# Что включает в себя база данных?

**CSV данные в области  лазерно-индуцированной спектроскопии картофеля -** [**LIBS potatoes (kaggle.com)**](https://www.kaggle.com/datasets/adolfocobo/libs-potatoes?select=LIBS_spectra_dry_potatoes_withcolumns.csv)

# Структура:

Данные, которые были для проекта, получены в результате измерения, кторые проводились в инженерной группе по фотонике Университета Кантабрии (Испания).

Мы работаем над лазерно-индуцированной спектроскопией рассеяния. Этот спектроскопический метод улавливает свет, испускаемый плазмой, генерируемой лазерным импульсом, который удаляет небольшое количество материала. Информация в каждом спектре представляет собой химический состав удаляемого материала на атомном уровне, через линии излучения на определенных длинах волн каждого атомного вида. Там также должна быть некоторая молекулярная информация, поскольку некоторые молекулы снова рекомбинируются,когда плазма охлаждается, и они также могут излучать некоторое количество света.

# Реализация:

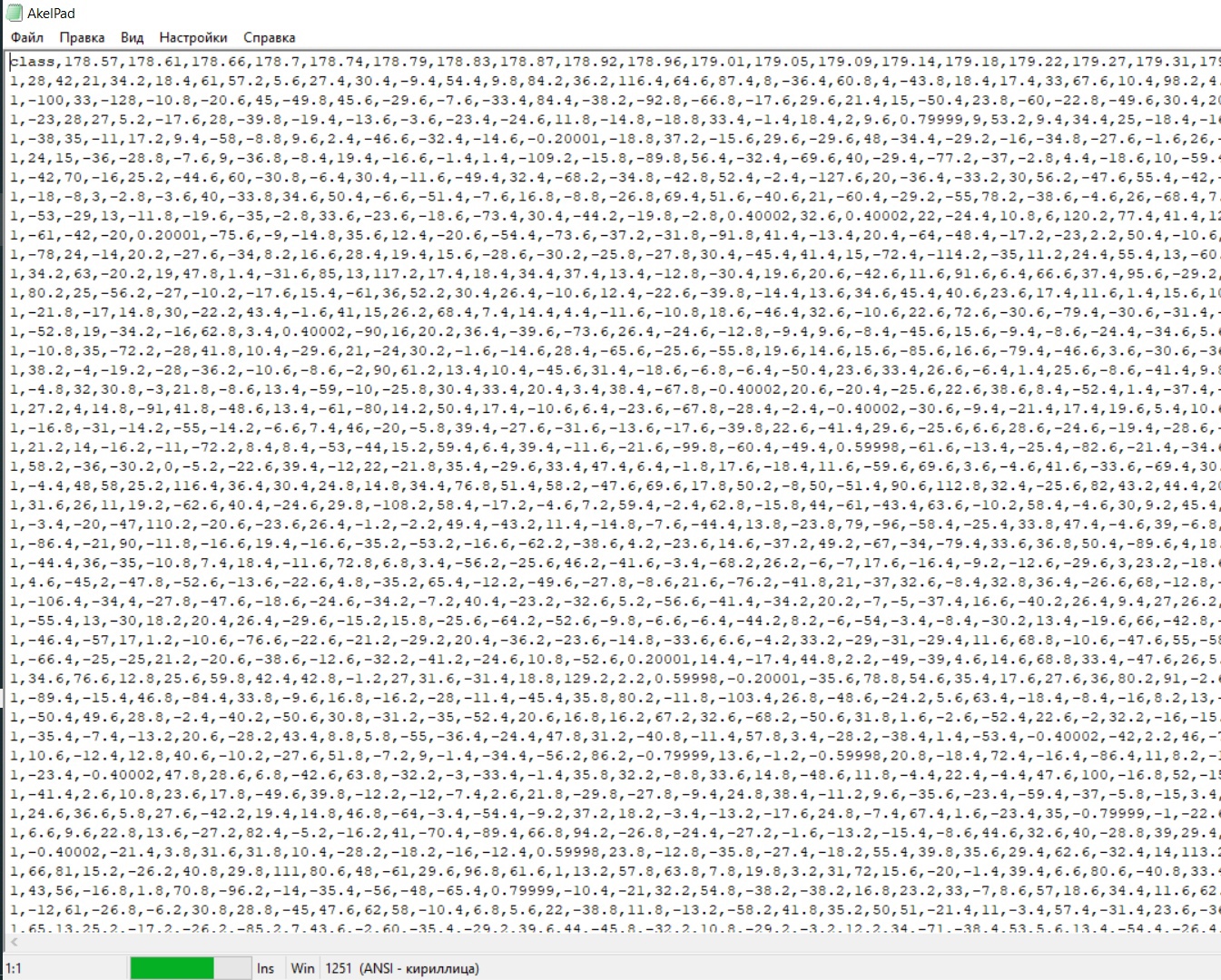
Не знаю надо не надо, если что убери совсем этот заголовок

# Участие каждого члена команды:

* Крайнов Константин Олегович
  + Распределил задачи между участниками проекта.
  + Отслеживал прогресс выполнения задач.
  + Создал коллективный проект с доступом на разрешение
  + Создал базу данных.
  + Визуализировал данные на Python.
  + Нормализовала данные базы данных.
  + Проводил тестирование и проверку выполнения задач.
  + Развернул базу данных на web-сервере <https://db4free.net>
  + Создал Jupiter notebook для извлечения основных данных из созданной базы данных.
* Джабаров Артем Дамирович
* Создал триггеры в базе данных (для автоматического удаления связанных записей из других таблиц при удалении определенных данных из таблицы)
  + Проводил тестирование и проверку выполнения задач.
  + Проводил проверку заполнения данных.
  + Создал Docker для разворачивания созданной базы данных.
* Гребцова Анастасия Алексеевна
  + Описала проект, созданную БД (документация) и участие (авторство) каждого члена команды.
* Создала представление (виртуальные таблицы v\_experiment\_data.sql, v\_lyambda\_count\_test.sql, v\_potato\_count\_test.sql), для наиболее удобного получения доступа к записям.
  + Провела тестирование и проверку выполнения задач.
* Куренкова Полина Генадьевна
  + Создал EER-диаграмму
  + Создала Надо описать Полинину работу!
  + Провела тестирование и проверку выполнения задач.

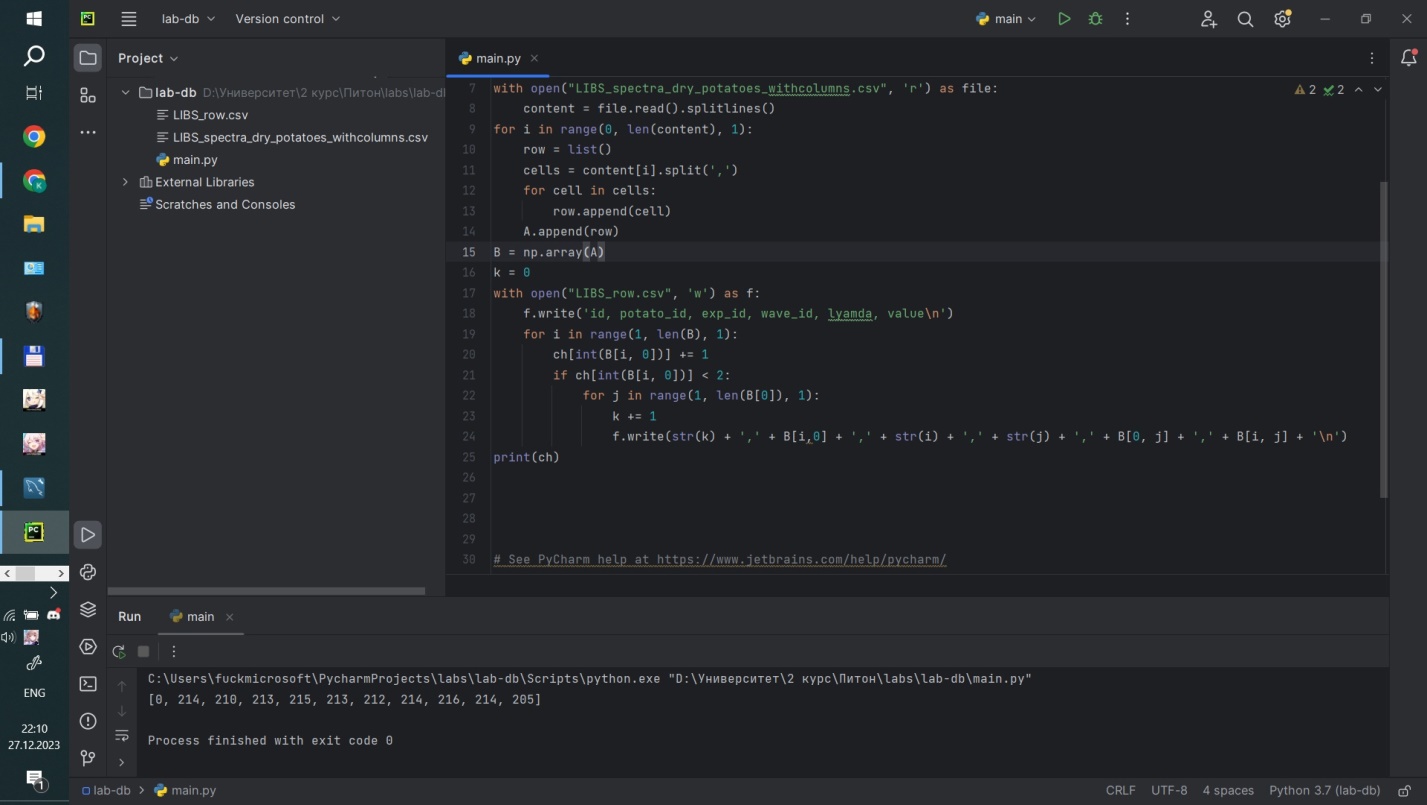
# Процесс создания и нормализации базы данных:

Начальный (необработанный файл)

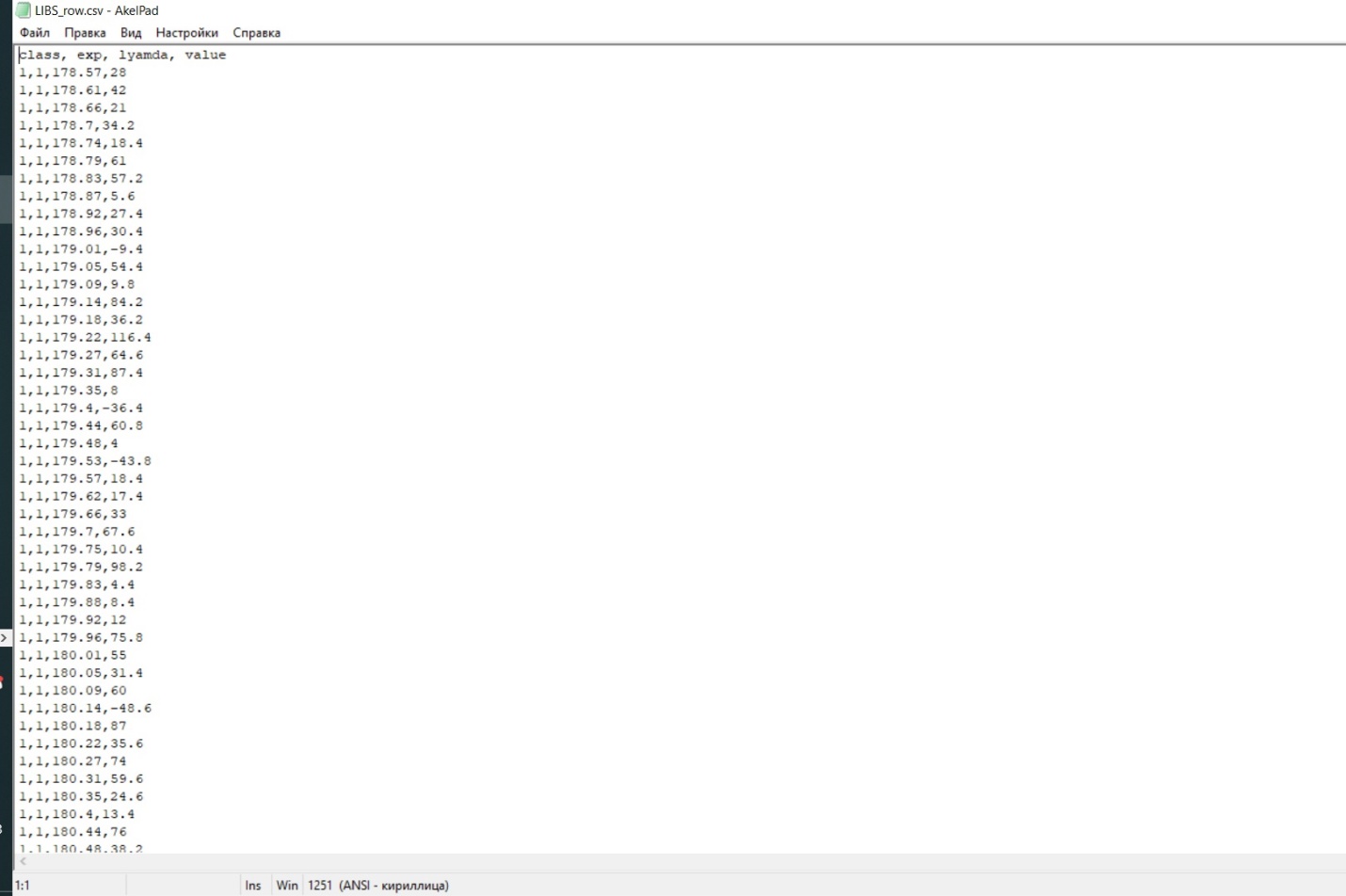


Конвертированный полный файл с данными полученными в результате спектрографии размером 0.5Гб на 33 млн строк. Файл имеет слишком большой размер для прямой записи в базы данных, т.е. количество строк и колонок в файле превышает количество доступных колонок таблицы в базе данных.

Обрабатываем файл, с помощью программы прописанной в Python.



Конвертированный файл, созданный на Python, грузится стандартным загрузчиком Worhbench "Table Data Import Wizard" в создаваемую автоматически таблицу "libst\_row".

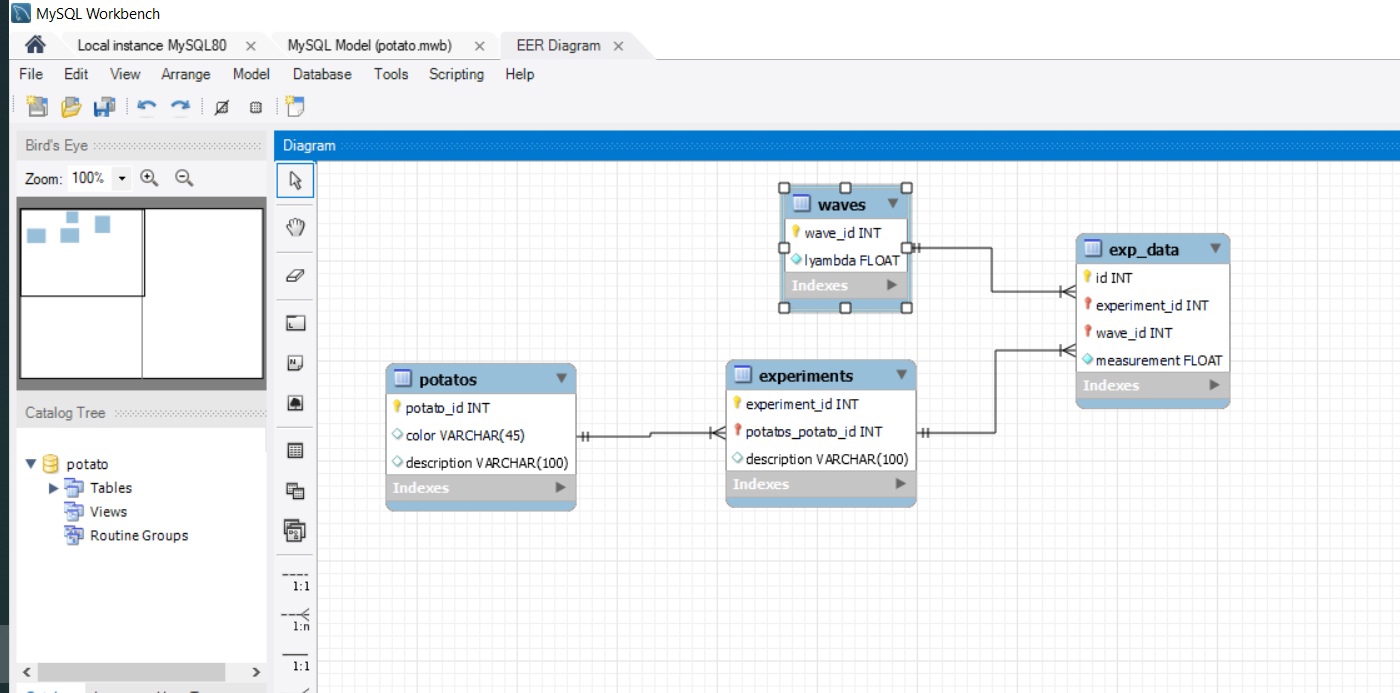


# Создание ER-диаграммы:

Вот общий алгоритм создания ER-диаграммы в MySQL Workbench:

1. Запустите MySQL Workbench и откройте окно "SQL Editor".
2. Импортируйте структуру базы данных в MySQL Workbench, используя команду "Import". Выберите файл с описанием базы данных (обычно это файл .sql).
3. Создайте новую модель данных в окне "Models" (Модели данных).
4. В окне "Models" выберите созданную модель данных и нажмите правой кнопкой мыши. Выберите "Create Diagram" (Создать диаграмму) для создания новой ER-диаграммы.
5. В окне "Diagram" (Диаграмма) перетащите сущности (таблицы) из модели данных на диаграмму. Вы можете изменить их имена и внешний вид, щелкнув правой кнопкой мыши и выбрав соответствующие опции.
6. Для каждой сущности вы можете добавить атрибуты (поля) и определить их типы данных.
7. Определите связи между сущностями, перетащив линии связи между ними и указав тип связи (один-ко-многим, многие-ко-многим и т.д.).
8. Добавьте дополнительные элементы, такие как ограничения, триггеры и индексы, если это необходимо.
9. Сохраните ER-диаграмму, чтобы сохранить ее в файл или в базу данных.
10. Сохраните ER-диаграмму, чтобы сохранить ее в файл или в базу данных.

Готовая диаграмма:



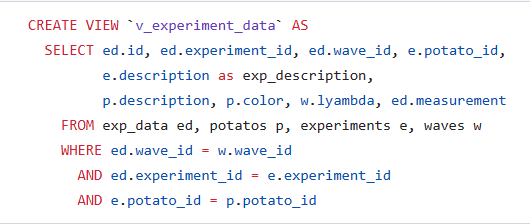
# Процесс создания триггеров:

Данный код содержит определение трех триггеров в базе данных. Давайте разберем каждую часть по порядку:  
  
1. DELIMITER // - Эта строка устанавливает новый разделитель для команд SQL. Обычно по умолчанию разделителем служит точка с запятой (;), но здесь мы устанавливаем двойной слэш (//) в качестве разделителя, чтобы обозначить конец каждой команды.  
  
2. DROP TRIGGER IF EXISTS TD\_potatos // - Эта команда удаляет триггер с именем TD\_potatos, если он существует. IF EXISTS используется для того, чтобы избежать ошибок, если триггер еще не создан.  
  
3. CREATE TRIGGER TD\_potatos BEFORE DELETE ON potatos FOR EACH ROW - Создает триггер с именем TD\_potatos, который срабатывает перед операцией удаления (BEFORE DELETE) в таблице potatos для каждой строки.  
  
4. BEGIN - Начало блока кода триггера.  
  
5. DELETE FROM experiments e WHERE e.potato\_id = OLD.potato\_id; - Это действие триггера, которое удаляет связанные записи из таблицы experiments, где potato\_id равен старому значению (OLD.potato\_id) строки, которая была удалена из таблицы potatos.  
  
6. END // - Завершение блока кода триггера.  
  
7. Аналогичные операции выполняются для триггеров TD\_experiments и TD\_waves, только в соответствующих таблицах и с их соответствующими связанными данными.  
  
8. DELIMITER; - Восстанавливает стандартный разделитель команд SQL.  
  
Таким образом, этот код создает три триггера в базе данных, каждый из которых предназначен для автоматического удаления связанных записей из других таблиц при удалении определенных данных из таблицы potatos, experiments и waves.

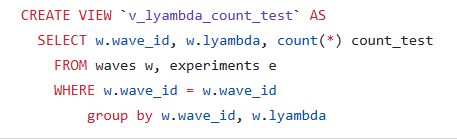
# Процесс создания представления:

Представление — это виртуальная таблица, содержимое которой определяется запросом.

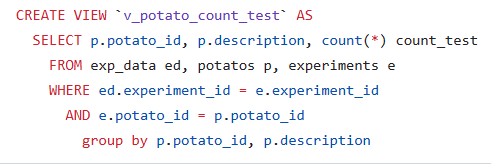
1.Данный код соединяет таблицы potatos , experiments, exp.data и waves. Связь их можно посмотреть на ER-диаграмме. Объединяет столбцы из уже существующих четырёх таблиц (ид. эксперимента, ид. длины волны, ид. картошки, описание эксперимента, описание картошки, цвет картошки, длина волны, показатель яркости света).



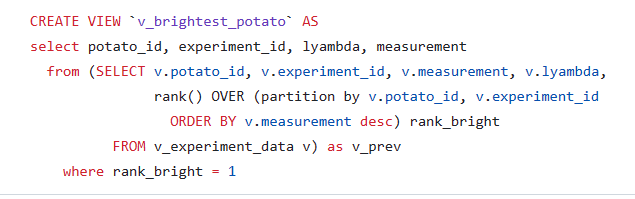
2.Данный код соединяет таблицы exp.data и waves. Связь их можно посмотреть на ER-диаграмме.



3.Данный код соединяет таблицы potatos , experiments. Связь их можно посмотреть на ER-диаграмме.



4. В этом коде мы выбираем из всех измерений над картошкой то, где самое яркая волна (самый высокий показатель) и записываем их.



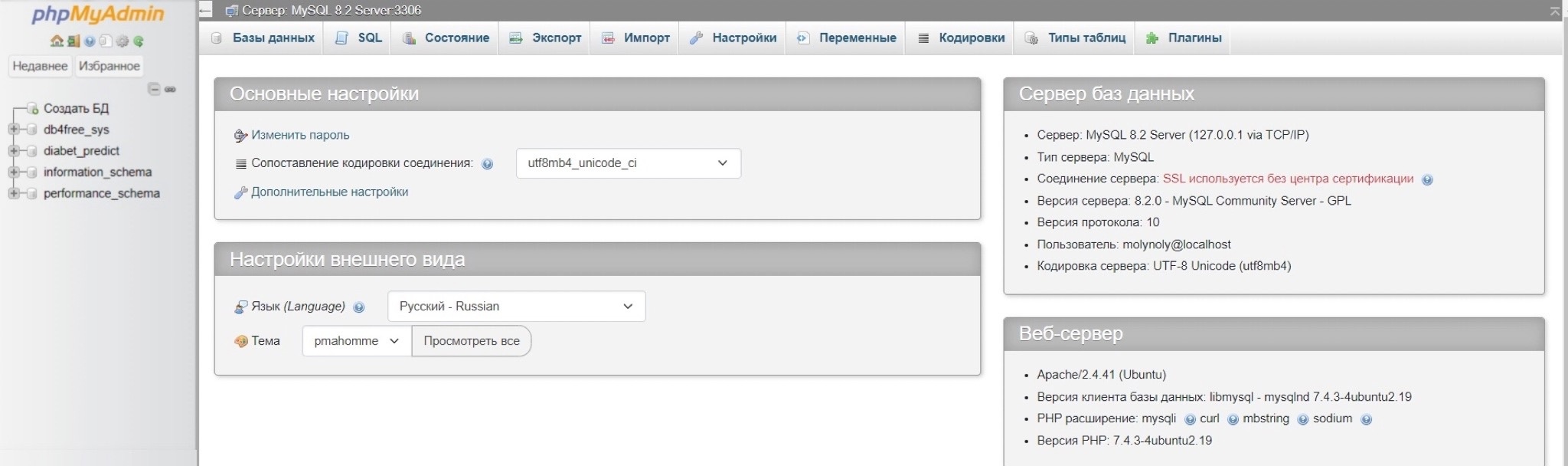
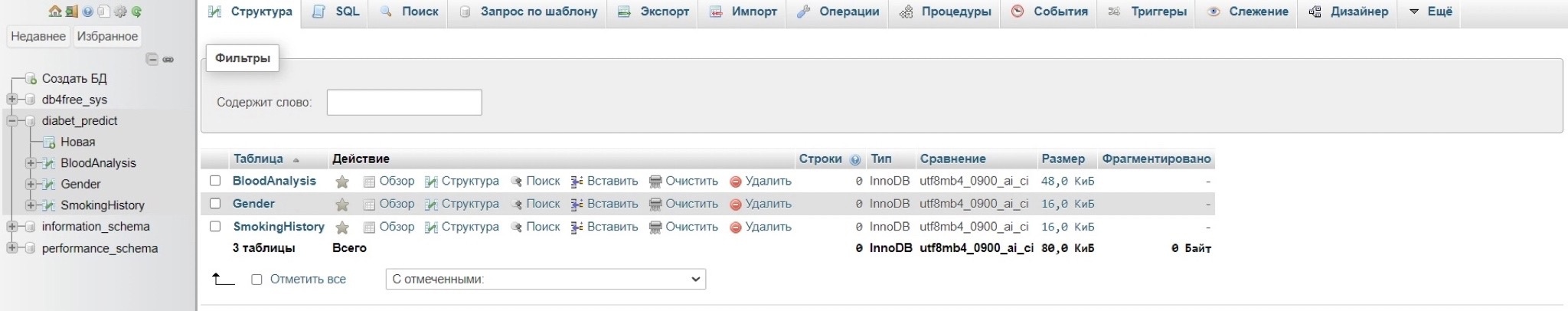
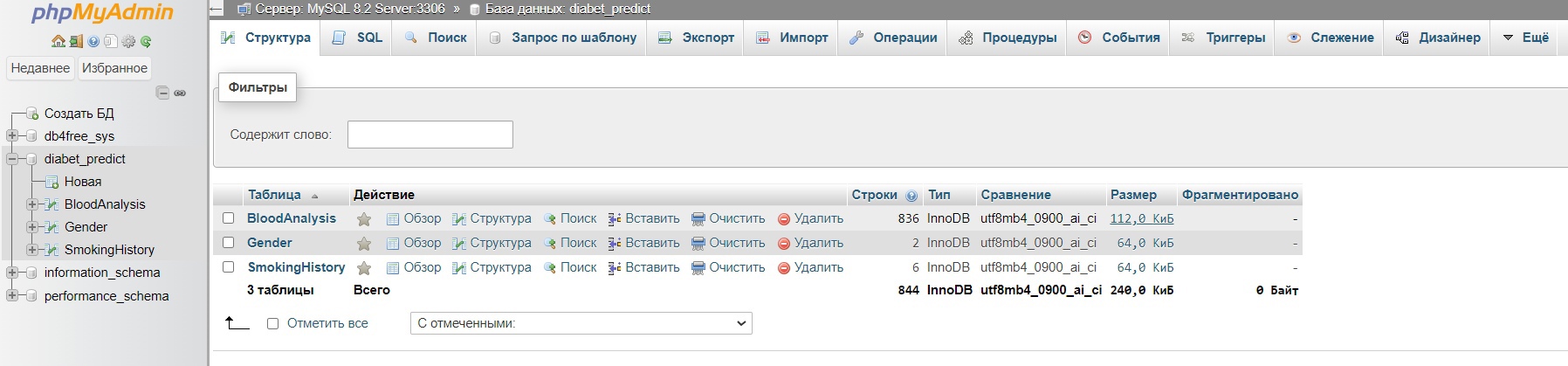
Таким образом 4 кода с функцией представление, каждый из которых упрощает и уменьшает время на вывод запроса.

# Что делала Полина нужно расписать и если ты сделаешь графики или доп. задания.

Это пример выполнения доп.заданий

**Как использовать проект?**

**Установка базы данных на**[**db4free.net**](https://db4free.net/)

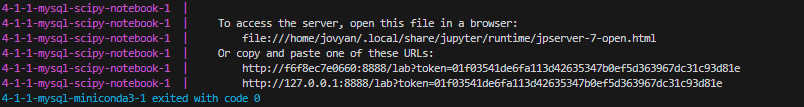
1. **Регистрация на сайте db4free.net Проходим регистрацию на**[**db4free.net**](https://db4free.net/signup.php)**, где также указываем название новой базы данных mysql diabet\_predict. Подтверждаем почту и заходим на сайт [phpMyAdmin](https://db4free.net/phpMyAdmin/index.php?route=/)**
2. **В левой части сайта мы можем увидеть уже созданную базу данных diabet\_predict.[](https://github.com/testmapper-official/4-1-1-mysql/blob/main/data/db4free1.png) Переходим в раздел SQL, куда пишем наши скрипты patients\_data.sql по созданию пустых таблиц, запускаем. В левой панели в нашей базе данных мы можем увидеть только что созданные таблицы.[](https://github.com/testmapper-official/4-1-1-mysql/blob/main/data/db4free2.png)**
3. **Далее заходим во вкладку Импорт и загружаем наш csv-файл, откуда импортируем данные в нашу базу данных. Файл нужно загружать строго по инструкции на сайте.**
4. **Снова переходим в раздел SQL и пишем скрипт dataset2.sql по импорту данных из файла csv. Только в скрипте нужно написать то имя файла, которое в указывали при импорте. Далее заходим в раздел Структура и проверяем заполнились ли таблицы.[](https://github.com/testmapper-official/4-1-1-mysql/blob/main/data/db4free3.png)**

**Как подключиться к базе данных на сервере**[**db4free.net**](https://db4free.net/)

**При подключении указать следующие параметры:**

* **Host: db4free.net**
* **Port: 3306**
* **Database: diabet\_predict**
* **User: diabetic**
* **Password: dbbasa58**

**Как запустить проект при помощи Docker**

1. **Для начала нужно установить [Docker](https://www.docker.com/get-started/).**
2. **После установки Docker, запустите его, а затем перейдите в командную строку с директорией этого проекта. Как это сделать, можно почитать**[**здесь**](https://qna.habr.com/q/555833)**.**
3. **Когда вы находитесь в корневой папке проекта, пропишите следующие команды: docker-compose build и docker-compose up**
4. **Дальше произойдет установка образов, которые требуются для проекта. Как только установка завершится, у вас запустятся контейнеры под следующими портами:**
   * **127.0.0.1:8080 : phpMyAdmin, на котором в дальнейшем будет разворачиваться база данных diabet\_predict *MySQL 8.0*. В файле docker-compose.yml указаны данные для авторизации (по умолчанию).**
   * **127.0.0.1:8888 : Jupyter Notebook, на котором можно выполнить визуализацию и анализ данных. Все файлы для работы с данными хранятся в директории */notebooks/*. Для авторизации требуется токен, который генерируется каждый раз случайно *(чтобы его получить, достаточно найти его при запуске в командной строке)*.[](https://github.com/testmapper-official/4-1-1-mysql/blob/main/data/jupyter_auth.png)**