|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования  **"МИРЭА — Российский технологический университет"**  **РТУ МИРЭА**   |  | | --- | | Институт информационных технологий (ИИТ) | | Кафедра промышленной информатики (ПИ) | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ** | | | | |
|  | | | | |
| по дисциплине «Разработка баз данных» | | | | |
| **Студент группы** | ИМБО-02-22 |  |  |
|  | *(учебная группа, фамилия, имя, отчество студента)* |  | *(подпись студента)* |
| **Преподаватель** | Ст. Преподаватель Баев И.Б. |  |  |
| *(должность, ученая степень, звание, фамилия, имя, отчество преподавателя) (подпись преподавателя)* | | | |

|  |  |
| --- | --- |
| Работа выполнена | «\_\_» \_\_\_\_ 2024 г. |
| «Зачтено» | «\_\_» \_\_\_\_ 2024 г. |
|  |  |

Москва 2024

Содержание

[Описание предметной области 3](#_Toc180706633)

[физическая модель базы данных 4](#_Toc180706634)

[Результата выполнения практических работ в SQL Command Line 5](#_Toc180706635)

[Результат выполнения практической работы по использованию СУБД 30](#_Toc180706636)

[Перенос базы данных 30](#_Toc180706637)

[Команды модификации данных (DML) 40](#_Toc180706639)

[Выборка данных (DQL) 43](#_Toc180706640)

[Процедуры 48](#_Toc180706641)

[Функции 52](#_Toc180706642)

[Триггеры 55](#_Toc180706643)

[Оконные функции 58](#_Toc180706644)

[Заключение 62](#_Toc180706645)

[Список использованных источников 63](#_Toc180706646)

Описание предметной области

В данной работе была рассмотрена

физическая модель базы д анных

На Рисунке 1 представлена физическая модель рассматриваемой предметной области.

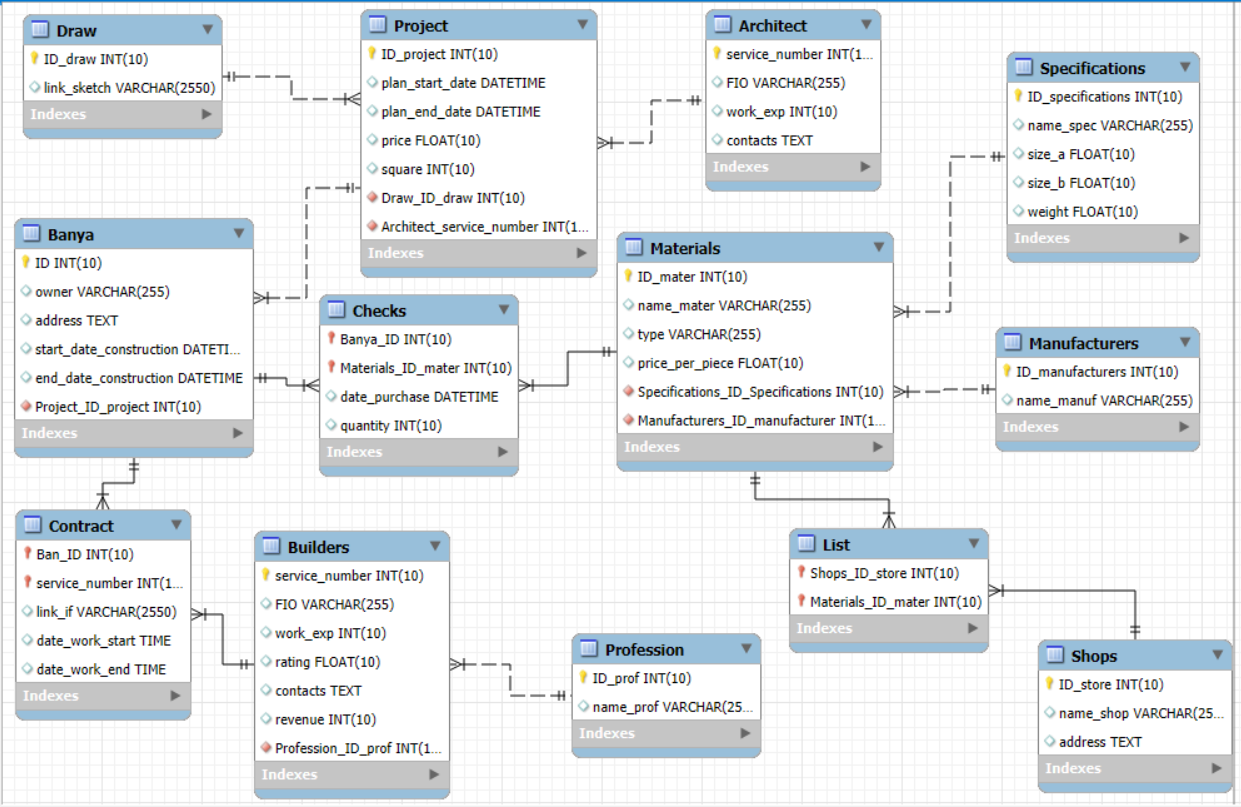


Рисунок 1 — Физическая модель предметной области

Результата выполнения практических работ в SQL Command Line

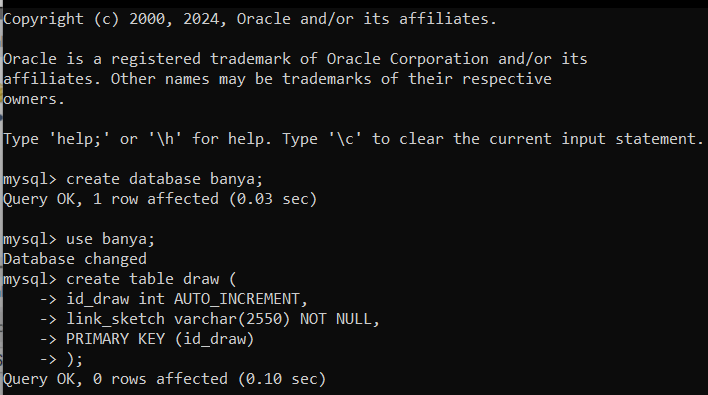
Для начала создадим БД banya. Результат представлен на Рисунке 2.

Рисунок 2 — Создание базы данных

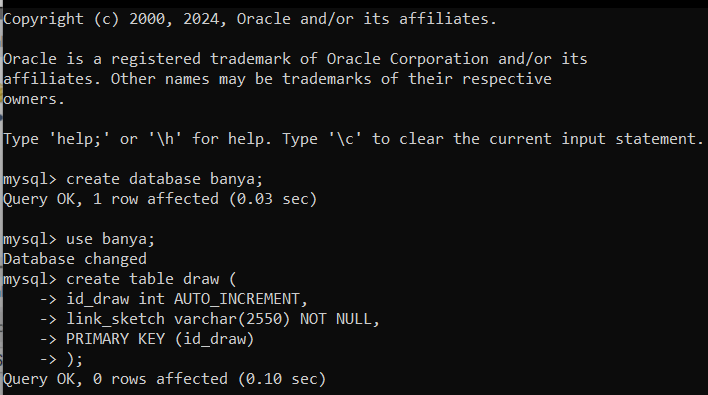
Для нашей работы выберем нашу БД banya. Результат представлен на Рисунке 3.

Рисунок 3 — Использование базы данных

Теперь необходимо создать и заполнить все необходимые таблицы. Каждый из столбцов имеет свой тип данных и ограничение. Начнем с центральной для данной задачи таблицы draw, которая содержит информацию о чертежах. Она включает в себя столбцы для уникального идентификатора каждого чертежа (id\_draw), ссылка на чертежи (link\_sketch).

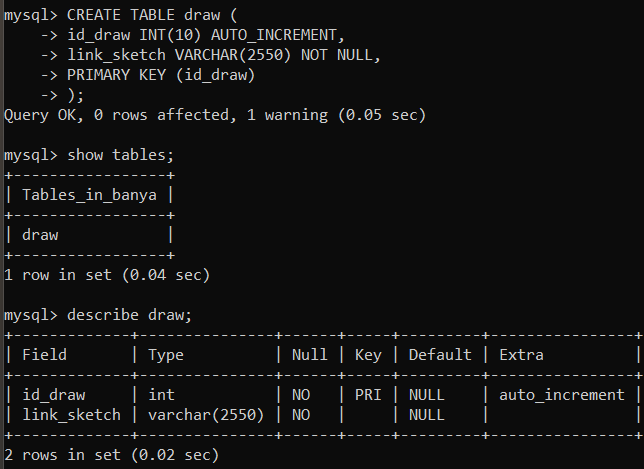


Рисунок 4 — Создание таблицы draw

В таблице architect содержит информацию об нанятых архитекторах, которые делают проекты. Она содержит столбцы служебный номер (service\_number), ФИО (fio), опыт работы в годах (work\_exp), контакты (contacts)

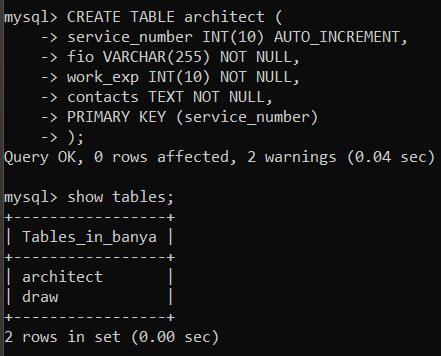


Рисунок 5 — Создание таблицы architect

Таблица profession хранятся название профессии. Она содержит столбцы для идентификатора професии (id\_prof), название профессии (name\_prof).

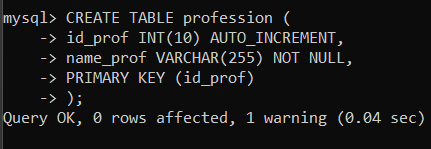


Рисунок 6 — Создание таблицы profession

Таблица shops предназначена для местонахождении магазин стройматериалов. Она включает в себя столбцы для уникального идентификатора магазина (id\_store), название магазина (name\_shop), и его адреса (address).

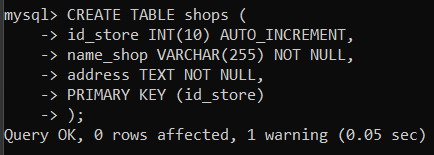


Рисунок 7 — Создание таблицы shops

Таблица manufacturers содержит название производителей купленных или некупленных материалов. Она включает столбцы для идентификаторов производителей (id\_manufacturers), названия производителя (name\_manuf).

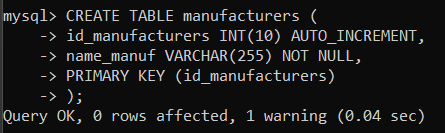


Рисунок 8 — Создание таблицы manufacturers

Таблица specifications хранит данные о характеристиках материалов. Она включает столбцы для идентификатора материала (id\_specifications), названия (name\_spec), size\_a\_metr (длина), size\_b\_metr (ширина), weight\_kilo (вес). Эта таблица обеспечивает подробную информацию о материалах.

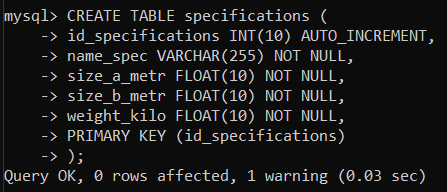


Рисунок 9 — Создание таблицы specifications

Таблица project содержит информацию об планах постройки. Она включает столбцы для идентификаторов проекта (id\_project), начало планирования постройки (plan\_start\_date), конец постройки по плану (plan\_end\_date), площадь (square), цена проекта (price), уникальные чертежи (draw\_id\_draw), сервисный номер архитектора (architect\_service\_number).

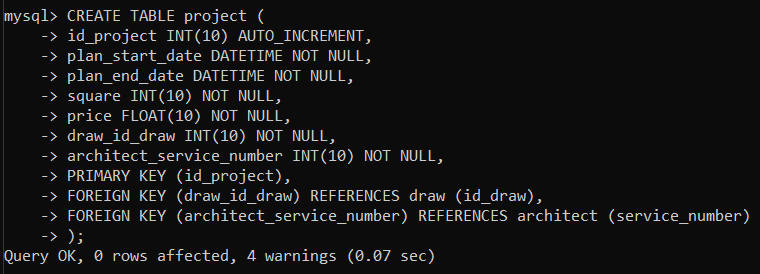


Рисунок 10 — Создание таблицы project

Таблица banya сведения о владельце бани. Она включает столбцы для уникального идентификатора владельца (id), имя владельца (owner), адрес, где находится баня (address), дата начало постройки (start\_date\_constuction), дата окончания постройки, идентификанция проекта (project\_id).

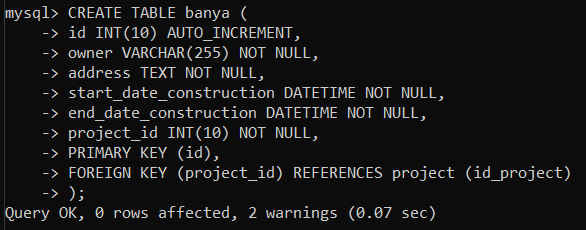


Рисунок 11 — Создание таблицы banya

Таблица materials содержит информацию о материалах,. Она включает столбцы для уникального идентификатора материала (id\_mater), названия материала (name\_mater), тип (type), цена за одну штуку (price\_per\_piece), идентификатор характеристики (specifications\_id), идентификатор производителя (manufacturers\_id).

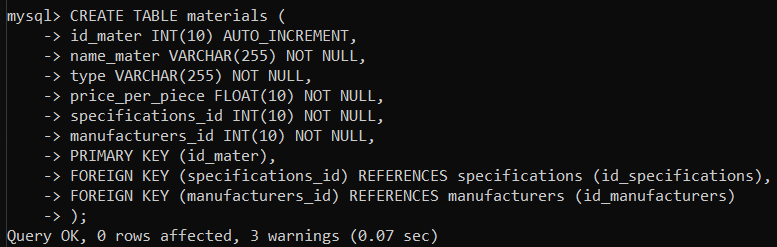


Рисунок 12 — Создание таблицы materials

Таблица list связывает материалы и магазины. Она включает столбцы для идентификаторов магазина и материала.

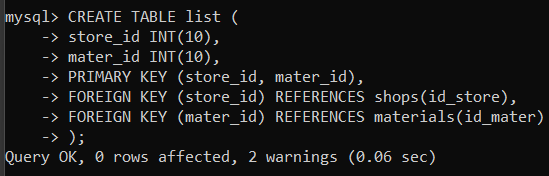


Рисунок 13 — Создание таблицы list

Таблица checks содержит данные о купленных материалах. Она включает столбцы для идентификаторов бани и материала, дата покупки (data\_purchase), в каком количестве купили (quantity).

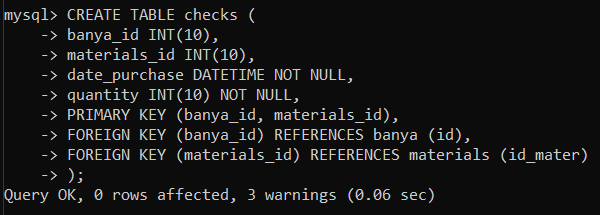


Рисунок 14 — Создание таблицы checks

Таблица builders хранит данные о всех строителях. Она включает столбцы для уникального номера строителя (service\_number), ФИО (fio), опыт работы (work\_exp), контакты (contacts), зарплата (revenue), идентификатор професии (profession\_id).

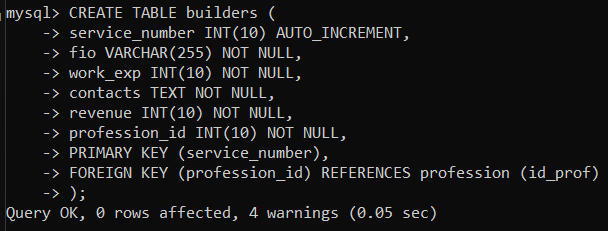


Рисунок 15 — Создание таблицы builders

Таблица contract содержит все договоры. Она включает столбцы для уникального идентификатора бани (ban\_id), идентификатора строителя (service\_number), ссылка на условия работы (link\_if), начало рабочей смены (date\_work\_start), конец рабочей смены (date\_work\_end).

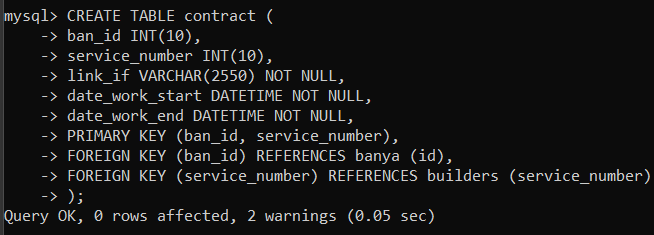


Рисунок 16 — Создание таблицы contract

Произведем просмотр созданных таблиц в базе данных.

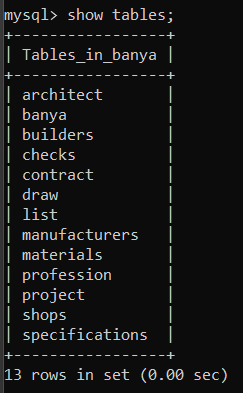


Рисунок 17 — Просмотр таблиц базы данных banya

Внесем данные в таблицы shops и architect. Для этого введем представленные ниже команду.

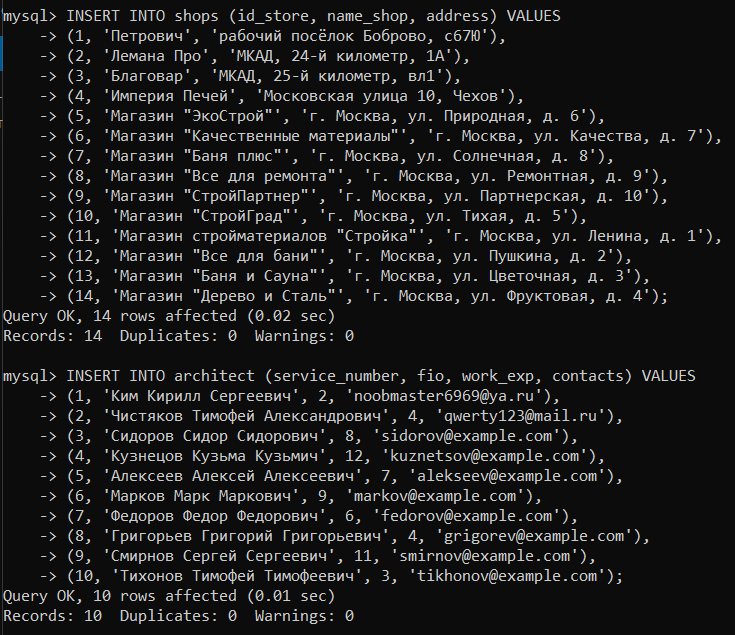


Рисунок 18 — Заполнение таблиц shops и architect

Внесем данные в таблицы manufacturers и profession. Для этого введем представленные ниже команду.

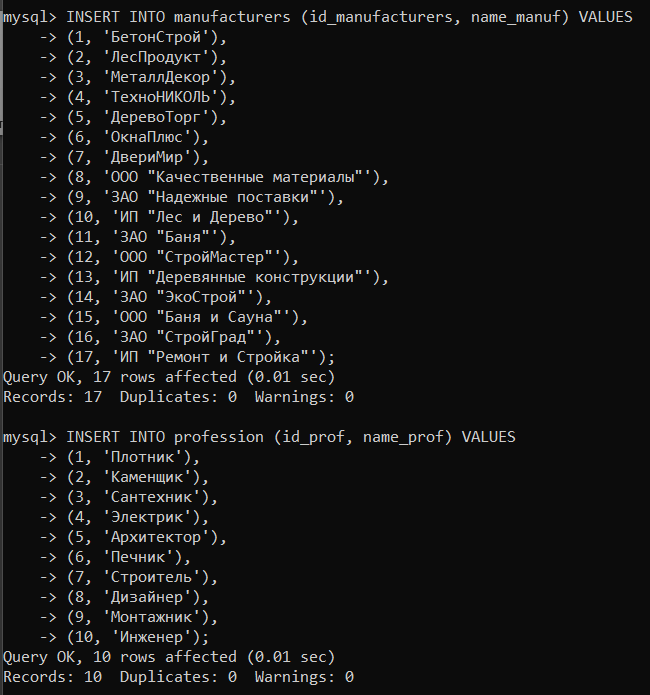


Рисунок 19 — Заполнение таблиц manufacrurers и profession

Внесем данные в таблицы draw и specifications. Для этого введем представленные ниже команду.

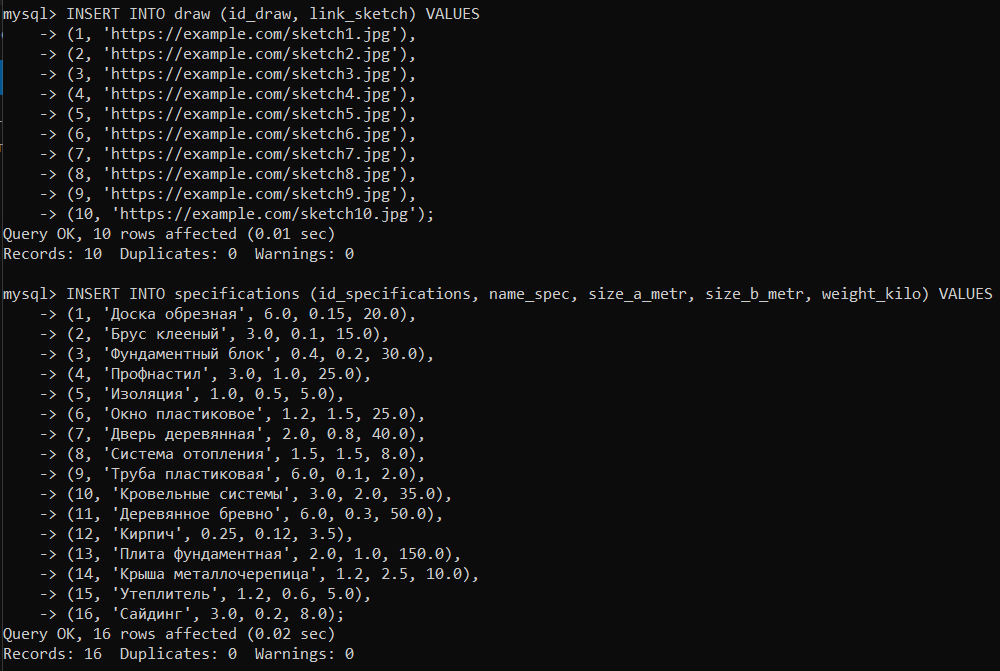


Рисунок 20 — Заполнение таблиц draw и specifications

Внесем данные в таблицы project и banya. Для этого введем представленные ниже команду.

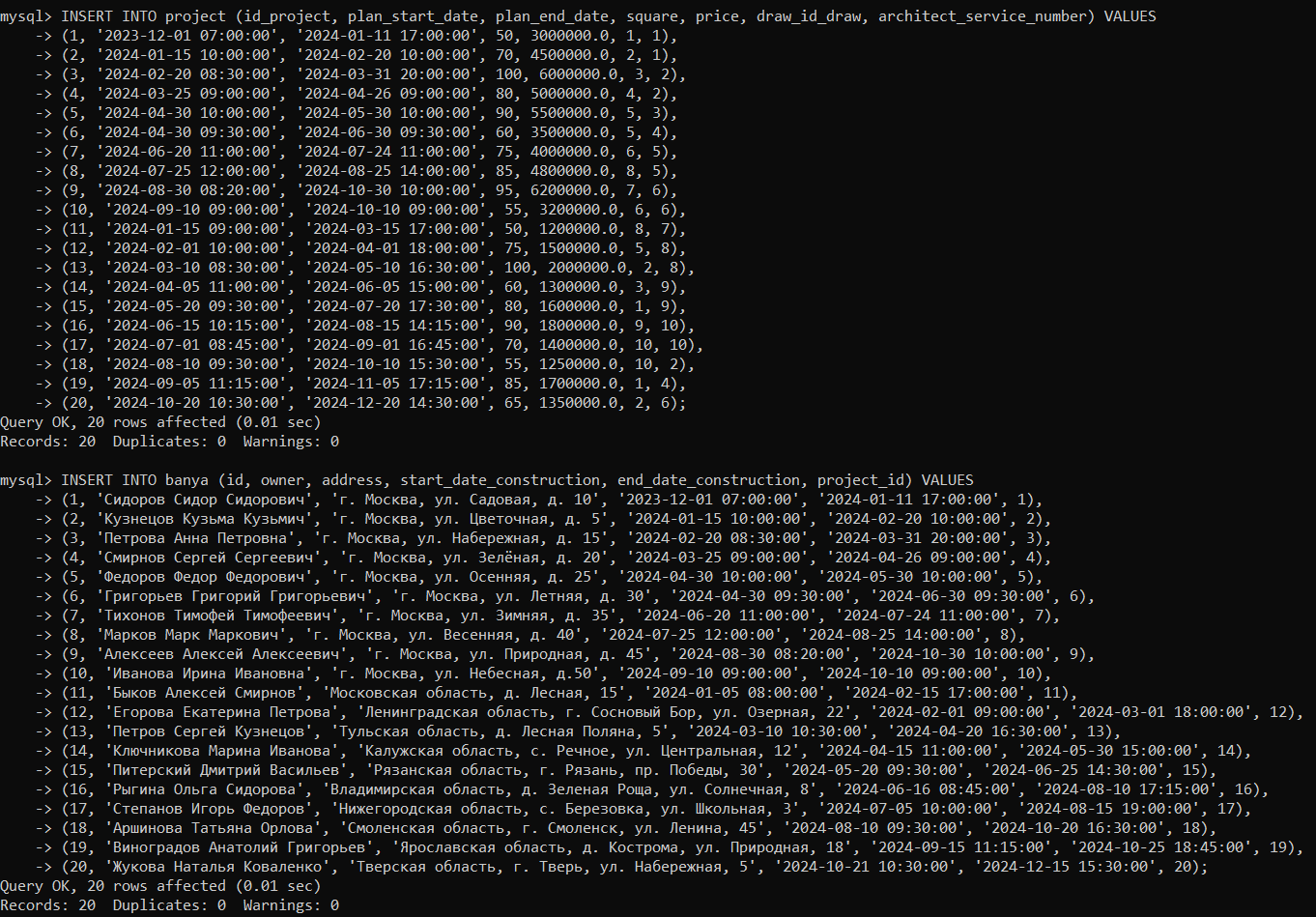


Рисунок 21 — Заполнение таблиц project и banya

Внесем данные в таблицу materials. Для этого введем представленные ниже команду.

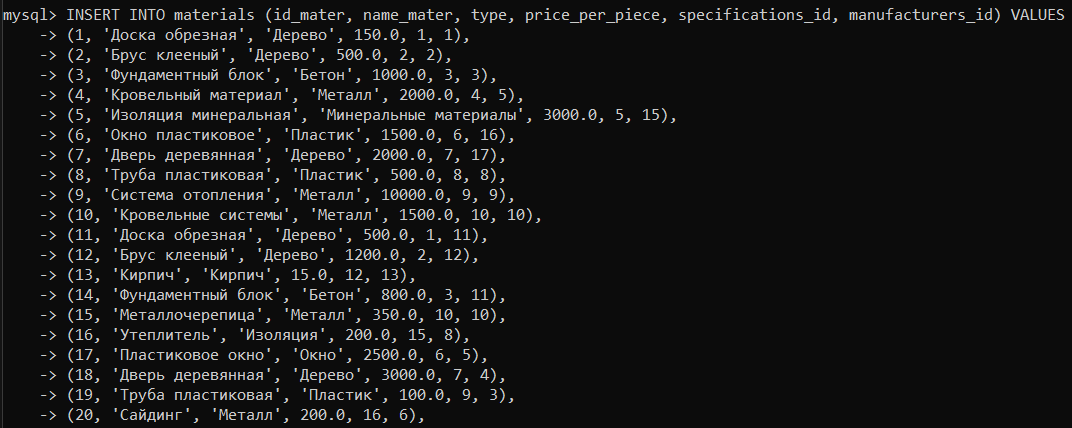


Рисунок 22 — Заполнение таблиц materials

Внесем данные в таблицы list. Для этого введем представленные ниже команду.

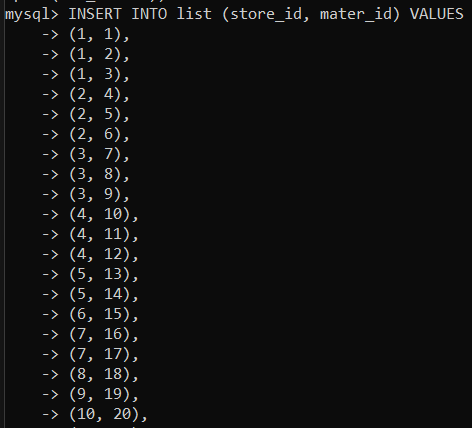


Рисунок 23 — Заполнение таблиц list

Внесем данные в таблицу checks. Для этого введем представленные ниже команду.

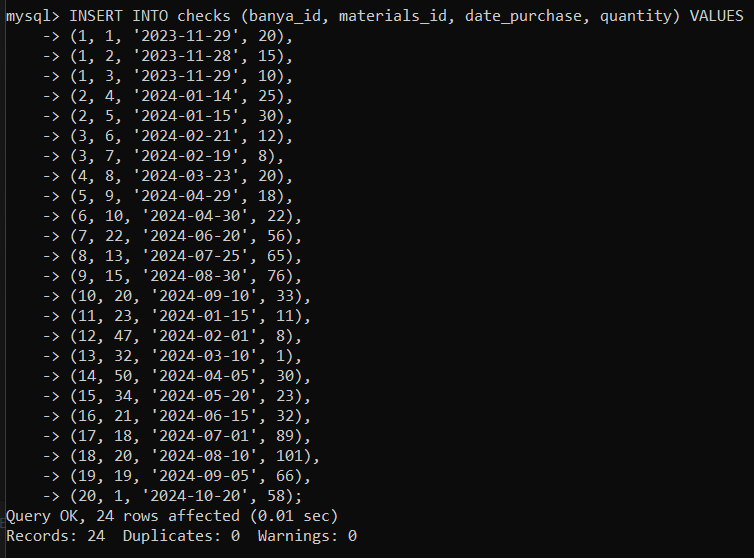


Рисунок 24 — Заполнение таблицы checks

Внесем данные в таблицы builders и contract. Для этого введем представленные ниже команду.

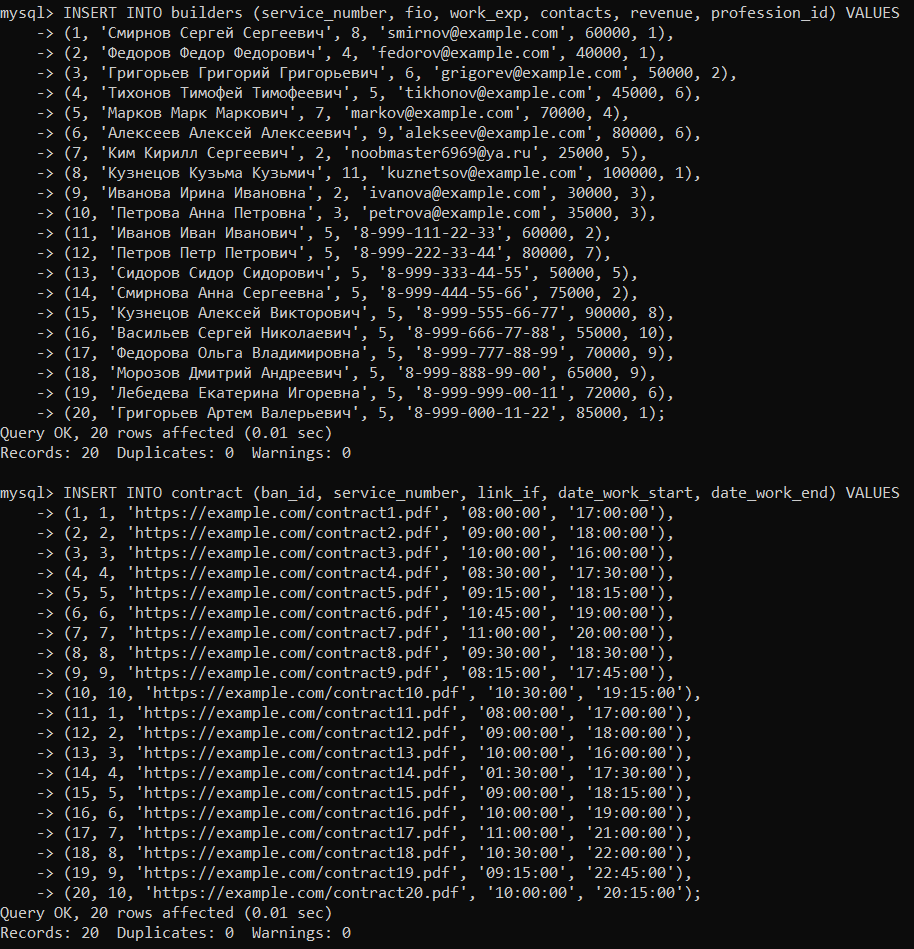


Рисунок 25 — Заполнение таблицы builders и contract

Просмотрим все таблицы после заполнения при помощи команды SELECT \* FROM «название таблицы»;

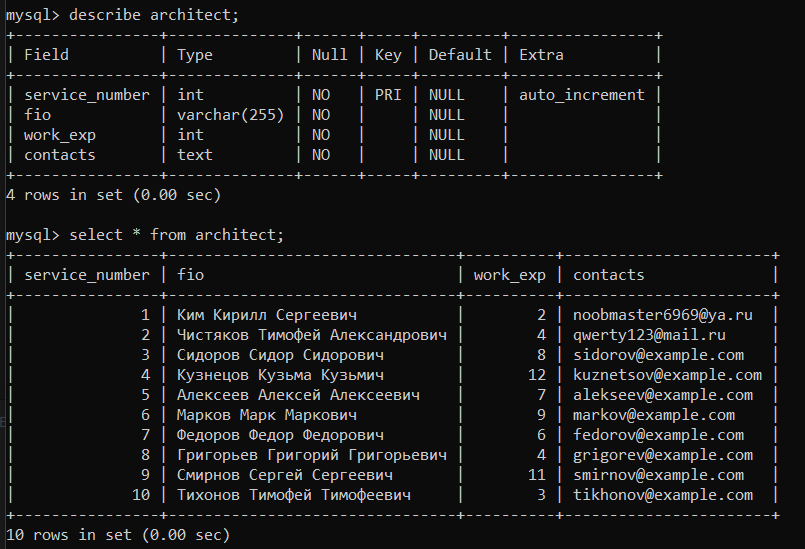


Рисунок 26 — Содержание таблицы architect

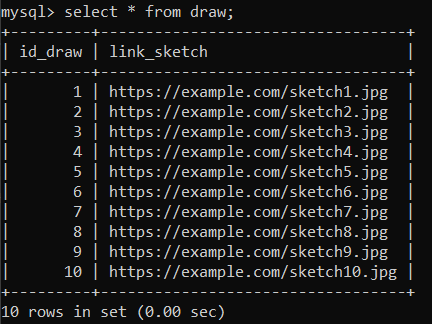


Рисунок 27 — Содержание таблицы draw

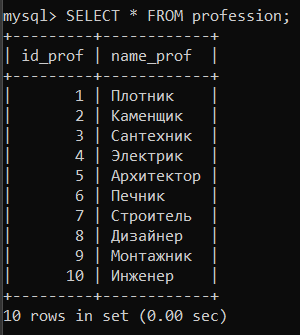


Рисунок 28 — Содержание таблицы profession

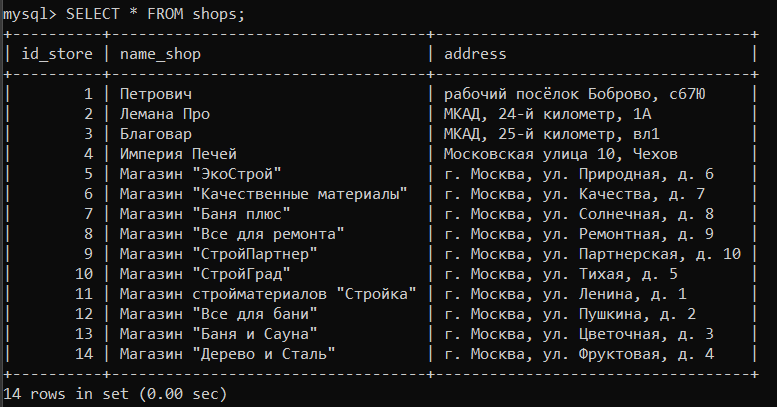


Рисунок 29 — Содержание таблицы shops

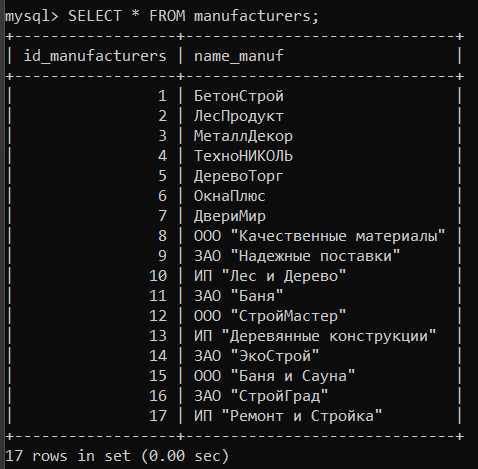


Рисунок 30 — Содержание таблицы manufacturers

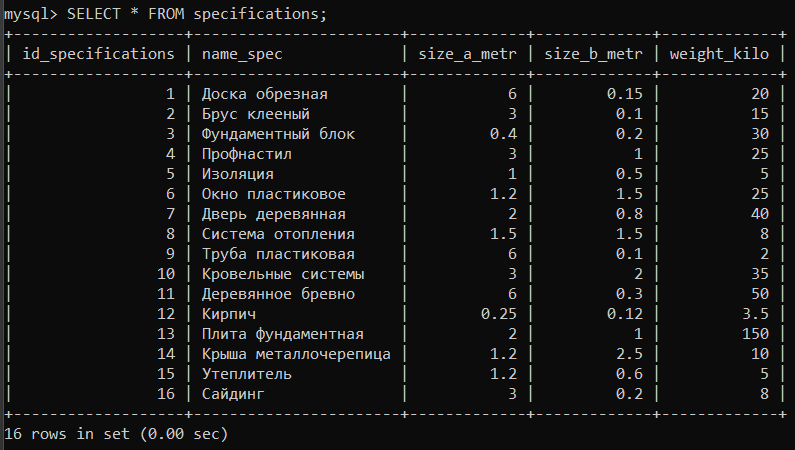


Рисунок 31 — Содержание таблицы specifications

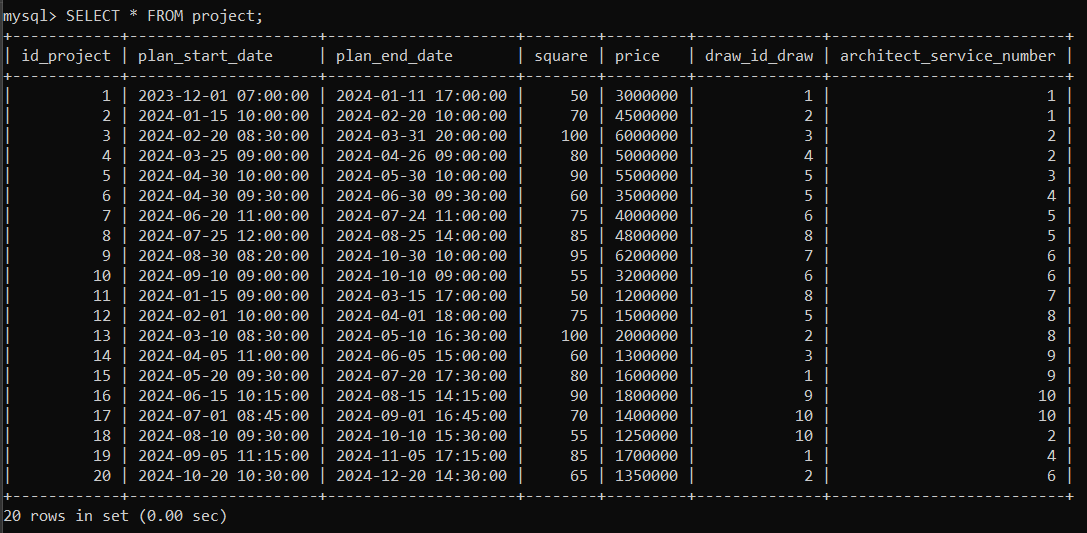


Рисунок 32 — Содержание таблицы project

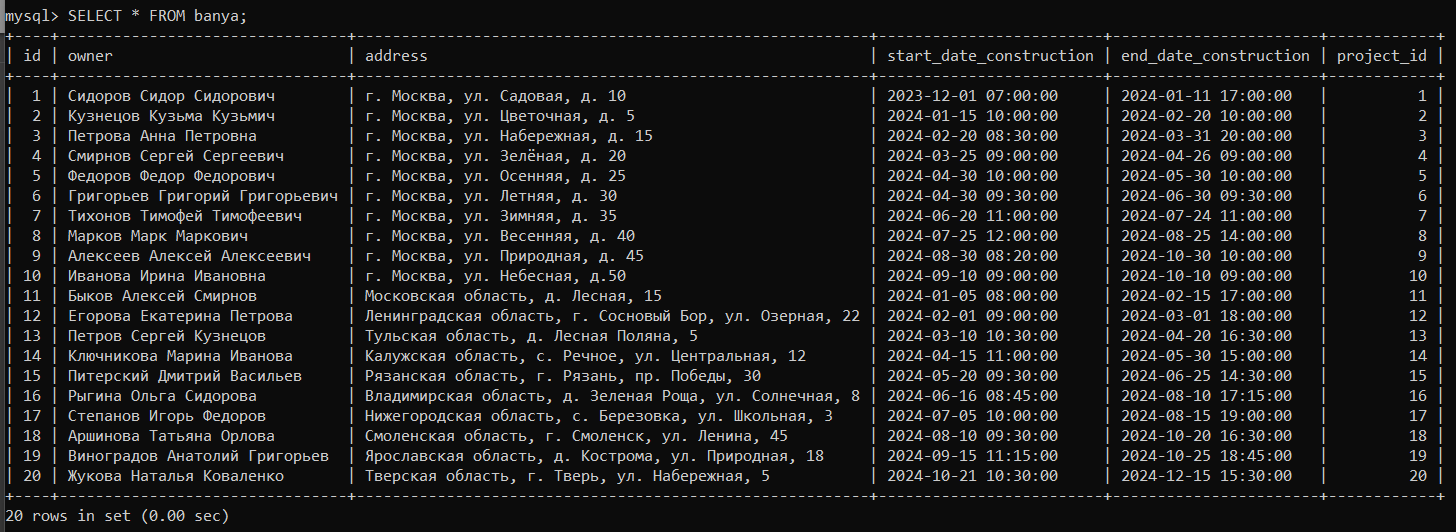


Рисунок 33 — Содержание таблицы banya

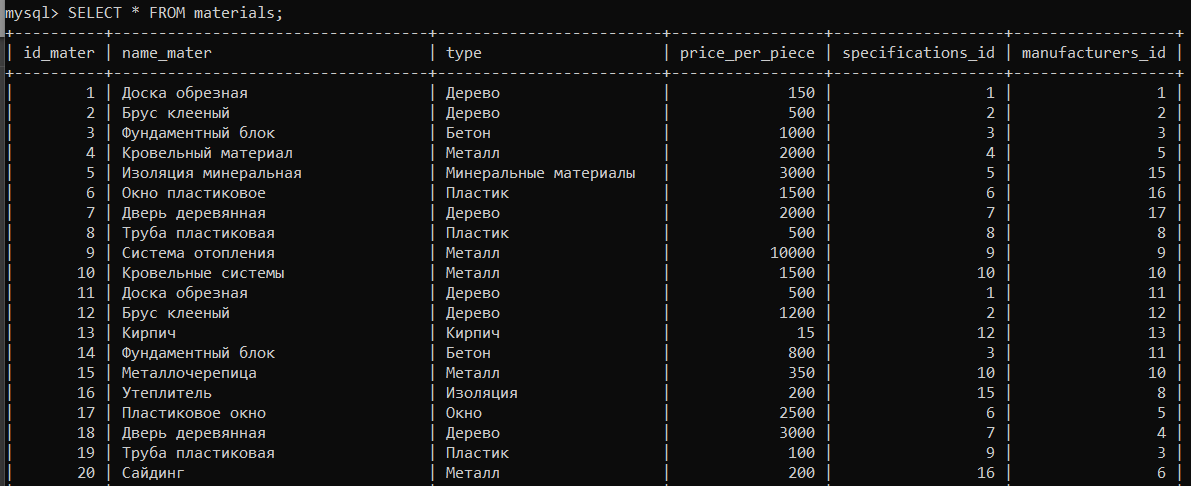


Рисунок 34 — Содержание таблицы materials

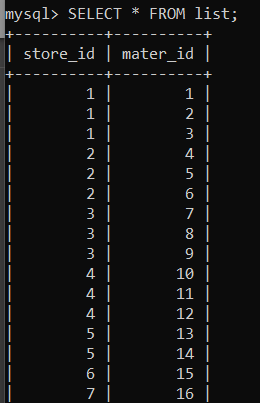


Рисунок 35 — Содержание таблицы list

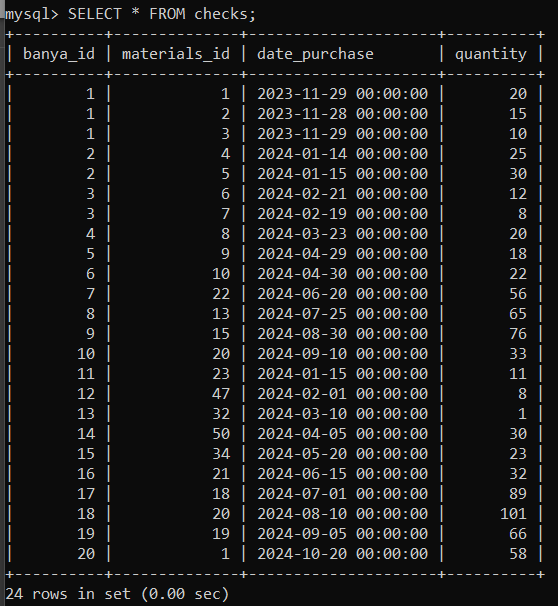


Рисунок 36 — Содержание таблицы checks

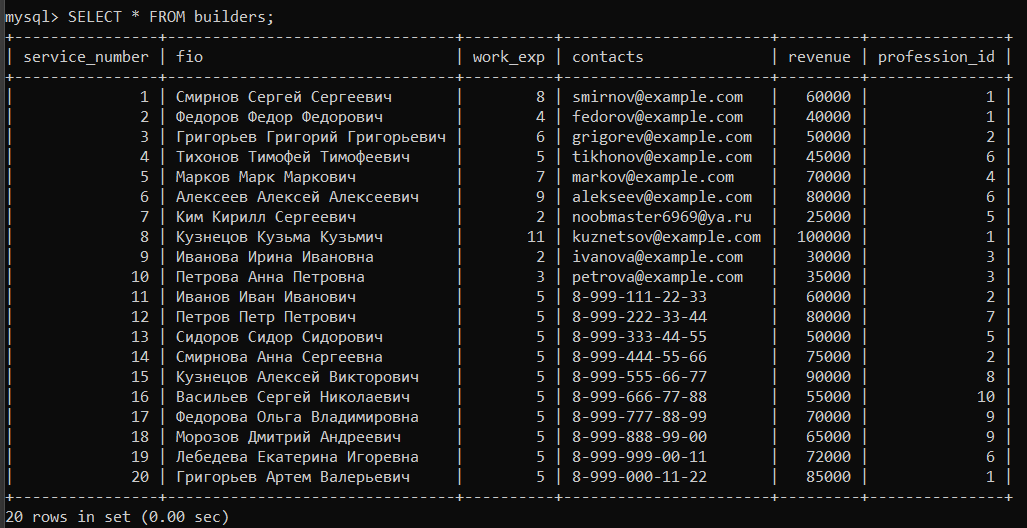


Рисунок 36 — Содержание таблицы builders

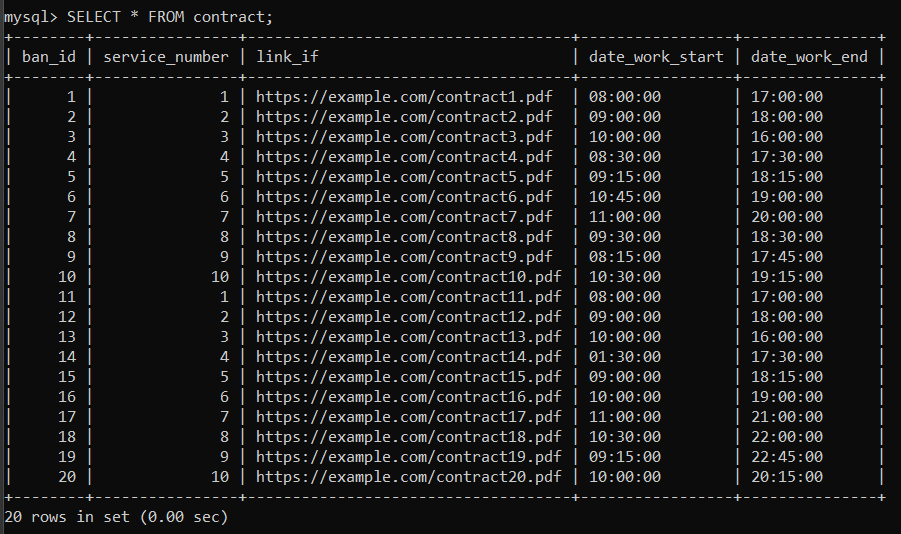


Рисунок 37 — Содержание таблицы contract

Просмотр столбца из таблицы при помощи команды SELECT «имя» FROM «имя»;

SELECT fio FROM architect;



Рисунок 38 — Вывод столбца fio

Просмотр несколько столбцов из таблицы при помощи команды SELECT «имя», «имя» FROM «имя»;

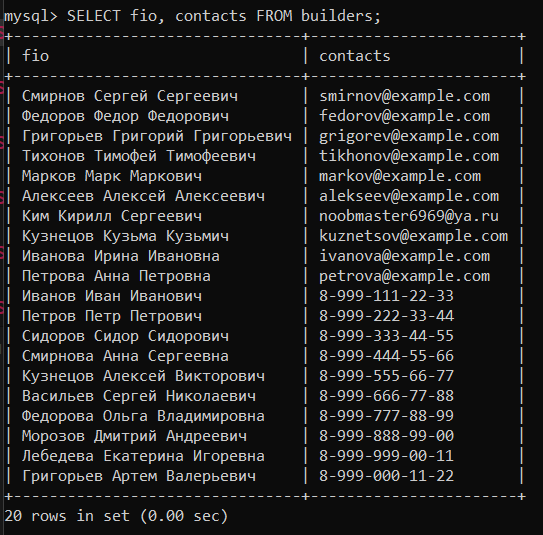


Рисунок 39 — Вывод столбцов fio и contacts строителей

Сортировка столбца по фамилиям по возрастанию при помощи команды SELECT \* FROM имя\_таблицы ORDER BY имя\_столбца\_сортировки;



Рисунок 40 — Сортировка материалов в порядке убывания

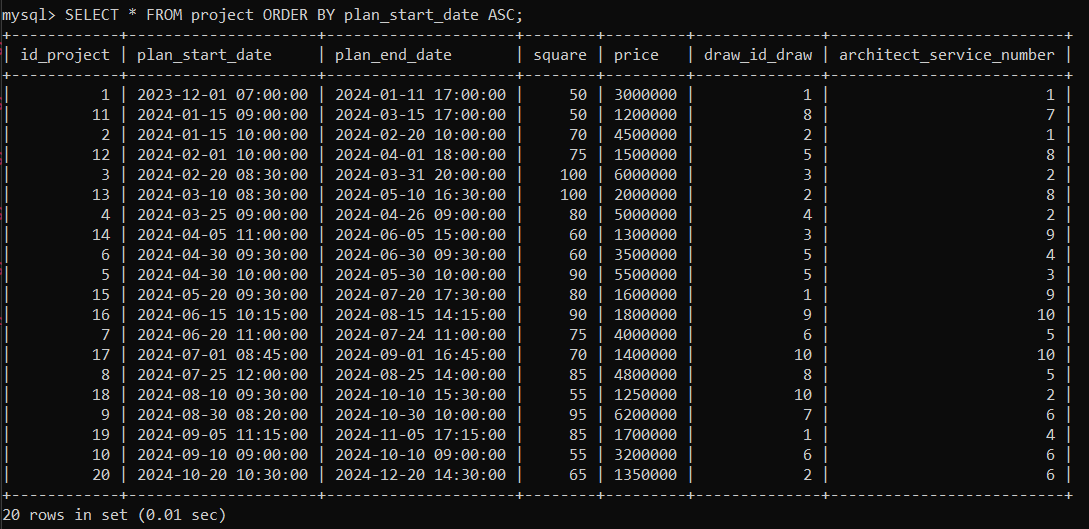


Рисунок 41 — Сортировка проектов по дате начало планирования в порядке возрастания

Очень часто бывает, что все информация из таблицы не нужна. Например, необходимо узнать, архитектор, который имеет опыт работы «4 года». Для этого в SQL есть ключевое слово WHERE. Отсортируем значения в столбцах по заданному условию.

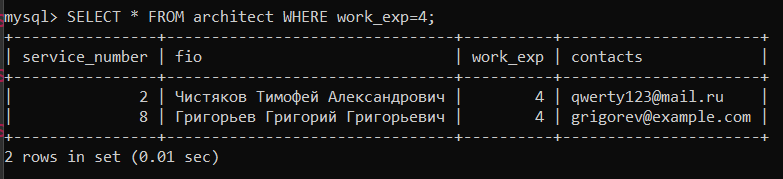


Рисунок 42 — Условие для таблицы architect

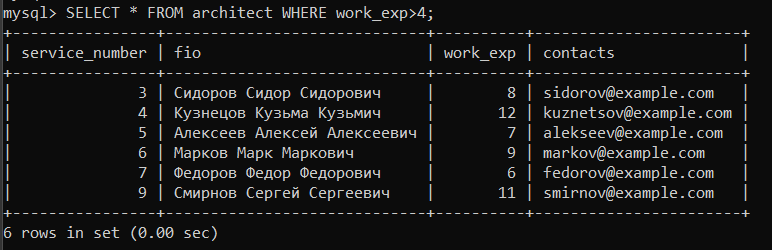


Рисунок 43 — Условие для таблицы architect

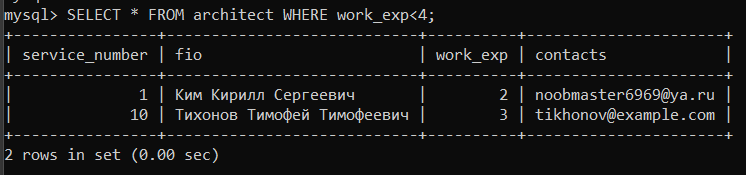


Рисунок 44 — Условие для таблицы architect

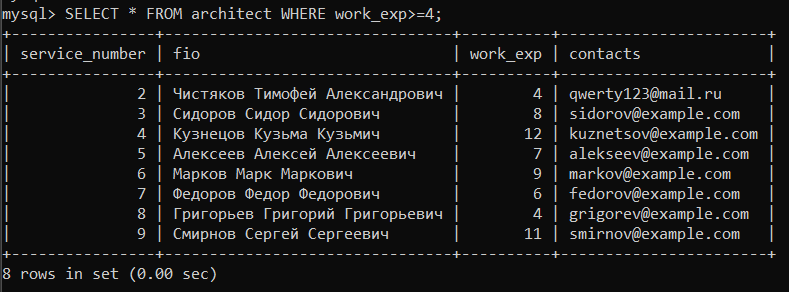


Рисунок 45 — Условие для таблицы architect

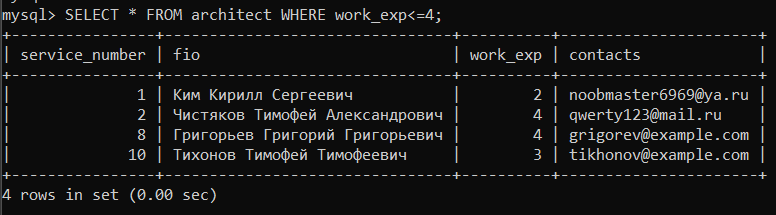


Рисунок 46 — Условие для таблицы architect

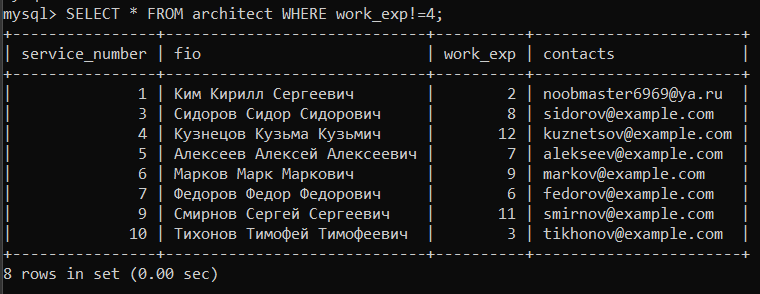


Рисунок 47 — Условие для таблицы architect

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 48 — Условие для таблицы architect



Рисунок 49 — Отбираются значения не имеющие значения в указанном поле

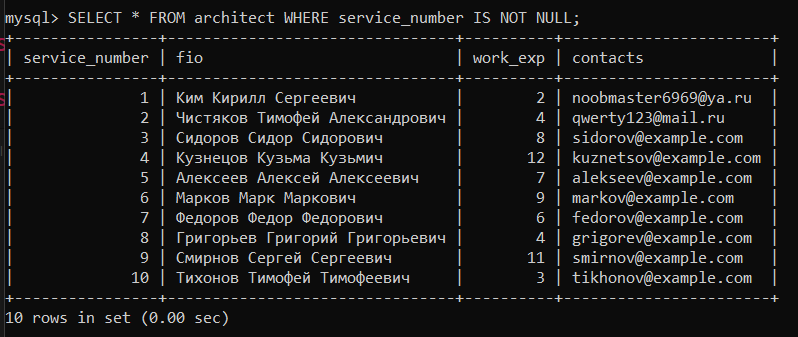


Рисунок 50 — Отбираются значения имеющие значения в указанном поле

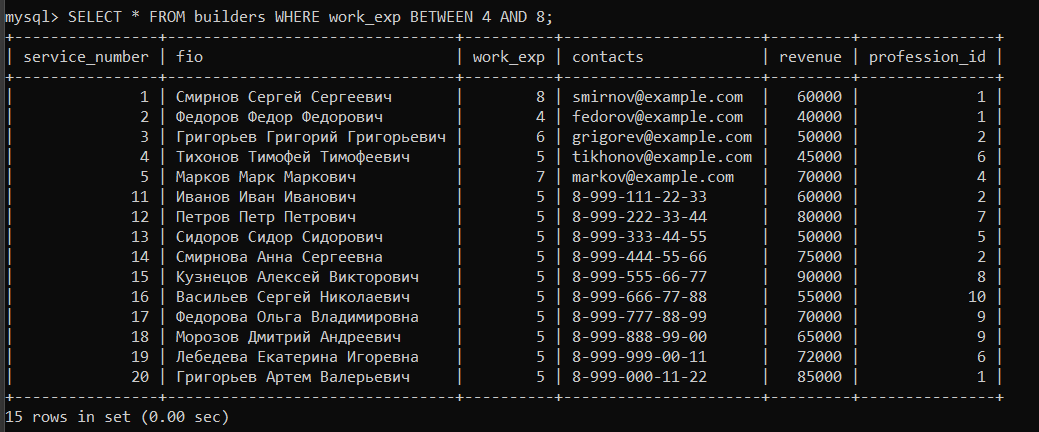


Рисунок 51 — Выборка строителей с опытом работы в определенном диапазоне

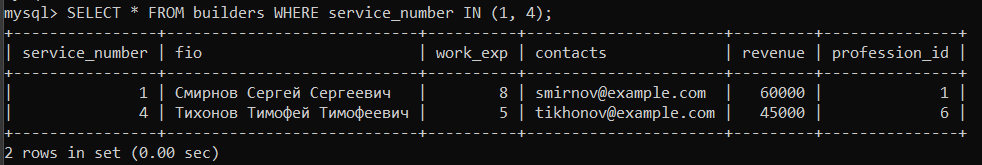


Рисунок 52 — Отбираются значения соответствующие указанным

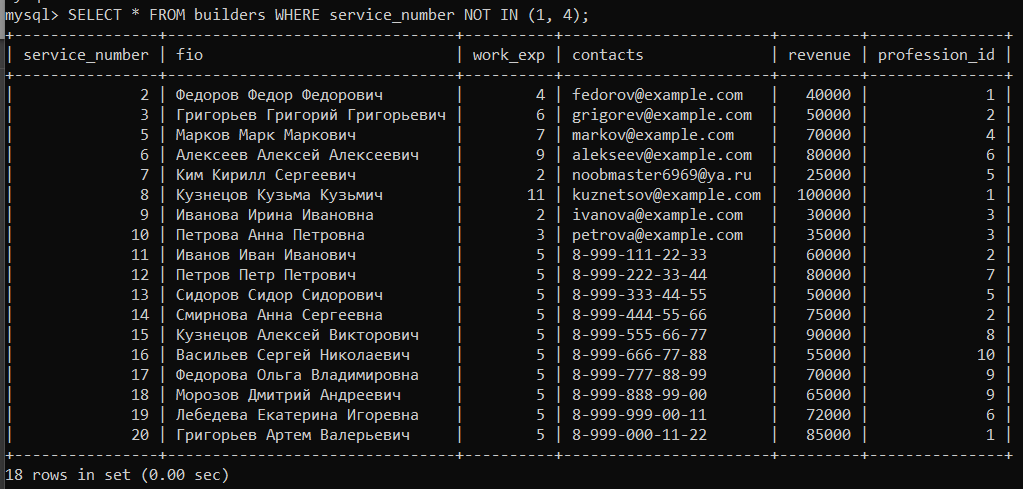


Рисунок 53 — Отбираются значения не соответствующие указанным

Выведем из таблицы shops строчки, где в списке продуктов встречается «МКАД».

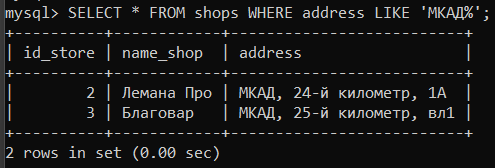


Рисунок 54 — Отбираются значения, соответствующие образцу с использованием LIKE

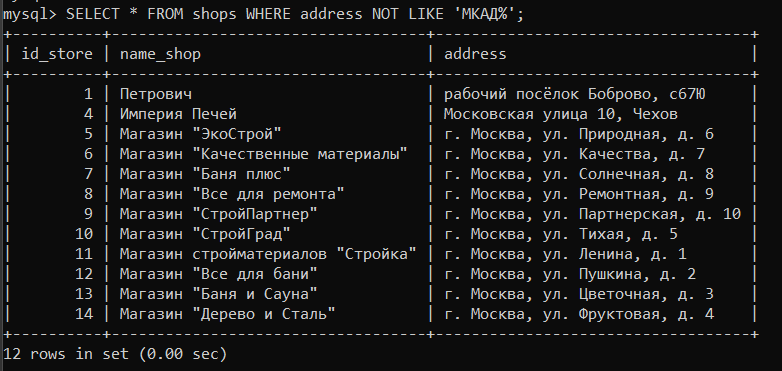


Рисунок 55 — Отбираются не соответствующие значения с использованием NOT LIKE

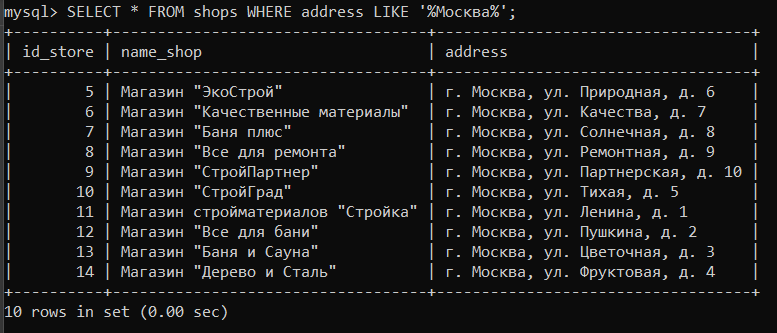


Рисунок 56 — Метасимвол

**Изменение данных в таблице.**

Забыли добавить в таблице builders – rep (рейтинг).

Для добавления столбцов в таблицу используется оператор ALTER TABLE — ADD COLUMN.

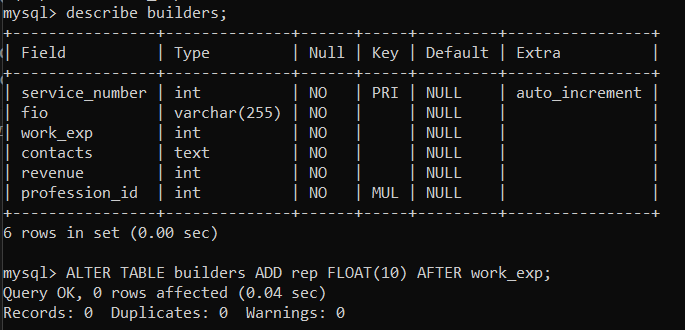


Рисунок 57 — Добавление нового столбца в таблицу

Запишем значения в новый столбец.

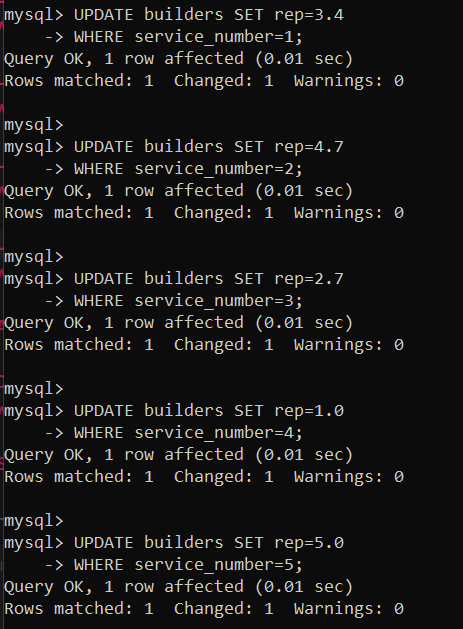


Рисунок 58 — Добавления значений

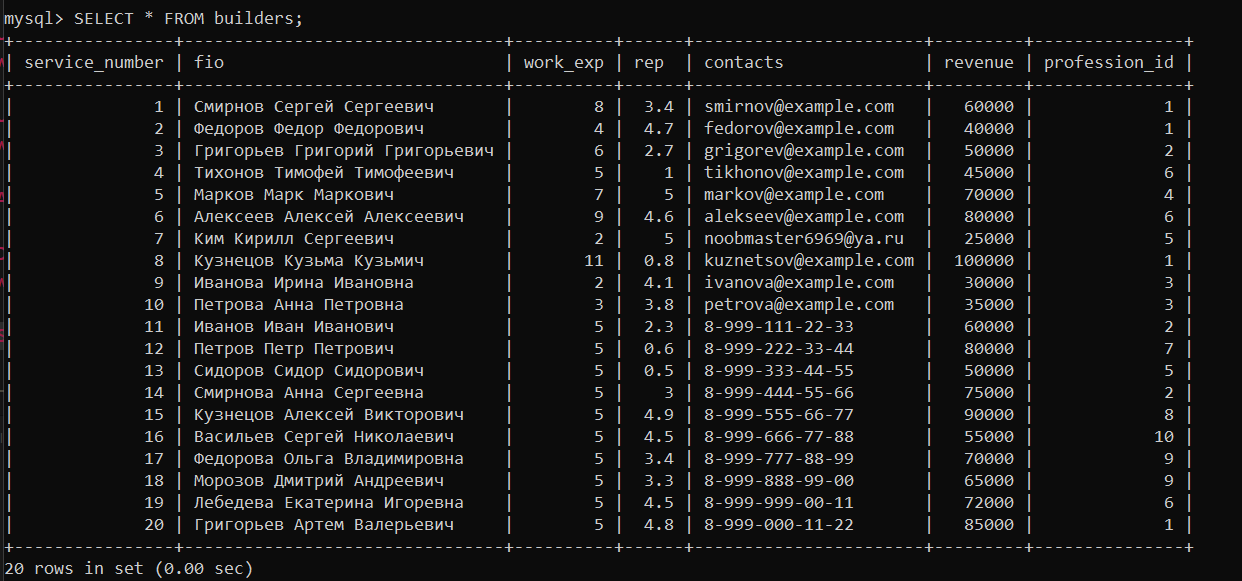


Рисунок 55 — Обновление информации

Переименовать столбец с изменением типа данных командой ALTER TABLE builders CHANGE COLUMN rep rating FLOAT(10);

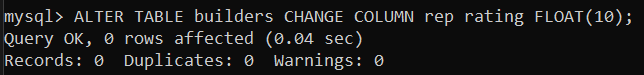


Рисунок 56 — Замена названия столбца

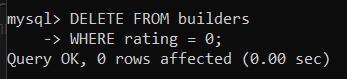


Рисунок 57 — Удаление записей с 0 рейтингом

Операция соединения. Вывести металл и название производителя:

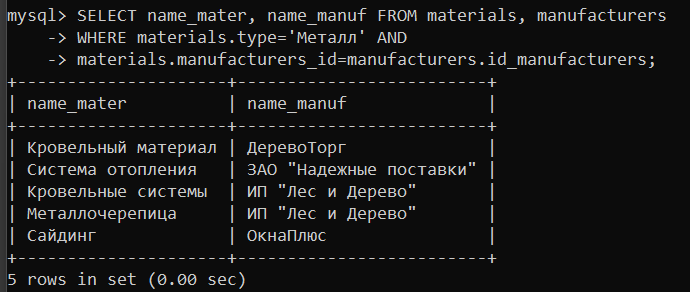


Рисунок 58 — Вывод информации по критериям

Операция объединения. Вывести название материалов и цену:

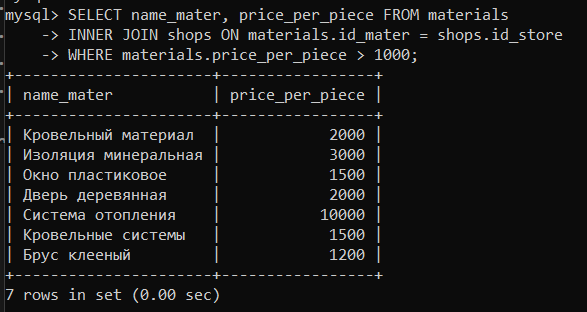


Рисунок 59 — Вывод информации по критериям

Операция пересечения. Вывести название магазина, адрес, что купить какой материл:

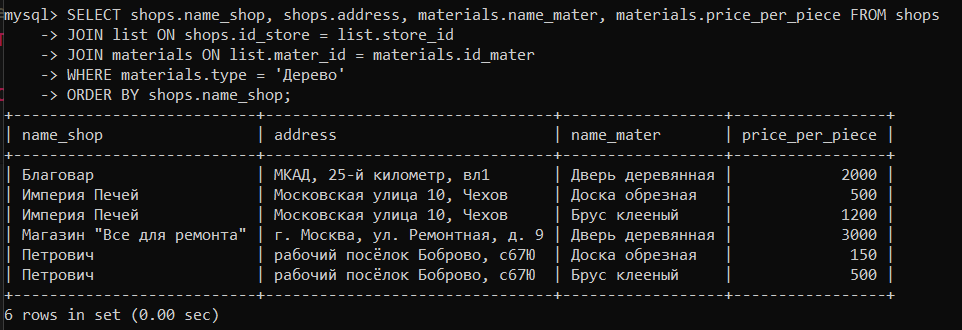


Рисунок 60 — Вывод информации по критериям

Операция разности. Вывести ФИО строителя и опыт работы:

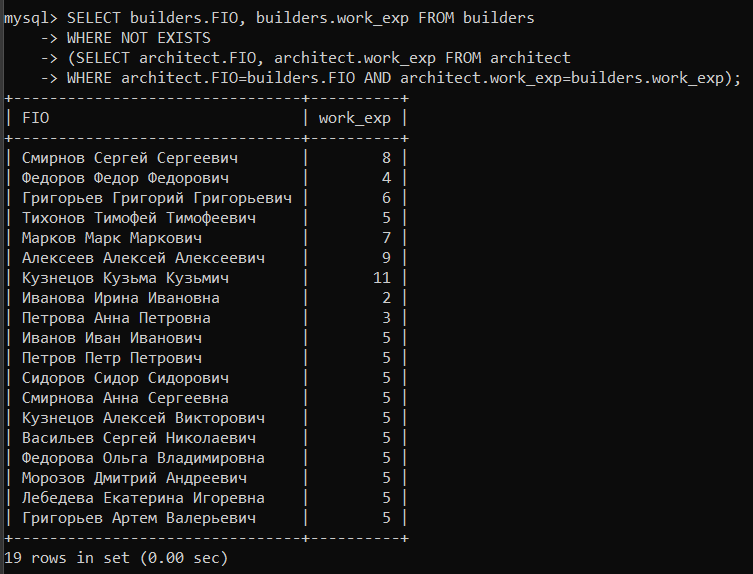


Рисунок 61 — Вывод информации по критериям

Операция группировки (GROUP BY).

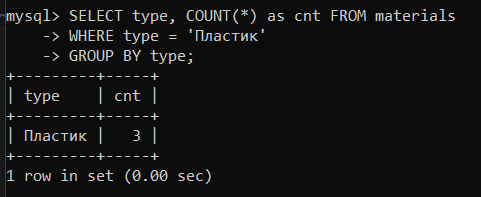


Рисунок 62 — Вывод количество пластиков

Операция сортировки. Запрос для получения списка материалов, отсортированных по цене за единицу.

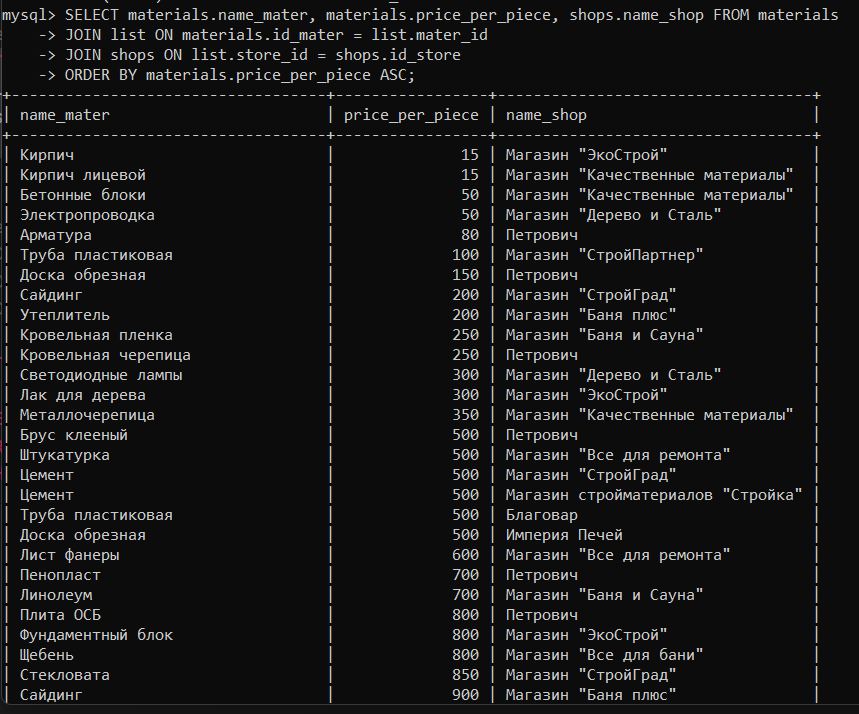


Рисунок 63 — Вывод материалов

Операция деления. Вывести название медикамента и подробный статус исследования.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 62 — Вывести название медикамента и статус исследования

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание Рисунок 63 — Название медикамента, имя исследователя и статус исследования

Вывести название медикамента и имя участника испытания, если он есть через LEFT JOIN.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 64 — Вывести название медикамента и имя участника испытания

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 65 — Вывести название медикамента и все фазы исследований

Вывести все столбцы из таблиц ingredients и medication\_ingredients посредством RIGHT JOIN.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 66 — Вывод посредством RIGHT JOIN без повторения

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 67 — Название ингредиента и количество медикаментов, в которых он используется

Результат выполнения практической работы по использованию СУБД

Перенос базы данных

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 68 — Стартовая страница миграции

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 69 — Выбор необходимых таблиц

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 70 — Предпросмотр сгенерированного скрипта

Листинг кода для переноса моей базы данных:

*Листинг 1 — Создание базы данных*

|  |
| --- |
| -- MySQL dump 10.13 Distrib 8.0.40, for Win64 (x86\_64)  --  -- Host: localhost Database: drugs  -- ------------------------------------------------------  -- Server version 8.0.40  /\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;  /\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;  /\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;  /\*!50503 SET NAMES utf8 \*/;  /\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;  /\*!40103 SET TIME\_ZONE='+00:00' \*/;  /\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;  /\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;  /\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;  /\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;  --  -- Table structure for table `clinical\_trials`  --  CREATE SCHEMA IF NOT EXISTS drugs;  USE drugs;  DROP TABLE IF EXISTS `clinical\_trials`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `clinical\_trials` (  `id\_medication` int NOT NULL,  `id\_trial` int NOT NULL AUTO\_INCREMENT,  `name` varchar(255) NOT NULL,  `description` text,  `start\_date` date NOT NULL,  `end\_date` date DEFAULT NULL,  `status` enum('in\_process','completed','canceled') NOT NULL,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_trial`),  KEY `id\_medication` (`id\_medication`),  CONSTRAINT `clinical\_trials\_ibfk\_1` FOREIGN KEY (`id\_medication`) REFERENCES `medications` (`id\_medication`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `clinical\_trials`  --  --  -- Table structure for table `ingredients`  --  DROP TABLE IF EXISTS `ingredients`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `ingredients` (  `id\_ingredient` int NOT NULL AUTO\_INCREMENT,  `ingredient\_name` varchar(255) NOT NULL,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_ingredient`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `ingredients`  --  --  -- Table structure for table `medication\_dosages`  --  DROP TABLE IF EXISTS `medication\_dosages`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `medication\_dosages` (  `clinical\_trials\_id\_medication` int NOT NULL,  `clinical\_trials\_id\_trial` int NOT NULL,  `id\_dosage` int NOT NULL AUTO\_INCREMENT,  `dosage\_type` enum('tablet','capsule','liquid','injection') NOT NULL,  `unit\_of\_measurement` varchar(255) NOT NULL,  `dosage\_amount` decimal(10,2) NOT NULL,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_dosage`),  KEY `clinical\_trials\_id\_medication` (`clinical\_trials\_id\_medication`,`clinical\_trials\_id\_trial`),  CONSTRAINT `medication\_dosages\_ibfk\_1` FOREIGN KEY (`clinical\_trials\_id\_medication`, `clinical\_trials\_id\_trial`) REFERENCES `clinical\_trials` (`id\_medication`, `id\_trial`)  ) ENGINE=InnoDB AUTO\_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `medication\_dosages`  --  --  -- Table structure for table `medication\_ingredients`  --  DROP TABLE IF EXISTS `medication\_ingredients`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `medication\_ingredients` (  `id\_medication` int NOT NULL,  `id\_ingredient` int NOT NULL,  `unit\_of\_measurement` varchar(255) NOT NULL,  `quantity` decimal(10,2) NOT NULL,  PRIMARY KEY (`id\_medication`,`id\_ingredient`),  KEY `id\_ingredient` (`id\_ingredient`),  CONSTRAINT `medication\_ingredients\_ibfk\_1` FOREIGN KEY (`id\_medication`) REFERENCES `medications` (`id\_medication`),  CONSTRAINT `medication\_ingredients\_ibfk\_2` FOREIGN KEY (`id\_ingredient`) REFERENCES `ingredients` (`id\_ingredient`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `medication\_ingredients`  --  LOCK TABLES `medication\_ingredients` WRITE;  /\*!40000 ALTER TABLE `medication\_ingredients` DISABLE KEYS \*/;  INSERT INTO `medication\_ingredients` VALUES (1,1,'mg',250.00),(2,2,'mg',100.50),(3,3,'ml',50.75),(4,4,'g',15.25),(5,5,'IU',10000.00),(6,6,'mg',300.00),(7,7,'ml',20.50),(8,8,'g',7.50),(9,9,'CFU',1000.00),(10,10,'mg',50.00);  /\*!40000 ALTER TABLE `medication\_ingredients` ENABLE KEYS \*/;  UNLOCK TABLES;  --  -- Table structure for table `medication\_patents`  --  DROP TABLE IF EXISTS `medication\_patents`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `medication\_patents` (  `id\_patent` int NOT NULL AUTO\_INCREMENT,  `id\_medication` int NOT NULL,  `patent\_number` varchar(255) NOT NULL,  `patent\_type` enum('composition','method','formula') NOT NULL,  `expiration\_date` date NOT NULL,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_patent`),  KEY `id\_medication` (`id\_medication`),  CONSTRAINT `medication\_patents\_ibfk\_1` FOREIGN KEY (`id\_medication`) REFERENCES `medications` (`id\_medication`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `medication\_patents`  --  LOCK TABLES `medication\_patents` WRITE;  --  -- Table structure for table `medication\_side\_effects`  --  DROP TABLE IF EXISTS `medication\_side\_effects`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `medication\_side\_effects` (  `trial\_participants\_clinical\_trials\_id\_medication` int NOT NULL,  `trial\_participants\_clinical\_trials\_id\_trial` int NOT NULL,  `trial\_participants\_id\_participant` int NOT NULL,  `id\_side\_effect` int NOT NULL AUTO\_INCREMENT,  `side\_effect` varchar(255) NOT NULL,  `severity` enum('minor','moderate','strong') NOT NULL,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_side\_effect`),  KEY `trial\_participants\_id\_participant` (`trial\_participants\_id\_participant`,`trial\_participants\_clinical\_trials\_id\_medication`,`trial\_participants\_clinical\_trials\_id\_trial`),  CONSTRAINT `medication\_side\_effects\_ibfk\_1` FOREIGN KEY (`trial\_participants\_id\_participant`, `trial\_participants\_clinical\_trials\_id\_medication`, `trial\_participants\_clinical\_trials\_id\_trial`) REFERENCES `trial\_participants` (`id\_participant`, `clinical\_trials\_id\_medication`, `clinical\_trials\_id\_trial`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `medication\_side\_effects`  --  --  -- Table structure for table `medications`  --  DROP TABLE IF EXISTS `medications`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `medications` (  `id\_medication` int NOT NULL AUTO\_INCREMENT,  `name` varchar(255) NOT NULL,  `description` text,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_medication`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `medications`  --  --  -- Table structure for table `regulatory\_approvals`  --  DROP TABLE IF EXISTS `regulatory\_approvals`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `regulatory\_approvals` (  `id\_approval` int NOT NULL AUTO\_INCREMENT,  `id\_medication` int NOT NULL,  `country` varchar(255) NOT NULL,  `approval\_date` date NOT NULL,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_approval`),  KEY `id\_medication` (`id\_medication`),  CONSTRAINT `regulatory\_approvals\_ibfk\_1` FOREIGN KEY (`id\_medication`) REFERENCES `medications` (`id\_medication`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `regulatory\_approvals`  --  --  -- Table structure for table `researchers`  --  DROP TABLE IF EXISTS `researchers`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `researchers` (  `id\_researcher` int NOT NULL AUTO\_INCREMENT,  `studies\_id\_medication` int NOT NULL,  `studies\_id\_study` int NOT NULL,  `name` varchar(255) NOT NULL,  `email` varchar(255) DEFAULT NULL,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_researcher`),  KEY `studies\_id\_medication` (`studies\_id\_medication`,`studies\_id\_study`),  CONSTRAINT `researchers\_ibfk\_1` FOREIGN KEY (`studies\_id\_medication`, `studies\_id\_study`) REFERENCES `studies` (`id\_medication`, `id\_study`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `researchers`  --  --  -- Table structure for table `studies`  --  DROP TABLE IF EXISTS `studies`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `studies` (  `id\_medication` int NOT NULL,  `id\_study` int NOT NULL AUTO\_INCREMENT,  `start\_date` date NOT NULL,  `end\_date` date DEFAULT NULL,  `status` enum('in\_process','completed','canceled') NOT NULL,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_study`),  KEY `id\_medication` (`id\_medication`),  CONSTRAINT `studies\_ibfk\_1` FOREIGN KEY (`id\_medication`) REFERENCES `medications` (`id\_medication`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `studies`  --  DROP TABLE IF EXISTS `study\_phases`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `study\_phases` (  `studies\_id\_medication` int NOT NULL,  `studies\_id\_study` int NOT NULL,  `id\_phase` int NOT NULL AUTO\_INCREMENT,  `name` varchar(255) NOT NULL,  `description` text,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_phase`),  KEY `studies\_id\_medication` (`studies\_id\_medication`,`studies\_id\_study`),  CONSTRAINT `study\_phases\_ibfk\_1` FOREIGN KEY (`studies\_id\_medication`, `studies\_id\_study`) REFERENCES `studies` (`id\_medication`, `id\_study`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `study\_phases`  --  --  -- Table structure for table `trial\_participants`  --  DROP TABLE IF EXISTS `trial\_participants`;  /\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;  /\*!50503 SET character\_set\_client = utf8mb4 \*/;  CREATE TABLE `trial\_participants` (  `clinical\_trials\_id\_medication` int NOT NULL,  `clinical\_trials\_id\_trial` int NOT NULL,  `id\_participant` int NOT NULL AUTO\_INCREMENT,  `role` enum('microorganism','animal','patient') NOT NULL,  `created\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  `updated\_at` timestamp NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (`id\_participant`),  KEY `clinical\_trials\_id\_medication` (`clinical\_trials\_id\_medication`,`clinical\_trials\_id\_trial`),  KEY `idx\_trial\_participants\_fk` (`id\_participant`,`clinical\_trials\_id\_medication`,`clinical\_trials\_id\_trial`),  CONSTRAINT `trial\_participants\_ibfk\_1` FOREIGN KEY (`clinical\_trials\_id\_medication`, `clinical\_trials\_id\_trial`) REFERENCES `clinical\_trials` (`id\_medication`, `id\_trial`)  ) ENGINE=InnoDB AUTO\_INCREMENT=11 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  /\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;  --  -- Dumping data for table `trial\_participants`  --  /\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;  /\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;  /\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;  /\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;  /\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;  /\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;  /\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;  -- Dump completed on 2024-10-24 14:19:21 |

Команды модификации данных (DML)

Оболочки проектирования содержат средства визуального внесения данных в таблицы. Естественно, добавление записей (особенно массовое) может проводиться с помощью команд SQL, которые объединены в один сценарий. Итак, сценарий внесения данных в базу для MS SQL Server может быть таким:

*Листинг 2 — Массовое добавление данных*

|  |
| --- |
| -- Populating 'regulatory\_approvals' table  INSERT INTO regulatory\_approvals (id\_medication, country, approval\_date) VALUES      (1, 'USA', '2023-08-20'),      (2, 'EU', '2023-07-15'),      (3, 'Japan', '2023-06-30'),      (4, 'Canada', '2023-09-10'),      (5, 'UK', '2023-08-05'),      (6, 'Australia', '2023-09-30'),      (7, 'Germany', '2023-07-20'),      (8, 'France', '2023-08-15'),      (9, 'China', '2023-06-10'),      (10, 'India', '2023-09-25');  -- Populating 'ingredients' table  INSERT INTO ingredients (ingredient\_name) VALUES      ('Active Ingredient A'),      ('Compound B'),      ('Extract C'),      ('Substance D'),      ('Vitamin E'),      ('Mineral F'),      ('Herbal Extract G'),      ('Probiotic H'),      ('Enzyme I'),      ('Amino Acid J');  -- Populating 'medication\_ingredients' table  INSERT INTO medication\_ingredients (id\_medication, id\_ingredient, unit\_of\_measurement, quantity) VALUES      (1, 1, 'mg', 250.00),      (2, 2, 'mg', 100.50),      (3, 3, 'ml', 50.75),      (4, 4, 'g', 15.25),      (5, 5, 'IU', 10000),      (6, 6, 'mg', 300.00),      (7, 7, 'ml', 20.50),      (8, 8, 'g', 7.50),      (9, 9, 'CFU', 1000),      (10, 10, 'mg', 50.00);  -- Populating 'medication\_dosages' table  INSERT INTO medication\_dosages (clinical\_trials\_id\_medication, clinical\_trials\_id\_trial, dosage\_type, unit\_of\_measurement, dosage\_amount) VALUES      (1, 1, 'tablet', 'mg', 200.00),      (2, 2, 'capsule', 'mg', 150.00),      (3, 3, 'liquid', 'ml', 10.00),      (4, 4, 'liquid', 'g', 5.00),      (5, 5, 'injection', 'ml', 2.50),      (6, 6, 'tablet', 'mg', 300.00),      (7, 7, 'tablet', 'g', 15.00),      (8, 8, 'tablet', 'mg', 75.00),      (9, 9, 'capsule', 'mg', 200.00),      (10, 10, 'liquid', 'ml', 12.50);  -- Populating 'medication\_side\_effects' table  INSERT INTO medication\_side\_effects (trial\_participants\_clinical\_trials\_id\_medication, trial\_participants\_clinical\_trials\_id\_trial, trial\_participants\_id\_participant, side\_effect, severity) VALUES      (1, 1, 1, 'Headache', 'minor'),      (2, 2, 2, 'Dizziness', 'moderate'),      (3, 3, 3, 'Nausea', 'minor'),      (4, 4, 4, 'Skin irritation', 'minor'),      (5, 5, 5, 'Mild fever', 'minor'),      (6, 6, 6, 'Stomach discomfort', 'moderate'),      (7, 7, 7, 'Dry mouth', 'minor'),      (8, 8, 8, 'Sleepiness', 'moderate'),      (9, 9, 9, 'Allergic reaction', 'strong'),      (10, 10, 10, 'Diarrhea', 'moderate');  -- Populating 'medication\_patents' table  INSERT INTO medication\_patents (id\_medication, patent\_number, patent\_type, expiration\_date) VALUES      (1, 'PAT-123456', 'composition', '2035-06-20'),      (2, 'PAT-789012', 'method', '2033-03-15'),      (3, 'PAT-345678', 'formula', '2038-12-31'),      (4, 'PAT-901234', 'composition', '2036-09-05'),      (5, 'PAT-567890', 'method', '2034-04-20'),      (6, 'PAT-159236', 'formula', '2037-11-10'),      (7, 'PAT-456789', 'composition', '2039-05-30'),      (8, 'PAT-234567', 'method', '2032-02-10'),      (9, 'PAT-678901', 'formula', '2031-08-25'),      (10, 'PAT-012345', 'composition', '2030-01-15'); |

Выборка данных (DQL)

Запрос 1. Операция проекции. Осуществляется выбор только части полей таблицы, т.е. производится вертикальная выборка данных.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 71 — Выборка только id и названия медикаментов

Запрос 2. Операция селекции. Осуществляется горизонтальная выборка – в результат попадают только записи, удовлетворяющие условию.

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рисунок 72 — Выборка завершенных исследований

Запрос 3. Операции соединения. Здесь следует выделить декартово произведение и на его основе соединение по условию, а также естественное соединение (по одноименным полям или равенству полей с одинаковым смыслом).

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 73 — Декартово произведение медикаментов и исследователей

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 74 — Соединение медикаментов и клинических испытаний по id медикамента

Запрос 4. Операция объединения. Теоретико-множественные операции часто можно записать с помощью логических операций, примененных в конструкции WHERE запроса. Также можно задать логическим ИЛИ.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 75 — Объединение медикаментов

Запрос 5. Операция пересечения. В простых случаях эту операцию можно описать с помощью логической операции AND. В более сложных случаях эта операция определяется чаще всего с помощью подзапроса и ключевого слова EXISTS, которое показывает наличие похожего элемента во множестве, которое задается подзапросом.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 76 — Пересечение медикаментов с одобрениями в США и Европе

Запрос 6. Операция разности. Эта операция также определяется часто с помощью подзапроса с ключевым словом NOT EXISTS, которое показывает отсутствие элемента во множестве, задаваемом подзапросом.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 77 — Медикаменты без одобрений в США

Запрос 7. Операция группировки. Эта операция связана со своеобразной сверткой таблицы по полям группировки. Помимо полей группировки результат запроса может содержать итоговые агрегирующие функции по группам (COUNT, SUM, AVG, MAX, MIN).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 78 — Количество медикаментов в каждой фазе исследования

Запрос 8. Операция сортировки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 79 — Сортировка медикаментов в обратном алфавитном порядке

Запрос 9. Операция деления. Это самая нетривиальная операция реляционной алгебры, которая обычно применяется тогда, когда требуется найти все записи первой таблицы, которые соединяются естественным образом со всеми записями второй таблицы.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 80 — Медикаменты, у которых есть клинические испытания с пациентами

Процедуры

1. Процедура для получения средней продолжительности исследований для каждого медикамента. Код представлен в Листинге 3. Результат работы процедуры представлен на Рисунке 81.

*Листинг 3 — Процедура для получения средней продолжительности исследований для каждого медикамента*

|  |
| --- |
| CREATE PROCEDURE GetAverageStudyDurationByMedication()  BEGIN  SELECT m.id\_medication, AVG(DATEDIFF(ct.end\_date, ct.start\_date)) AS avg\_duration  FROM medications m  JOIN studies ct ON m.id\_medication = ct.id\_medication  GROUP BY m.id\_medication;  END; |

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 81 — Процедура для получения средней продолжительности исследований для каждого медикамента

1. Добавление нового клинического испытания с проверкой на допустимые даты. Код представлен в Листинге 4. Результат работы процедуры представлен на Рисунке 82.

*Листинг 4 — Процедура для добавления нового клинического испытания с проверкой на допустимые даты*

|  |
| --- |
| CREATE PROCEDURE AddClinicalTrialWithDateCheck(IN p\_med\_id INT, IN p\_name VARCHAR(255), IN p\_description TEXT, IN p\_start\_date DATE, IN p\_end\_date DATE)  BEGIN  DECLARE valid\_dates INT;    IF p\_start\_date > p\_end\_date THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'Start date cannot be after end date';  END IF;    INSERT INTO clinical\_trials (id\_medication, name, description, start\_date, end\_date, status)  VALUES (p\_med\_id, p\_name, p\_description, p\_start\_date, p\_end\_date, 'in\_process');  END; |

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 82 — Процедура для добавления нового клинического испытания с проверкой на допустимые даты

1. Количество участников испытаний для медикамента. Код представлен в Листинге 5. Результат работы процедуры представлен на Рисунке 83.

*Листинг 5 — Процедура для получения количества участников испытаний для медикамента*

|  |
| --- |
| CREATE PROCEDURE GetParticipantCountForMedication(IN p\_med\_id INT)  BEGIN  SELECT COUNT(\*) AS participant\_count  FROM trial\_participants tp  JOIN clinical\_trials ct ON tp.clinical\_trials\_id\_medication = ct.id\_medication AND tp.clinical\_trials\_id\_trial = ct.id\_trial  WHERE ct.id\_medication = p\_med\_id;  END; |

Изображение выглядит как текст, линия, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 83 — Процедура вывода информации об ученике

1. Обновления статуса исследования. Код представлен в Листинге 6. Результат работы процедуры представлен на Рисунке 84.

*Листинг 6 — Процедура для обновления статуса исследования*

|  |
| --- |
| CREATE PROCEDURE UpdateStudyStatus(IN p\_study\_id INT, IN p\_new\_status VARCHAR(20))  BEGIN  UPDATE studies  SET status = p\_new\_status  WHERE id\_study = p\_study\_id;  END; |

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

Рисунок 84 — Процедура для обновления статуса исследования

1. Обновление даты истечения патента с добавлением определенного количества лет. Код представлен в Листинге 5. Результат работы процедуры представлен на Рисунке 64.

*Листинг 7 — Процедура для обновления даты истечения патента с добавлением определенного количества лет*

|  |
| --- |
| CREATE PROCEDURE UpdatePatentExpirationDate(IN p\_patent\_id INT, IN p\_years\_to\_add INT)  BEGIN  UPDATE medication\_patents  SET expiration\_date = DATE\_ADD(expiration\_date, INTERVAL p\_years\_to\_add YEAR)  WHERE id\_patent = p\_patent\_id;  END; |

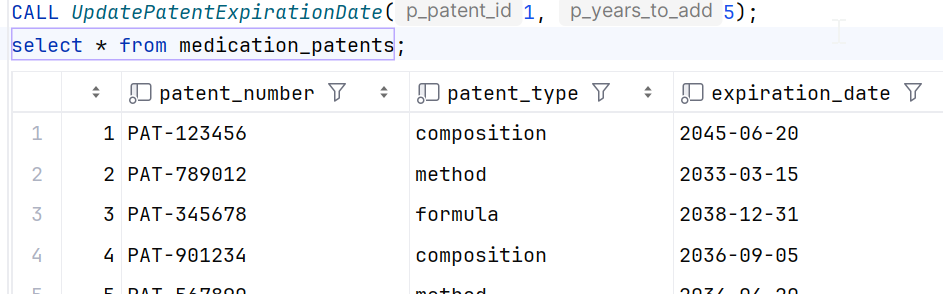


Рисунок 85 — Процедура для обновления даты истечения патента с добавлением лет

1. Процедура для обновления статуса всех исследований на "отменено" для определенного медикамента. Код представлен в Листинге 6. Результат работы процедуры представлен на Рисунке 65.

*Листинг 8 — Процедура вывода информации о питающихся*

|  |
| --- |
| CREATE PROCEDURE CancelStudiesForMedication(IN p\_med\_id INT)  BEGIN  UPDATE studies  SET status = 'canceled'  WHERE id\_medication = p\_med\_id;  END; |

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Рисунок 86 — Процедура для обновления статуса всех исследований на "отменено" для определенного медикамента

Функции

1. Функция для получения количества исследований для каждого медикамента. Код представлен в Листинге 9. Результат работы процедуры представлен на Рисунке 87.

*Листинг 9 — Функция для получения количества исследований для каждого медикамента*

|  |
| --- |
| CREATE FUNCTION GetStudyCount(p\_med\_id INT) RETURNS INT  READ SQL DATA  BEGIN  DECLARE study\_count INT;  SELECT COUNT(\*) INTO study\_count  FROM studies  WHERE id\_medication = p\_med\_id;  RETURN study\_count;  END; |

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 87 — Функция для получения количества исследований для каждого медикамента

1. Функция для проверки, является ли медикамент одобренным в конкретной стране. Код представлен в Листинге 10.

*Литстинг 10 —* *Функция проверки наличия одобрения в стране*

|  |
| --- |
| CREATE FUNCTION IsMedicationApproved(p\_med\_id INT, p\_country VARCHAR(255)) RETURNS BOOLEAN  READS SQL DATA  BEGIN  DECLARE approval\_exists BOOLEAN;  SELECT EXISTS (SELECT 1 FROM regulatory\_approvals WHERE id\_medication = p\_med\_id AND country = p\_country) INTO approval\_exists;  RETURN approval\_exists;  END;  RETURN AIDI;  END |

Изображение выглядит как текст, снимок экрана, программное обеспечение, линия

Автоматически созданное описание

Рисунок 88 — Функция проверки наличия одобрения в стране

1. Функция для получения количества участников испытаний с определенной ролью. Код представлен в Листинге 11.

*Литстинг 11 —* *Функция получения количества участников испытаний с определенной ролью*

|  |
| --- |
| CREATE FUNCTION GetParticipantCountByRole(p\_role VARCHAR(20)) RETURNS INT  READS SQL DATA  BEGIN  DECLARE participant\_count INT;  SELECT COUNT(\*) INTO participant\_count  FROM trial\_participants  WHERE role = p\_role;  RETURN participant\_count;  END; |

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Рисунок 89 — Функция получения количества участников с определенной ролью

1. Функция для проверки, есть ли у медикамента клинические испытания в процессе. Код представлен в Листинге 12.

*Литстинг 12 — Функция для проверки текущих испытаний медикамента*

|  |
| --- |
| CREATE FUNCTION HasActiveClinicalTrials(p\_med\_id INT) RETURNS BOOLEAN  READ SQL DATA  BEGIN  DECLARE has\_active\_trials BOOLEAN;  SELECT EXISTS (SELECT 1 FROM clinical\_trials WHERE id\_medication = p\_med\_id AND status = 'in\_process') INTO has\_active\_trials;  RETURN has\_active\_trials;  END; |

Изображение выглядит как текст, программное обеспечение, Шрифт, число

Автоматически созданное описание

Рисунок 90 — Функция для проверки текущих испытаний медикамента

Триггеры

1. Триггер для проверки допустимого количества дозировки. Код представлен в Листинге 13.

*Листинг 13 — Триггер проверки дозировки*

|  |
| --- |
| CREATE TRIGGER check\_dosage\_limit  BEFORE INSERT ON medication\_dosages  FOR EACH ROW  BEGIN  IF NEW.dosage\_amount > 1000 THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'Dosage amount exceeds limit';  END IF;  END; |

1. Триггер для обновления статуса испытания при добавлении побочного эффекта. При добавлении записи о побочном эффекте в таблицу medication\_side\_effects, этот триггер может обновлять статус испытания в таблице clinical\_trials на "completed", если побочный эффект имеет высокую степень тяжести ("strong"). Код представлен в Листинге 14.

*Листинг 14 — Триггер обновления статуса испытания при добавлении побочного эффекта*

|  |
| --- |
| CREATE TRIGGER update\_trial\_status\_on\_severe\_side\_effect  AFTER INSERT ON medication\_side\_effects  FOR EACH ROW  BEGIN  IF NEW.severity = 'strong' THEN  UPDATE clinical\_trials  SET status = 'completed'  WHERE id\_trial = NEW.id\_trial;  END |

1. При добавлении одобрения в нескольких странах для одного препарата в таблице regulatory\_approvals, этот триггер может обновлять статус испытания в таблице clinical\_trials на "completed". Код представлен в Листинге 15.

*Листинг 15 — Обновление статуса испытания при одобрении в нескольких странах*

|  |
| --- |
| CREATE TRIGGER update\_trial\_status\_on\_multiple\_approvals  AFTER INSERT ON regulatory\_approvals  FOR EACH ROW  BEGIN  DECLARE approval\_count INT;  SELECT COUNT(\*) INTO approval\_count  FROM regulatory\_approvals  WHERE id\_medication = NEW.id\_medication;    IF approval\_count > 1 THEN  UPDATE clinical\_trials  SET status = 'completed'  WHERE id\_trial = (SELECT id\_trial FROM studies WHERE id\_medication = NEW.id\_medication);  END IF;  END; |

1. Этот триггер может отправлять уведомление (например, по электронной почте) владельцу патента, когда дата истечения (expiration\_date) патента близко. Код представлен в Листинге 16.

*Листинг 16 — обновления статуса испытания одобрении в нескольких странах*

|  |
| --- |
| CREATE TRIGGER notify\_patent\_expiration  BEFORE UPDATE ON medication\_patents  FOR EACH ROW  BEGIN  IF DATEDIFF(NEW.expiration\_date, CURRENT\_DATE) BETWEEN 0 AND 30 THEN  INSERT INTO notifications (recipient, message)  VALUES (  (SELECT email FROM researchers WHERE id\_medication = NEW.id\_medication),  'Your patent is expiring soon!'  );  END IF;  END; |

Оконные функции

Ниже приведены примеры оконных функций из списка:

1. **Агрегатные функции**

*Литстинг 17 — Агрегатные функции*

|  |
| --- |
| SELECT id\_patent, id\_medication, patent\_number, patent\_type, expiration\_date, created\_at, updated\_at,  *COUNT*(*\**) OVER (PARTITION BY patent\_type) AS count\_patents\_type,  *SUM*(CASE WHEN patent\_type = 'composition' THEN 1 ELSE 0 END) OVER () AS sum\_composition\_patents,  *AVG*(*DATEDIFF*(expiration\_date, created\_at)) OVER () AS avg\_patent\_duration,  *MAX*(expiration\_date) OVER () AS max\_expire\_date,  *MIN*(created\_at) OVER () AS min\_created\_at FROM medication\_patents; |

Изображение выглядит как текст, число, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 91 — Вызов агрегатных функций

1. **Ранжирующие функции**

*Литстинг 18 — Ранжирующие функции*

|  |
| --- |
| SELECT id\_patent, id\_medication, patent\_number, patent\_type, expiration\_date, created\_at, updated\_at,  ROW\_NUMBER() OVER (ORDER BY expiration\_date) AS row\_num,  RANK() OVER (ORDER BY expiration\_date DESC) AS rank\_num,  DENSE\_RANK() OVER (ORDER BY expiration\_date DESC) AS dense\_rank\_num,  NTILE(4) OVER (ORDER BY expiration\_date DESC) AS ntile\_num  FROM medication\_patents; |

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 92 — Вызов ранжирующих функций

1. **Функции смещения**

*Литстинг 19 — Функции смещения*

|  |
| --- |
| SELECT id\_patent, id\_medication, patent\_number, patent\_type, expiration\_date, created\_at, updated\_at,  LAG(expiration\_date, 1) OVER (ORDER BY expiration\_date) AS prev\_expiration,  LEAD(expiration\_date, 1) OVER (ORDER BY expiration\_date) AS next\_expiration,  FIRST\_VALUE(expiration\_date) OVER (ORDER BY expiration\_date) AS first\_expiration,  LAST\_VALUE(expiration\_date) OVER (ORDER BY expiration\_date) AS last\_expiration  FROM medication\_patents; |

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 93 — Вызов функций смещения

Заключение

В течение семестра была создана база данных

Список использованных источников

1. Дейт, К. Дж. Введение в системы баз данных / учебное пособие / К. Дж. Дейт; — Москва: «Вильяме» 2005. — 1316 с.
2. Дьяков И.А. Базы данных. Язык SQL: Учебное пособие. — Тамбов: ТГТУ, 2004. — 80 с.
3. Шварц Б., Зайцев П., Ткаченко В., Заводны Дж., Ленц А., Бэллинг Д. MySQL. Оптимизация производительности, 2-е издание. — Пер. с англ. — Спб.: Символ-Плюс, 2010. — 832 с.: ил.
4. Э.В. Сысоев, Е.В. Бурцева. Базы данных: лекции к курсу. — Тамбов : Изд-во Тамб. гос. техн. ун-та, 2007. — 48 с.
5. Баженова И.Ю. Разработка распределенных приложений баз данных: Курс лекций. — М.: МГУ им. М.В. Ломоносова, 2006. — 203 с.
6. Зиборов В.В. MS Visual C++ 2010 в среде .NET. Библиотека программиста. — Спб.: Питер, 2012. - 320 с.: ил.