

**Задание 1.** Используя справочные материалы по MySQL Workbench, опишите назначение пунктов меню Management (“Управление”), “Instance” (“Экземпляр БД”) и “Performance” (“Производительность”).

### **Раздел “Management”:**

1. Раздел “Server Status”. В разделе отображается общая информация о сервере и подключении к нему. Информация логически сгруппирована. Можно выделить следующие группы:

- Общая информация (например, название хоста, номер порта, версия БД).
- Настройки сервера (например, включен ли брандмауэр, используется ли SSL)
- Каталоги сервера
- Сводка по используемым ресурсам компьютера (ОЗУ, процессор и т. д.)
- Настройки соединения SSL (если SSL включена).

2. Раздел "Client Connections" предоставляет информацию о текущих подключениях к серверу базы данных MySQL. Это важный раздел, который позволяет администраторам и разработчикам отслеживать активные соединения с базой данных и управлять ими при необходимости. Ниже приведены основные аспекты, которые вы можете наблюдать и управлять в данном разделе:

- Информация о подключениях: Вы сможете увидеть список активных подключений к серверу, включая IP-адреса клиентов, имена пользователей, номера потоков запросов и другие сведения о текущей активности.
- Статус подключений: Вы сможете определить статус каждого подключения, такой как установлено ли подключение, активно ли оно, отправляет или принимает данные.
- Управление подключениями: Вы сможете управлять подключениями, например, завершать их, если есть необходимость или если определённое подключение вызывает проблемы. Это позволяет освободить ресурсы сервера и улучшить производительность.
- Дополнительная информация: В разделе также может быть предоставлена дополнительная информация о запросах, выполняемых клиентами, таких как длительность выполнения запросов, объем передаваемых данных и другие метрики, которые помогут вам лучше понять активность пользователей и при необходимости принять меры.

3. Раздел "User and Privileges" предоставляет возможность управлять пользователями и их правами доступа к базе данных MySQL. Этот раздел играет ключевую роль в обеспечении безопасности данных, контроле доступа и управлении пользователями базы данных. Вот основные аспекты, которые вы можете наблюдать и настраивать в данном разделе:

- Управление пользователями: Вы можете создавать новых пользователей, удалять существующих, изменять их пароли и другие атрибуты учетных записей. Это позволяет эффективно управлять списком пользователей, обеспечивая доступ только тем, кому это действительно необходимо.
- Назначение привилегий: Вы можете определять привилегии доступа для каждого пользователя, контролируя какие операции он может выполнять в базе данных. Например, вы можете предоставить права на чтение данных, запись, изменение структуры таблиц, управление пользователями и другие действия.

- Управление ролями: Вы можете определять роли и группы пользователей, что упрощает управление правами доступа. Назначая роли пользователям, вы повышаете безопасность и снижаете риск ошибок в управлении привилегиями.
- Отзыв привилегий: Если необходимо, вы также можете отозвать привилегии у пользователя, если они стали ненужными или пользователь выходит из определенного контекста использования данных.

4. Раздел "Status and System Variables" предоставляет информацию о текущем статусе вашего сервера базы данных MySQL и позволяет просматривать и изменять различные системные переменные, которые влияют на работу сервера. Этот раздел является важным инструментом для мониторинга и настройки параметров сервера MySQL. Вот основные аспекты, которые вы можете наблюдать и редактировать в данном разделе:

- Информация о статусе: Вы можете просмотреть общую информацию о текущем состоянии сервера, такую как количество подключений, потоков, состояние работы и другие ключевые метрики. Это позволяет вам оперативно отслеживать работу сервера и выявлять потенциальные проблемы.
- Системные переменные: Вы можете просматривать и изменять системные переменные MySQL, такие как параметры конфигурации, настройки памяти, размеры буферов, ограничения на соединения и другие параметры. Это позволяет вам настраивать сервер в соответствии с потребностями вашего проекта и оптимизировать его производительность.
- Мониторинг ресурсов: Раздел "Status and System Variables" также может включать информацию о использовании ресурсов сервера, таких как использование CPU, памяти, дискового пространства и других системных ресурсов. Это позволяет вам отслеживать нагрузку на сервер и принимать меры для оптимизации его работы.
- Настройка и оптимизация: Путем изменения системных переменных по вашему усмотрению, вы можете настроить сервер для оптимальной производительности и безопасности. Например, вы можете увеличить размер буфера или изменить параметры кэширования для улучшения скорости выполнения запросов.

5. Раздел "Data Export" в MySQL Workbench предоставляет возможность экспортировать данные из вашей базы данных MySQL в различные форматы для создания резервных копий, обмена данными с другими системами или для анализа информации во внешних приложениях. Вот основные аспекты, которые вы можете использовать при экспорте данных из MySQL Workbench:

- Выбор данных для экспорта: Вы можете выбирать конкретные таблицы, запросы или даже всю базу данных для экспорта. Это позволяет вам гибко настроить объем данных, который вы хотите экспортировать.
- Форматы экспорта: Раздел "Data Export" поддерживает различные форматы для экспорта данных, такие как SQL, CSV, JSON, XML и другие. Вы можете выбрать соответствующий формат в зависимости от ваших потребностей и требований вашего проекта.
- Настройки экспорта: Вы можете настраивать параметры экспорта, такие как выбор кодировки символов, настройку разделителей для CSV файлов, включение или исключение определенных данных и другие опции для точной настройки процесса экспорта.

- **Расписание экспорта:** MySQL Workbench также предоставляет возможность создания расписания экспорта данных, что позволяет автоматизировать процесс создания резервных копий или обмена данными в удобное для вас время.

6. Раздел "Data Import/Restore" в MySQL Workbench предназначен для импорта данных в вашу базу данных MySQL из внешних источников и для восстановления данных из резервных копий или дампов баз данных. Этот раздел позволяет вам легко загружать данные из различных форматов и источников в вашу базу данных. Вот основные аспекты, которые вы можете использовать для импорта и восстановления данных:

- **Выбор источника данных:** Вы можете импортировать данные из различных источников, таких как SQL файлы, CSV файлы, JSON файлы, дампы баз данных и другие форматы данных. Это позволяет вам легко передавать информацию в вашу базу данных для обновления или восстановления.
- **Параметры импорта:** В разделе "Data Import/Restore" вы можете задать различные параметры импорта, такие как тип данных (текстовые, числовые и т. д.), разделители столбцов, настройки кодировки и другие опции, которые помогут правильно интерпретировать и загрузить данные.
- **Обработка конфликтов:** При импорте данных возможны конфликты, например, дубликаты записей или ограничения целостности данных. В MySQL Workbench вы можете настроить способы обработки таких конфликтов, выбрав, например, замену существующих записей или игнорирование дубликатов.
- **Восстановление данных:** Когда дело касается восстановления данных из резервной копии или дампа базы данных, раздел "Data Import/Restore" предоставляет вам возможность быстро восстановить данные после сбоя или нештатной ситуации.

## **Раздел "Instance":**

1. Раздел "Startup / Shutdown" в MySQL Workbench предоставляет возможность управлять процессом запуска и остановки экземпляра базы данных MySQL. Этот раздел является ключевым для управления жизненным циклом вашего сервера баз данных, позволяя вам инициировать запуск сервера, его остановку или перезапуск. Вот основные детали этого раздела:

- **Запуск сервера:** При помощи раздела "Startup / Shutdown" вы можете инициировать процесс запуска сервера базы данных MySQL. Это позволяет вам активировать доступ к вашей базе данных и начать обработку запросов.
- **Остановка сервера:** Процесс остановки сервера также осуществляется через данный раздел. При остановке сервера все активные соединения прекращаются, и работа сервера заканчивается.
- **Перезапуск сервера:** В случае необходимости перезапустить сервер, вы также можете воспользоваться функциональностью раздела "Startup / Shutdown". Это может быть полезно при внесении изменений в настройки сервера или после установки обновлений.
- **Управление циклом жизни сервера:** Раздел "Startup / Shutdown" предоставляет вам возможность эффективно управлять жизненным циклом сервера MySQL, обеспечивая контроль над его запуском, остановкой и перезапуском в удобное для вас время.

2. Раздел "Server Logs" в MySQL Workbench предоставляет возможность просмотра и анализа логов действий, происходящих на вашем сервере базы данных MySQL. Логи

сервера содержат ценную информацию о событиях, ошибках, успешных операциях и других действиях, происходящих на сервере. Вот основные аспекты, которые стоит учитывать при работе с логами сервера MySQL:

- **Типы логов:** В этом разделе вы можете увидеть различные типы логов, такие как лог ошибок, лог запросов, лог подключений и другие, в зависимости от настроек вашего сервера. Каждый тип логов предоставляет информацию о конкретном аспекте работы сервера.
- **Информация в логах:** Логи содержат информацию о различных событиях, таких как запуск и остановка сервера, ошибки выполнения запросов, успешные операции, информация о подключениях к серверу, предупреждения и многое другое. Эта информация может быть полезна для отслеживания работы сервера и диагностики проблем.
- **Мониторинг производительности:** Просмотр логов сервера также предоставляет возможность мониторинга производительности базы данных. Вы можете анализировать время выполнения запросов, количество запросов, использование ресурсов и другие метрики производительности для выявления узких мест и оптимизации запросов.
- **Поиск и фильтрация:** Вы можете осуществлять поиск и фильтрацию информации в логах, чтобы быстро найти необходимую информацию. Это позволяет более эффективно анализировать данные и находить нужную информацию.

3. Раздел "Options File" в MySQL Workbench предназначен для работы с конфигурационным файлом сервера базы данных MySQL, который обычно называется `my.cnf`. Этот раздел позволяет просматривать, редактировать и сохранять настройки сервера, что позволяет вам точно настроить параметры работы базы данных. Вот основные аспекты, которые стоит учитывать при работе с опциями файла настроек:

- **Просмотр текущих настроек:** В разделе "Options File" вы можете просмотреть текущие настройки сервера MySQL, которые определены в файле конфигурации. Это включает различные параметры, такие как настройки памяти, буферов, соединений, безопасности и другие.
- **Редактирование параметров:** Вы можете изменять значения параметров в файле настроек прямо через MySQL Workbench. Это позволяет вам адаптировать настройки сервера под свои потребности и оптимизировать работу базы данных.
- **Сохранение изменений:** После внесения изменений в файл конфигурации через раздел "Options File", вы можете сохранить эти изменения, что применит их к работе сервера при следующем запуске или перезапуске.
- **Откат настроек:** В случае необходимости, вы также можете откатить сделанные изменения в файле настроек до предыдущего состояния, что обеспечивает гибкость в управлении параметрами сервера.

## **Раздел "Performance":**

1. Раздел "Dashboard" в MySQL Workbench представляет собой инструмент для мониторинга и анализа производительности вашего сервера базы данных MySQL. Эта панель предоставляет обзорную информацию о различных аспектах работы сервера, позволяя отслеживать ключевые метрики и выявлять потенциальные узкие места или проблемы производительности. Вот основные элементы и возможности, которые предлагает раздел "Dashboard":

- **Общий обзор:** На главной странице панели производительности вы можете увидеть сводные показатели о состоянии сервера, такие как загрузка процессора, использование памяти, количество активных соединений, нагрузку на сервер и другие ключевые метрики.
- **Графики и диаграммы:** Панель производительности предлагает графическое представление статистики и метрик производительности, что позволяет вам наглядно отслеживать динамику изменения показателей через графики и диаграммы.
- **Мониторинг ресурсов:** Вы можете отслеживать использование ресурсов сервера, таких как CPU, память, дисковое пространство, сетевой трафик и другие параметры, что помогает вам контролировать и оптимизировать работу сервера.
- **Анализ производительности запросов:** Один из важных аспектов панели производительности - анализ производительности SQL-запросов. Вы можете выявлять медленные запросы, оптимизировать их выполнение и улучшать общую производительность вашей базы данных.
- **Настройки оптимизации:** Панель производительности также может предлагать рекомендации и советы по оптимизации производительности сервера на основе данных мониторинга, что помогает вам принимать решения для улучшения работы сервера.

2. Раздел "Performance Reports" в MySQL Workbench предоставляет возможность генерировать и анализировать отчёты о производительности вашего сервера базы данных MySQL. Эти отчёты содержат важную информацию о нагрузке на сервер, выполнении запросов, использовании ресурсов и других ключевых метриках производительности. Давай рассмотрим более подробно основные аспекты и функциональность раздела "Performance Reports":

- **Типы отчётов:** В разделе "Performance Reports" вы можете получить доступ к различным типам отчётов, которые охватывают различные аспекты производительности сервера MySQL, такие как нагрузка CPU, сетевой трафик, производительность запросов, использование памяти и другие.
- **Анализ метрик производительности:** Отчёты о производительности позволяют вам анализировать ключевые метрики сервера, такие как время выполнения запросов, количество запросов, нагрузку на сервер, индексирование и другие показатели, что помогает вам оценить эффективность работы базы данных.
- **Графики и диаграммы:** Отчёты обычно содержат графическое представление данных в виде графиков и диаграмм, что помогает вам визуализировать информацию и быстро выявлять тренды или аномалии в работе сервера.
- **Рекомендации по оптимизации:** На основе данных отчётов о производительности, вы можете получить рекомендации и советы по оптимизации работы сервера MySQL, что помогает вам улучшить производительность, устранить проблемы и повысить эффективность базы данных.

3. Раздел "Performance Schema Setup" в MySQL Workbench предоставляет возможность настройки и оптимизации схемы производительности MySQL. Performance Schema - это инструмент MySQL, который предоставляет подробную информацию о работе сервера и выполнении запросов, что помогает вам выявить узкие места, оптимизировать

производительность и улучшить работу базы данных. Давай рассмотрим, как можно использовать этот раздел:

- **Активация Performance Schema:** Сначала необходимо активировать Performance Schema на вашем сервере MySQL. В разделе "Performance Schema Setup" вы можете выполнить необходимые действия для его включения и настройки.
- **Конфигурация параметров:** Performance Schema предоставляет множество параметров конфигурации, которые можно настроить для сбора нужной информации о производительности. В данном разделе вы можете определить, какие данные и метрики вы хотите отслеживать.
- **Отслеживание метрик:** После настройки схемы производительности, вы получите доступ к различным метрикам о производительности сервера, таким как использование ресурсов, время выполнения запросов, количество запросов, блокировки и другие.
- **Оптимизация производительности:** Анализ данных, собранных через Performance Schema, помогает вам выявлять узкие места производительности и оптимизировать работу сервера базы данных. Это может включать оптимизацию запросов, настройку индексов, улучшение конфигурации и другие меры.

Задание 2. Создать и настроить новую базу данных simpledb.

SCHEMAS

Filter objects

▼

simpledb

▼

Tables

▼

users

▼

Columns

◆

id

◆

name

◆

email

▶

Indexes

▶

Foreign Keys

▶

Triggers

▶

Views

▶

Stored Procedures

▶

Functions

▶

sys

MySQL Server

26LAGL

WAZDF

Connection Name

mysql-test

Host:

634918t

Socket:

/var/run

Port:

3306

Version:

8.0.36 (l

Compiled For:

Linux (

Configuration File:

unknown

Running Since:

Fri Feb 1

Available Server Features

Performance Schema:

●

On

Windows At

### Задание 3

Скопируйте запрос, соответствующий созданию этой таблицы и вставьте его в отчет по выполнению этой лабораторной работы.

```
CREATE TABLE `simplified`.`users` (  
  `id` INT NOT NULL AUTO_INCREMENT,  
  `name` VARCHAR(45) NOT NULL,  
  `email` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE INDEX `new_tablecol_UNIQUE` (`email` ASC) VISIBLE);
```



#### Задание 4

Добавьте несколько примеров-записей в созданную таблицу.

```
INSERT INTO `simplifiedb`.`users` (`name`, `email`) VALUES ('Paul',  
'StickQuicksand@gmail.com');  
INSERT INTO `simplifiedb`.`users` (`name`, `email`) VALUES ('Marina',  
'ScriptMultiBlue@gmail.com');  
INSERT INTO `simplifiedb`.`users` (`name`, `email`) VALUES ('Katya',  
'PandaSquidSeal@gmail.com');
```

После этого обновите одно или несколько полей (например, name и/или email), нажмите Apply и сохраните полученный SQL-запрос в отчете.

Какой SQL-запрос при этом выполнится?

```
UPDATE `simplifiedb`.`users` SET `name` = 'Jonh' WHERE (`id` = '3');
```

### Задание 5.

Дополните таблицу users так, чтобы получилась таблица со следующими полями и параметрами: 1. id int pk, not null 2. name varchar(50) 3. email varchar(45) 4. gender ENUM('M', 'F') 5. bday Date 6. postal\_code varchar(10) 7. rating float 8. created TIMESTAMP CURRENT\_TIMESTAMP()

```
ALTER TABLE `simplifiedb`.`users`  
  
ADD COLUMN `gender` ENUM('M', 'F') NULL AFTER `email`,  
  
ADD COLUMN `bday` DATE NULL AFTER `gender`,  
  
ADD COLUMN `postal_code` VARCHAR(10) NULL AFTER `bday`,  
  
ADD COLUMN `rating` FLOAT NULL AFTER `postal_code`,  
  
ADD COLUMN `created` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP()  
AFTER `rating`,  
  
CHANGE COLUMN `name` `name` VARCHAR(50) NOT NULL ;
```

1. Тип данных TIMESTAMP: Поле типа TIMESTAMP в MySQL используется для хранения времени (даты и времени). Он обычно используется для отслеживания времени создания или обновления записей в таблице.

2. Значение по умолчанию CURRENT\_TIMESTAMP(): Использование CURRENT\_TIMESTAMP() в качестве значения по умолчанию для поля TIMESTAMP означает, что если при вставке новой записи в таблицу не указано значение для поля created, система автоматически установит текущее время и дату в это поле.

Все поля, кроме первых трех, могут иметь значение NULL.

**Задание 6.**

Дополните таблицу, добавив данные двумя способами:

- с помощью внесения данных вручную (как это было сделано ранее);
- с помощью выполнения SQL-запросов ниже;

```
INSERT INTO `simplifiedb`.`users` (`name`, `email`, `postal_code`, `gender`, `bday`, `rating`)  
VALUES ('Ekaterina', 'ekaterina.petrova@outlook.com', '145789', 'f', '2000-02-11', '1.123');  
INSERT INTO `simplifiedb`.`users` (`name`, `email`, `postal_code`, `gender`, `bday`, `rating`)  
VALUES ('Paul', 'paul@superpochta.ru', '123789', 'm', '1998-08-12', '1');
```

**Вручную:**

```
UPDATE `simplifiedb`.`users` SET `gender` = 'M', `postal_code` = '123456', `rating` = '1'  
WHERE (`id` = '1');
```

```
UPDATE `simplifiedb`.`users` SET `gender` = 'F', `postal_code` = '654321', `rating` = '2' WHERE  
(`id` = '2');
```

```
UPDATE `simplifiedb`.`users` SET `gender` = 'M', `postal_code` = '234156', `rating` = '3'  
WHERE (`id` = '3');
```

	id	name	email	gender	bday	postal_code	rating	created
	1	Paul	StickQuicksand@gmail.com	M	NULL	123456	1	2024-02-21 14:37:46
	2	Marina	ScriptMultiBlue@gmail.com	F	NULL	654321	2	2024-02-21 14:37:46
▶	3	Jonh	PandaSquidSeal@gmail.com	M	NULL	234156	3	2024-02-21 14:37:46
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

**С помощью запросов:**

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	id	name	email	gender	bday	postal_code	rating	created
▶	1	Paul	StickQuicksand@gmail.com	M	NULL	123456	1	2024-02-21 14:37:46
	2	Marina	ScriptMultiBlue@gmail.com	F	NULL	654321	2	2024-02-21 14:37:46
	3	Jonh	PandaSquidSeal@gmail.com	M	NULL	234156	3	2024-02-21 14:37:46
	4	Ekaterina	ekaterina.petrova@outlook.com	F	2000-02-11	145789	1.123	2024-02-21 14:55:36
	5	Paul	paul@superpochta.ru	M	1998-08-12	123789	1	2024-02-21 14:55:37
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Query 1 | SQL File 2\* x users | Limit to 1000 rows

```
1 • INSERT INTO `simplifiedb`.`users` (`name`, `email`, `postal_code`, `gender`, `bday`,  
2   `rating`) VALUES ('Ekaterina', 'ekaterina.petrova@outlook.com', '145789', 'f',  
3   '2000-02-11', '1.123');  
4 • INSERT INTO `simplifiedb`.`users` (`name`, `email`, `postal_code`, `gender`, `bday`,  
5   `rating`) VALUES ('Paul', 'paul@superpochta.ru', '123789', 'm', '1998-08-12', '1');  
6
```

### Задание 7.

С помощью кнопки “Export recordset to external file” и получите файл с SQL-запросами.

/\*

-- Query: SELECT \* FROM simplifiedb.users

LIMIT 0, 1000

-- Date: 2024-02-21 17:59

\*/

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (1,'Paul','StickQuicksand@gmail.com','M',NULL,'123456',1,'2024-02-21 14:37:46');
```

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (2,'Marina','ScriptMultiBlue@gmail.com','F',NULL,'654321',2,'2024-02-21 14:37:46');
```

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (3,'Jonh','PandaSquidSeal@gmail.com','M',NULL,'234156',3,'2024-02-21 14:37:46');
```

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (4,'Ekaterina','ekaterina.petrova@outlook.com','F','2000-02-11','145789',1.123,'2024-  
02-21 14:55:36');
```

```
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal_code`,`rating`,`created`)  
VALUES (5,'Paul','paul@superpochta.ru','M','1998-08-12','123789',1,'2024-02-21 14:55:37');
```

### Задание 8.

Создайте еще одну таблицу с названием resume со следующей структурой: - resumeid, INT, PK, NN, AI - userid, INT, NN - title, VARCHAR(100), NN - skills, TEXT - created, TIMESTAMP, Default / Expression: CURRENT\_TIMESTAMP().

При конструировании внизу во вкладке Foreign Keys определите так называемый внешний ключ (foreign key), который будет определять связь между текущей таблицей resume и уже созданной таблицей user.

Введите в таблицу слева в столбец Foreign Key (внешний ключ): userid и определите таблицу, где они будут находиться: simpledb.users

В таблицу Foreign key details 'userid' щелкните мышкой рядом с полем userid так, чтобы оно было выделено и определите столбец-источник для значений - id.

Последний шаг: определить действия при операциях On Update и On Delete: для обоих выберите - Cascade.

```
CREATE TABLE `simpledb`.`resume` (  
  `resumeid` INT NOT NULL AUTO_INCREMENT,  
  `userid` INT NOT NULL,  
  `title` VARCHAR(100) NOT NULL,  
  `skills` MEDIUMTEXT NULL,  
  `created` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP(),  
  PRIMARY KEY (`resumeid`),  
  INDEX `userid_idx` (`userid` ASC) VISIBLE,  
  CONSTRAINT `userid`  
    FOREIGN KEY (`userid`)  
    REFERENCES `simpledb`.`users` (`id`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE);
```

**Опишите как будет вести себя СУБД при удалении связанных записей из таблиц users и resume.**

При удалении связанных записей из таблиц users и resume, поведение СУБД будет следующим:

1. При удалении записи из таблицы users:

- Если у пользователя удалены все данные из таблицы resume, то удаление пользователя из таблицы users пройдет без проблем.

- При наличии связанных записей в таблице resume, из-за внешнего ключа с опцией ON DELETE CASCADE, при удалении пользователя из таблицы users, все связанные записи из таблицы resume, связанные с этим пользователем (по user\_id), будут также удалены автоматически (CASCADE).

2. При удалении записи из таблицы resume:

- Удаление записи из таблицы resume не повлияет на данные в таблице users, так как удаление из дочерней таблицы (resume) не требует каскадного удаления из родительской таблицы (users).

## Задание 9.

Наполните вторую таблицу данными так, чтобы в ней была информация хотя бы о нескольких резюме, связанных с уже существующими пользователями из таблицы users.

Result Grid					
		Filter Rows:		Edit: 	
				Export/Im	
	resumeid	userid	title	skills	created
	NULL	1	Web Developer	JavaScript, React, Node.js	NULL
	NULL	2	Data Analyst	SQL, Python, Data Visualization	NULL
▶*	NULL	NULL	NULL	NULL	

/\*

-- Query: SELECT \* FROM simplifiedb.resume

LIMIT 0, 1000

-- Date: 2024-02-21 18:27

\*/

INSERT INTO `` (`resumeid`,`userid`,`title`,`skills`,`created`) VALUES (1,1,'Web Developer','JavaScript, React, Node.js','2024-02-21 15:25:57');

INSERT INTO `` (`resumeid`,`userid`,`title`,`skills`,`created`) VALUES (2,2,'Data Analyst','SQL, Python, Data Visualization','2024-02-21 15:25:57');

**Подумайте и напишите в отчете, сколько резюме может быть у одного пользователя (минимум и максимум)?**


По умолчанию, если требований ограничений нет, то каждый пользователь может иметь любое количество резюме в таблице resume, начиная от 0 и без ограничения сверху. Схема и логика взаимосвязи в базе данных будет определять минимальное и максимальное количество резюме, которые могут быть связаны с одним пользователем.

**Попробуйте добавить в таблицу resume строчку с userid несуществующего пользователя (такого пользователя id которого нет в таблице users).**

При выполнении подобной операции будет выдано сообщение об ошибке, указывающее на нарушение целостности данных из-за отсутствия соответствующего user\_id в таблице users. Это одно из важных требований для поддержания целостности данных в базе данных при использовании внешних ключей для связей между таблицами.

## Applying SQL script to the database

The following tasks will now be executed. Please monitor the execution.  
Press Show Logs to see the execution logs.

 Execute SQL Statements

Error: There was an error while applying the SQL script to the database.

### Message Log

```
Executing:
INSERT INTO `simplifiedb`.`resume` (`userid`, `title`, `skills`) VALUES (6, 'Graphic Designer', 'Adobe
Photoshop, UI/UX Design');

Operation failed: There was an error while applying the SQL script to the database.
ERROR 1452: 1452: Cannot add or update a child row: a foreign key constraint fails
(`simplifiedb`.`resume`, CONSTRAINT `userid` FOREIGN KEY (`userid`) REFERENCES `users` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE)
SQL Statement:
INSERT INTO `simplifiedb`.`resume` (`userid`, `title`, `skills`) VALUES (6, 'Graphic Designer', 'Adobe
Photoshop, UI/UX Design')
```



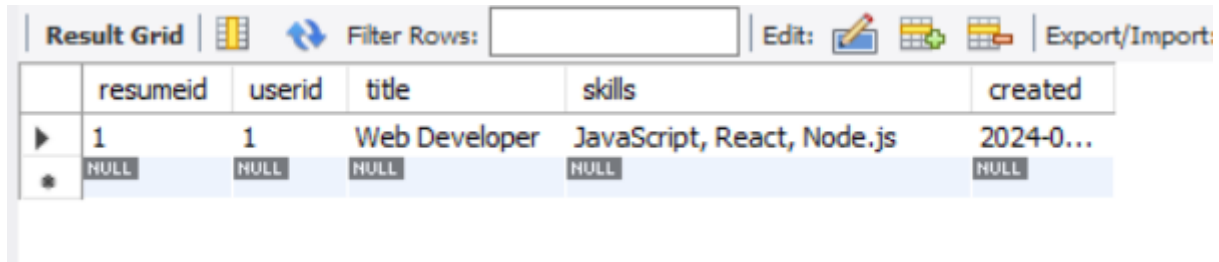
### Задание 10.

Удалите одного или двух таких пользователей, что для них существуют записи в таблице resume.

```
DELETE FROM `simplifiedb`.`users` WHERE (`id` = '2');
```

**Что произойдет со связанными сущностями в таблице resume?**

Соответствующее резюме будет удалено.

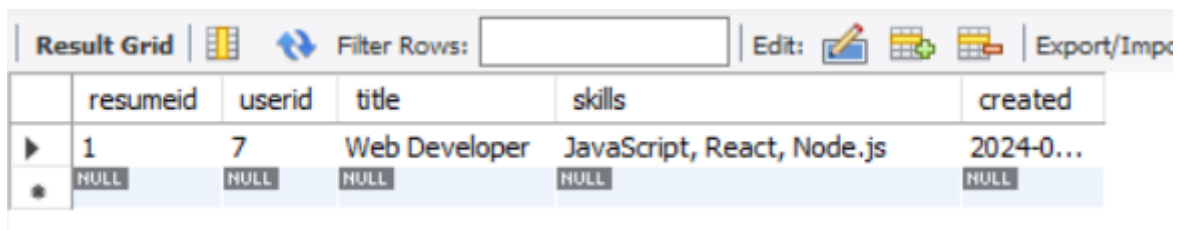


	resumeid	userid	title	skills	created
▶	1	1	Web Developer	JavaScript, React, Node.js	2024-0...
✱	NULL	NULL	NULL	NULL	NULL

**Что произойдет, если в таблице users будет изменен id какого-то существующего пользователя.**

```
UPDATE `simplifiedb`.`users` SET `id` = '7' WHERE (`id` = '1');
```

Произойдет автоматическое обновление.



	resumeid	userid	title	skills	created
▶	1	7	Web Developer	JavaScript, React, Node.js	2024-0...
✱	NULL	NULL	NULL	NULL	NULL