicine = 'solutions') AND
(WayOfUsing<>'external')AND(WayOfUsing<>'internal')AND(WayOfUsing<>'mix');
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure component_ended
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`component_ended`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `component_ended`()
BEGIN
    SELECT Component, Count, CriticalCount FROM storagecomponent WHERE
(Count<=CriticalCount)
   GROUP BY Component
   HAVING COUNT(*)>0;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure count_of_medicine
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`count_of_medicine`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `count_of_medicine`(IN comp
VARCHAR(45), IN date1 DATE, IN date2 DATE)
BEGIN
    DECLARE id_comp INT;
   SELECT idComponent INTO id_comp FROM StorageComponent WHERE
Component = comp;

    SELECT sum(c.CountOfComponent * r.CountOfMedicine) as 'COUNT'
```

```
   FROM (componentinmedicine c JOIN guideofpreparation g on c.idMedicine = g.ID)
JOIN recipes r ON r.Medicine = g.NameOfMedicine
   WHERE (r.RecipeDate BETWEEN date1 AND date2) AND (c.idComponent =
id_comp);

END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure get_info_wait_for_order
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`get_info_wait_for_order`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_info_wait_for_order`(IN
type_med VARCHAR(45))
BEGIN
   SELECT FullName,Phone,Adress FROM (orderbook o JOIN recipes r ON o.idOrder
= r.idRecipe) JOIN guideofpreparation g on g.NameOfMedicine = r.Medicine
   WHERE OrderStatus = 'WaitForComponent' AND TypeOfMedicine = type_med
   UNION
   SELECT 'Count of patients waiting', type_med, COUNT(*) FROM (orderbook o
JOIN recipes r ON o.idOrder = r.idRecipe) JOIN guideofpreparation g on
g.NameOfMedicine = r.Medicine
   WHERE OrderStatus = 'WaitForComponent' AND TypeOfMedicine = type_med;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure get_info_wait_order
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`get_info_wait_order`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_info_wait_order`()
BEGIN
   SELECT FullName,Phone,Adress FROM orderbook WHERE OrderStatus =
'WaitForComponent'
   UNION
   SELECT 'Count of patients:', '', COUNT(*) FROM orderbook WHERE OrderStatus =
'WaitForComponent';
END$$
```

```
DELIMITER ;

-- -----------------------------------------------------
-- procedure info_med
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`info_med`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `info_med`(IN med
VARCHAR(45))
BEGIN
    SELECT NameOfMedicine as 'Name', TypeOfMedicine as 'Type', Preparation,
Component, (Cost*CountOfComponent) as Cost, Count
   FROM guideofpreparation g JOIN (storagecomponent s JOIN componentinmedicine i
ON s.idComponent = i.idComponent)
   ON g.ID = i.idMedicine WHERE NameOfMedicine = med
   UNION
   SELECT s.NameOfMedicine, TypeOfMedicine, Preparation, '-', Cost, Count
   FROM guideofpreparation g JOIN storagemedicine s ON g.NameOfMedicine =
s.NameOfMedicine
   WHERE s.NameOfMedicine = med
   UNION
   SELECT 'Total:', ",", '', SUM(Cost*CountOfComponent), SUM(Count)
    FROM guideofpreparation g JOIN (storagecomponent s JOIN componentinmedicine i
ON s.idComponent = i.idComponent)
   ON g.ID = i.idMedicine WHERE NameOfMedicine = med;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure med_price
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`med_price`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `med_price`(IN med
VARCHAR(45))
BEGIN
    IF med = (SELECT NameOfMedicine FROM storagemedicine WHERE
NameOfMedicine = med) THEN
            SELECT NameOfMedicine, Cost FROM storagemedicine WHERE
NameOfMedicine = med;
```

```
    ELSE
            SELECT Component, CountOfComponent, Cost
      FROM guideofpreparation g JOIN (storagecomponent s JOIN
componentinmedicine i ON s.idComponent = i.idComponent) ON g.ID = i.idMedicine
      WHERE NameOfMedicine = med
      UNION
      SELECT CONCAT('**Total of ',NameOfMedicine),  SUM(CountOfComponent),
SUM(Cost*CountOfComponent)
      FROM guideofpreparation g JOIN (storagecomponent s JOIN
componentinmedicine i ON s.idComponent = i.idComponent) ON g.ID = i.idMedicine
      WHERE NameOfMedicine = med;
    END IF;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure medicine_ended
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`medicine_ended`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `medicine_ended`()
BEGIN
    SELECT s.NameOfMedicine, TypeOfMedicine, Count, CriticalCount FROM
storagemedicine s JOIN guideofpreparation g ON s.NameOfMedicine =
g.NameOfMedicine
   WHERE (Count<=CriticalCount)
   GROUP BY s.NameOfMedicine
   HAVING COUNT(*)>0;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure min_count_component
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`min_count_component`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `min_count_component`()
BEGIN
    SELECT Component, Count FROM storagecomponent WHERE (Count = (SELECT
MIN(Count) FROM storagecomponent));
```

```
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure min_count_med
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`min_count_med`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `min_count_med`()
BEGIN
    SELECT NameOfMedicine, Count FROM storagemedicine WHERE (Count =
(SELECT MIN(Count) FROM storagemedicine));
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure min_count_med_category
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`min_count_med_category`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `min_count_med_category`(IN
med_type VARCHAR(45))
BEGIN
    DECLARE min INT;
   SET min = (SELECT MIN(Count) FROM storagemedicine s JOIN guideofpreparation
g ON s.NameOfMedicine = g.NameOfMedicine WHERE  (TypeOfMedicine =
med_type));
    SELECT s.NameOfMedicine, TypeOfMedicine, Count FROM storagemedicine s
JOIN guideofpreparation g ON s.NameOfMedicine = g.NameOfMedicine
   WHERE (Count = min) AND (TypeOfMedicine = med_type);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure order_of_med
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`order_of_med`;
```

```
DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `order_of_med`(IN med
VARCHAR(45))
BEGIN
    SELECT idOrder, FullName, Phone, Adress, Medicine, OrderStatus FROM
orderbook o JOIN recipes r ON o.idOrder=r.idRecipe
   WHERE Medicine = med;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure order_of_type
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`order_of_type`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `order_of_type`(IN med_type
VARCHAR(45))
BEGIN
    SELECT idOrder, FullName, Phone, Adress, Medicine, OrderStatus FROM
orderbook o JOIN (recipes r JOIN guideofpreparation g ON r.Medicine =
g.NameOfMedicine) ON o.idOrder=r.idRecipe
   WHERE TypeOfMedicine = med_type;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure order_status
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`order_status`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `order_status`(IN or_status
VARCHAR(45))
BEGIN
    SELECT FullName, Phone, Adress FROM orderbook WHERE OrderStatus =
or_status
    UNION
    SELECT 'Total Number:', '', COUNT(*) FROM orderbook WHERE OrderStatus =
or_status;
```

```
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure patients_list_medicine
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`patients_list_medicine`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `patients_list_medicine`(IN
med VARCHAR(45), IN date1 DATE, IN date2 DATE)
BEGIN
    SELECT PatientFullName, RecipeDate, Medicine FROM recipes WHERE
(RecipeDate BETWEEN date1 AND date2) AND (Medicine = med)
   UNION
   SELECT 'Total number:', '', COUNT(*) FROM recipes WHERE (RecipeDate
BETWEEN date1 AND date2) AND (Medicine = med);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure patients_list_type
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`patients_list_type`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `patients_list_type`(IN
med_type VARCHAR(45), IN date1 DATE, IN date2 DATE)
BEGIN
    SELECT PatientFullName, RecipeDate, TypeOfMedicine FROM (recipes r JOIN
guideofpreparation g ON r.Medicine = g.NameOfMedicine) WHERE (r.RecipeDate
BETWEEN date1 AND date2) AND (TypeOfMedicine = med_type)
   UNION
   SELECT 'Total number:', '', COUNT(*) FROM recipes r JOIN guideofpreparation g
ON r.Medicine = g.NameOfMedicine WHERE (r.RecipeDate BETWEEN date1 AND
date2) AND (TypeOfMedicine = med_type);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure request_med_inprogress
```

```
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`request_med_inprogress`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `request_med_inprogress`()
BEGIN
    SELECT NameOfMedicine, OrderStatus FROM requestformedicine r JOIN
orderbook o ON r.idRequest = o.idOrder
   WHERE OrderStatus = 'InProgress'
   UNION
   SELECT 'Total Number:', COUNT(*) FROM requestformedicine r JOIN orderbook o
ON r.idRequest = o.idOrder
   WHERE OrderStatus = 'InProgress';
END$$

DELIMITER ;

-- -----------------------------------------------------
-- function sum_of_comp
-- -----------------------------------------------------

USE `pharmacy`;
DROP function IF EXISTS `pharmacy`.`sum_of_comp`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` FUNCTION `sum_of_comp`(med
VARCHAR(45)) RETURNS int(11)
BEGIN
    DECLARE s INT;
   SELECT SUM(Count) INTO s FROM (componentinmedicine c JOIN
storagecomponent s ON c.idComponent = s.idComponent) JOIN guideofpreparation g
ON g.ID=c.idMedicine WHERE NameOfMedicine = med;
    return s;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure technology_inprogress
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`technology_inprogress`;

DELIMITER $$
USE `pharmacy`$$
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `technology_inprogress`()
BEGIN
    SELECT NameOfMedicine, TypeOfMedicine, Preparation, Component,
CountOfComponent, OrderStatus
   FROM (recipes r JOIN orderbook o ON r.idRecipe = o.idOrder) JOIN
   (guideofpreparation g JOIN (componentinmedicine i JOIN storagecomponent s ON
i.idComponent = s.idComponent) ON g.ID=i.idMedicine)
   ON r.Medicine = g.NameOfMedicine
   WHERE OrderStatus = 'InProgress'
    UNION
    SELECT NameOfMedicine, TypeOfMedicine, Preparation, '', '', OrderStatus
   FROM (recipes r JOIN orderbook o ON r.idRecipe = o.idOrder) JOIN
guideofpreparation g ON r.Medicine = g.NameOfMedicine
   WHERE OrderStatus = 'InProgress' AND Preparation = 'Done';
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure technology_name
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`technology_name`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `technology_name`(IN
med_name VARCHAR(45))
BEGIN
    SELECT NameOfMedicine, TypeOfMedicine, Preparation, Component,
CountOfComponent
   FROM guideofpreparation g JOIN (componentinmedicine i JOIN storagecomponent s
ON i.idComponent = s.idComponent) ON g.ID=i.idMedicine
   WHERE NameOfMedicine = med_name
   UNION
   SELECT NameOfMedicine, TypeOfMedicine, Preparation, '', ''
   FROM guideofpreparation WHERE Preparation = 'Done' AND NameOfMedicine =
med_name;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure technology_type
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`technology_type`;
```

```
DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `technology_type`(IN med_type
VARCHAR(45))
BEGIN
    SELECT NameOfMedicine, TypeOfMedicine, Preparation, Component,
CountOfComponent
   FROM guideofpreparation g JOIN (componentinmedicine i JOIN storagecomponent s
ON i.idComponent = s.idComponent) ON g.ID=i.idMedicine
   WHERE TypeOfMedicine = med_type
   UNION
   SELECT NameOfMedicine, TypeOfMedicine, Preparation, '', ''
   FROM guideofpreparation WHERE Preparation = 'Done' AND TypeOfMedicine =
med_type;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure top_medicine
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`top_medicine`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `top_medicine`(IN category
VARCHAR(45))
BEGIN
    SELECT Medicine, Sales from ledger, GuideOfPreparation
   WHERE GuideOfPreparation.TypeOfMedicine = category AND
GuideOfPreparation.NameOfMedicine = ledger.Medicine
   order by Sales DESC limit 10;

   SELECT * from ledger order by Sales DESC limit 10;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure top_orders
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`top_orders`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `top_orders`()
```

```
BEGIN
    DECLARE max, min, count INT;
   SELECT COUNT(FullName) INTO max FROM orderbook GROUP BY FullName
ORDER BY COUNT(FullName) DESC limit 1;
   SELECT COUNT(FullName) INTO min FROM orderbook GROUP BY FullName
ORDER BY COUNT(FullName) limit 1;
   SET count = (min + max) / 2;
    SELECT *, COUNT(FullName) AS 'Count Of Orders' FROM orderbook
   GROUP BY FullName HAVING COUNT(FullName) >= count
   ORDER BY COUNT(FullName) DESC;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure upd_orderbook
-- -----------------------------------------------------

USE `pharmacy`;
DROP procedure IF EXISTS `pharmacy`.`upd_orderbook`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `upd_orderbook`()
BEGIN
    UPDATE orderbook, recipes SET orderbook.OrderStatus = 'Done' WHERE
(recipes.DateToBack < current_date()) AND (orderbook.OrderStatus<>'Completed')
AND (orderbook.idOrder = recipes.idRecipe);
   UPDATE orderbook, recipes SET orderbook.OrderStatus = 'InProgress' WHERE
(recipes.DateToBack > current_date()) AND (orderbook.OrderStatus<>'Done')AND
(orderbook.OrderStatus<>'WaitForComponent') AND (orderbook.idOrder =
recipes.idRecipe);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- function write_off_comp
-- -----------------------------------------------------

USE `pharmacy`;
DROP function IF EXISTS `pharmacy`.`write_off_comp`;

DELIMITER $$
USE `pharmacy`$$
CREATE DEFINER=`root`@`localhost` FUNCTION `write_off_comp`(med
VARCHAR(45), count INT) RETURNS int(11)
BEGIN
    DECLARE s INT;
```

```
    SELECT SUM(count*CountOfComponent) INTO s FROM componentinmedicine c
JOIN guideofpreparation g ON c.idMedicine = g.ID WHERE g.NameOfMedicine =
med;
   return s;
END$$

DELIMITER ;
USE `pharmacy`;

DELIMITER $$

USE `pharmacy`$$
DROP TRIGGER IF EXISTS `pharmacy`.`add_to_ledger` $$
USE `pharmacy`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `pharmacy`.`add_to_ledger`
AFTER INSERT ON `pharmacy`.`recipes`
FOR EACH ROW
BEGIN
   IF new.Medicine = (SELECT Medicine FROM ledger WHERE ledger.Medicine =
new.Medicine) THEN
            UPDATE ledger
      SET Sales = Sales + new.CountOfMedicine
      WHERE Medicine = new.Medicine;

    ELSE
            INSERT INTO ledger VALUES (new.Medicine, new.CountOfMedicine);
   END IF;
END$$


USE `pharmacy`$$
DROP TRIGGER IF EXISTS `pharmacy`.`write_off` $$
USE `pharmacy`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `pharmacy`.`write_off`
BEFORE INSERT ON `pharmacy`.`recipes`
FOR EACH ROW
BEGIN
   IF new.Seal='true' THEN
   IF new.Medicine = (SELECT NameOfMedicine FROM storagemedicine WHERE
NameOfMedicine = new.Medicine) THEN
            IF new.CountOfMedicine <= (SELECT Count FROM storagemedicine
WHERE NameOfMedicine = new.Medicine) THEN
                  SET new.DateToBack = DATE_ADD(new.RecipeDate,INTERVAL
(SELECT TimeOfPreparation FROM guideofpreparation WHERE
NameOfMedicine=new.Medicine) DAY);
```

```
                UPDATE storagemedicine SET Count = Count -
new.CountOfMedicine WHERE NameOfMedicine = new.Medicine;
        INSERT INTO orderbook(idOrder, FullName) VALUES(new.idRecipe,
new.PatientFullName);
            ELSE
                SET new.DateToBack = DATE_ADD(new.RecipeDate,INTERVAL
(SELECT TimeOfPreparation FROM guideofpreparation WHERE
NameOfMedicine=new.Medicine) + 20 DAY);
                INSERT INTO orderbook(idOrder, FullName, OrderStatus)
VALUES(new.idRecipe, new.PatientFullName,'WaitForComponent');
                INSERT INTO requestformedicine(NameOfMedicine, Count)
VALUES(new.Medicine, new.CountOfMedicine);
            END IF;
    ELSE
            IF      0 > ANY(SELECT Count -
CountOfComponent*new.CountOfMedicine FROM
                storagecomponent s JOIN componentinmedicine c ON
s.idComponent=c.idComponent
        JOIN guideofpreparation g ON g.ID=c.idMedicine WHERE
g.NameOfMedicine=new.Medicine AND Count -
CountOfComponent*new.CountOfMedicine<0) THEN

                SET new.DateToBack = DATE_ADD(new.RecipeDate, INTERVAL
(SELECT TimeOfPreparation FROM guideofpreparation WHERE
NameOfMedicine=new.Medicine) + 25 DAY);
                INSERT INTO orderbook(idOrder, FullName, OrderStatus)
VALUES(new.idRecipe, new.PatientFullName,'WaitForComponent');
                INSERT INTO requestformedicine(NameOfMedicine, Count)
SELECT Component, CountOfComponent*new.CountOfMedicine
        FROM (storagecomponent s JOIN componentinmedicine c ON s.idComponent =
c.idComponent) JOIN guideofpreparation g ON g.ID=c.idMedicine
        WHERE Count - CountOfComponent*new.CountOfMedicine <0 AND
NameOfMedicine = new.Medicine;
      ELSE
        SET new.DateToBack = DATE_ADD(new.RecipeDate,INTERVAL (SELECT
TimeOfPreparation FROM guideofpreparation WHERE
NameOfMedicine=new.Medicine) DAY);
        UPDATE storagecomponent, componentinmedicine, guideofpreparation
        SET storagecomponent.Count  = storagecomponent.Count -
componentinmedicine.CountOfComponent*new.CountOfMedicine
        WHERE guideofpreparation.NameOfMedicine=new.Medicine AND
componentinmedicine.idMedicine=guideofpreparation.ID AND
storagecomponent.idComponent=componentinmedicine.idComponent;
                INSERT INTO orderbook(idOrder, FullName)
VALUES(new.idRecipe, new.PatientFullName);
      END IF;
    END IF;
   END IF;
END$$
```

```
USE `pharmacy`$$
DROP TRIGGER IF EXISTS `pharmacy`.`supply_medicine` $$
USE `pharmacy`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `pharmacy`.`supply_medicine`
AFTER DELETE ON `pharmacy`.`requestformedicine`
FOR EACH ROW
BEGIN
    UPDATE storagemedicine SET Count = Count + old.Count WHERE
NameOfMedicine = old.NameOfMedicine;
   UPDATE storagecomponent SET Count = Count + old.Count WHERE Component =
old.NameOfMedicine;
   UPDATE storagemedicine SET ShelfLife = DATE_ADD(ShelfLife, INTERVAL 5
YEAR) WHERE NameOfMedicine = old.NameOfMedicine;
   UPDATE storagecomponent SET ShelfLife = DATE_ADD(ShelfLife, INTERVAL 5
YEAR) WHERE Component = old.NameOfMedicine;

END$$


USE `pharmacy`$$
DROP TRIGGER IF EXISTS `pharmacy`.`sale_component` $$
USE `pharmacy`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `pharmacy`.`sale_component`
AFTER UPDATE ON `pharmacy`.`storagecomponent`
FOR EACH ROW
BEGIN
    IF new.Count <= new.CriticalCount THEN
            INSERT INTO requestformedicine(NameOfMedicine, Count)
VALUES(new.Component, new.CriticalCount*3);
    END IF;
END$$


USE `pharmacy`$$
DROP TRIGGER IF EXISTS `pharmacy`.`sale_medicine` $$
USE `pharmacy`$$
CREATE
DEFINER=`root`@`localhost`
TRIGGER `pharmacy`.`sale_medicine`
AFTER UPDATE ON `pharmacy`.`storagemedicine`
FOR EACH ROW
BEGIN
    IF new.Count <= new.CriticalCount THEN
            INSERT INTO requestformedicine(NameOfMedicine, Count)
VALUES(new.NameOfMedicine, new.CriticalCount*5);
```

```
    END IF;
END$$


DELIMITER ;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```