**Лабораторная работа 1**

*Задание 1*

**Раздел “Management”:**

1. Раздел “Client Connections”.

Данный раздел отображает список активных и спящих клиентских подключений MySQL и добавляет возможность прерывать инструкции и соединения, а также просматривать дополнительные сведения о соединении и атрибуты.

2. Раздел “Users and Privileges”.

Данный раздел содержит список всех пользователей и привилегий, которые относятся к активному экземпляру сервера MySQL. На этой вкладке вы можете добавлять учетные записи пользователей и управлять ими, изменять привилегии и срок действия паролей.

Вкладка содержит несколько областей задач:

- User Accounts (перечисляются все учетные записи пользователей, связанные с активным подключением к MySQL);

- Login Tab (предоставляет следующую информацию, относящуюся к выбранной учетной записи пользователя: Login Name, Authentication Type, Limit to Hosts Matching, Password and Confirm Password);

- Account Limits Tab (определяет следующие ограничения для выбранной учетной записи пользователя:

Max. Queries: количество запросов, которые учетная запись может выполнить в течение одного часа;

Max. Updates: количество обновлений, которые учетная запись может выполнить в течение одного часа;

Max. Connections: количество раз, когда учетная запись может подключиться к серверу в течение часа;

Concurrent Connections: количество одновременных подключений к серверу, которые может иметь учетная запись.)

- Administrative Roles (позволяет назначить роль для каждого аккаунта, и включает в себя следующие роли: DBA, MaintenanceAdmin, ProcessAdmin, UserAdmin, SecurityAdmin, MonitorAdmin, DBManager, DBDesigner, ReplicationAdmin, BackupAdmin, Custom);

- Schema Privileges Tab (Позволяет уточнить способ назначения прав доступа к одной или нескольким схемам с помощью учетной записи пользователя)

3. Раздел “Status and System Variables”.

Содержит список полного набора серверных переменных для активного подключения к MySQL. Возможности:

- Сохранение системных переменных;

- Пользовательская группировка переменных;

4. Разделы “Data Export” и “Data Import/Restore”.

Данные разделы предназначены для выгрузки, загрузки и восстановления данных.

Методы экспорта и импорта данных:

- Мастер экспорта и импорта табличных данных (поддерживает операции импорта и экспорта с использованием файлов CSV и JSON и включает в себя несколько параметров конфигурации);

- Мастер экспорта и импорта данных SQL (позволяет экспортировать или импортировать SQL, сгенерированный из MySQL Workbench или с помощью команды mysqldump);

- Экспорт и импорт данных результатов (позволяет работать с результирующим набором, импортировать и экспортировать данные);

**Раздел “Instance”:**

1. Раздел “Startup/Shutdown”.

Данный раздел позволяет выполнять следующие действия по управлению службами:

- Просмотр журнала сообщений о запуске;

- Запуск и завершение работы экземпляра MySQL;

- Просмотр текущего состояние экземпляра MySQL;

2. Раздел “Server Logs”.

Данный раздел является журналом, в который записываются происходящие активности, к каждой из которых соответствует некоторый тип:

- Error log (проблемы, возникшие при запуске, выполнении или остановке mysqld);

- General query log (установленные клиентские соединения и выражения);

- Binary log (инструкции для изменения данных, которые также используются для репликации);

- Relay log (изменения данных, полученные с сервера-источника репликации);

- Slow query log (запросы, на обработку которых ушло больше секунд, чем на обработку long\_query\_time);

- DDL log (metadata log) (операции с метаданными, выполняемые операторами DDL);

3. Раздел “Options File”.

Данный раздел предназначен для предоставления доступа к опциональным файлам. Они предоставляют удобный способ указать часто используемые параметры, чтобы их не нужно было вводить в командной строке при каждом запуске программы.

**Раздел “Performance”**

1. Раздел “Dashboard”.

Данный раздел включает три панели, которые показывают информацию о скорости загрузки, выгрузки и эффективности процессов, происходящих в MySQL.

- Network Status (показывает скорость загрузки и выгрузки на нашем подключении MySQL);

- MySQL Status (демонстрирует статистику, в которой указана производительность и эффективность выполнения требуемых процессов);

- InnoDB Status (показывает пул буфера InnoDB и активности диска, генерируемой механизмом хранения InnoDB);

2. Раздел “Performance Reports”.

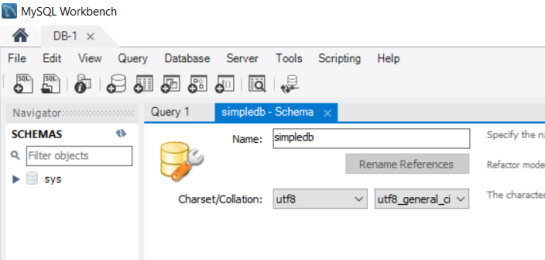
В данном разделе содержится большое количество отчётов о работе сервера MySQL. Данные отчёты помогают анализировать производительность баз данных MySQL, а также просматривать различные статистические данные, время ожидания и многое другое.

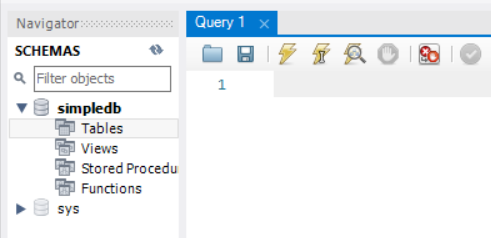
3. Раздел “Performance Schema Setup”.

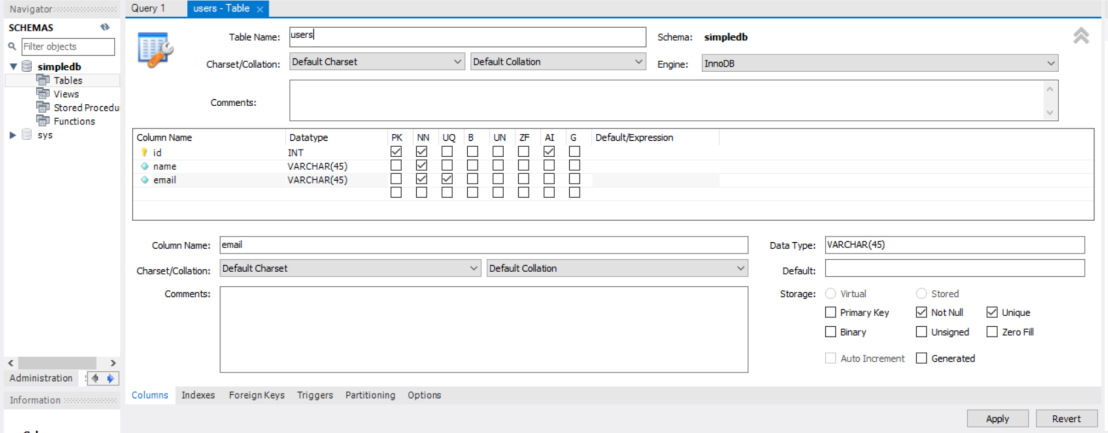
Данный раздел является графическим интерфейсом для конфигурирования и тонкой настройки инструментария схемы производительности.

После выбора на первом окне и нажатии “Show Advanced” открываются дополнительные разделы для более тщательной настройки: Easy Setup, Introduction, Instruments, Consumers, Actors & Objects, Threads, Options.

*Задание 2*







*Задание 3*

CREATE TABLE `simpledb`.`users` (

`id` INT NOT NULL AUTO\_INCREMENT,

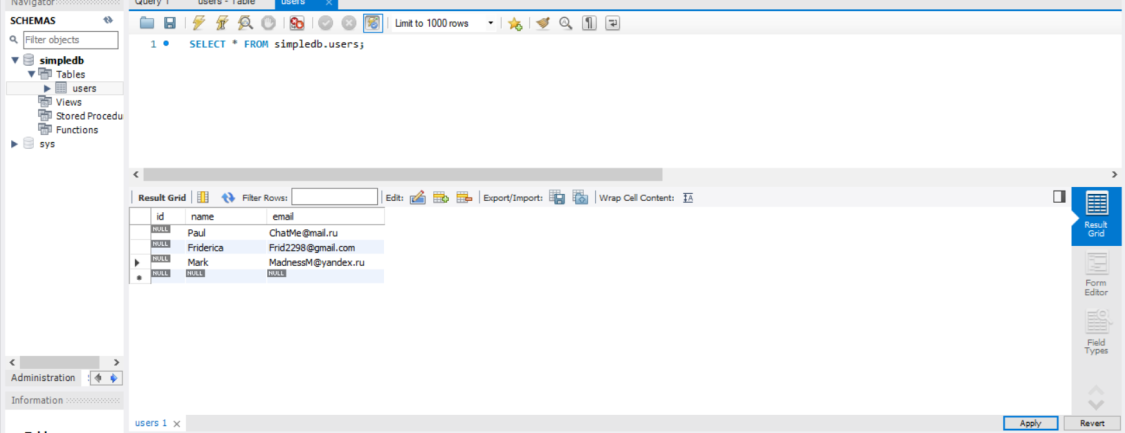
`name` VARCHAR(45) NOT NULL,

`email` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) VISIBLE);

*Задание 4*

**

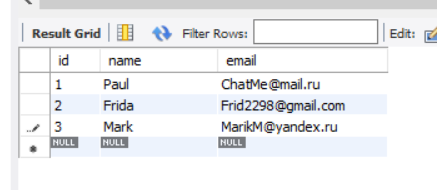
Заполнение таблицы:

INSERT INTO `simpledb`.`users` (`name`, `email`) VALUES ('Paul', 'ChatMe@mail.ru');

INSERT INTO `simpledb`.`users` (`name`, `email`) VALUES ('Friderica', 'Frid2298@gmail.com');

INSERT INTO `simpledb`.`users` (`name`, `email`) VALUES ('Mark', 'MadnessM@yandex.ru');

Изменение полей таблицы:



UPDATE `simpledb`.`users` SET `name` = 'Frida' WHERE (`id` = '2');

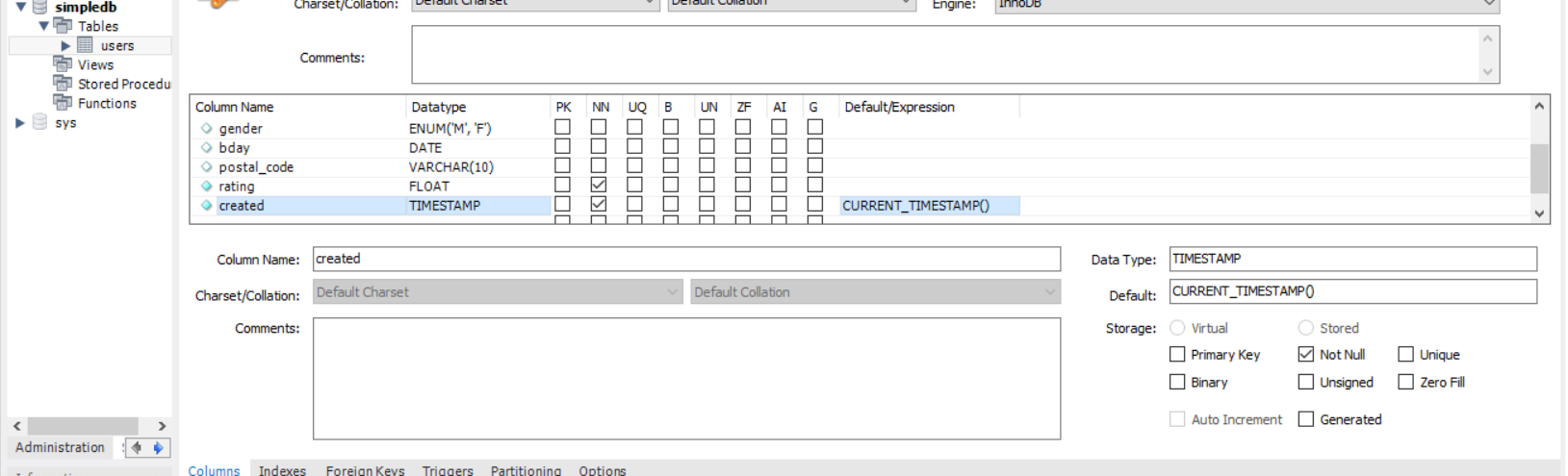
UPDATE `simpledb`.`users` SET `email` = 'MarikM@yandex.ru' WHERE (`id` = '3');

*Задание 5*

TIMESTAMP() получает и показывает время обновления страницы.

При установке значения CURRENT\_TIMESTAMP() по умолчанию, не будет происходить автоматического обновления при изменении других столбцов.

Иметь значение NULL, могут иметь такие поля, как: gender, bday и postal\_code. Поле rating должно иметь значение NOT NULL, т.к. какое-либо ранжирование ведёт не сам пользователь, а какой-либо алгоритм. А также поле TIMESTAMP должно иметь значение NOTNULL, т.к. оно имеет значение по умолчанию CURRENT\_TIMESTAMP().



Дополнение таблицы:

ALTER TABLE `simpledb`.`users`

ADD COLUMN `gender` ENUM('M', 'F') NULL AFTER `email`,

ADD COLUMN `bday` DATE NULL AFTER `gender`,

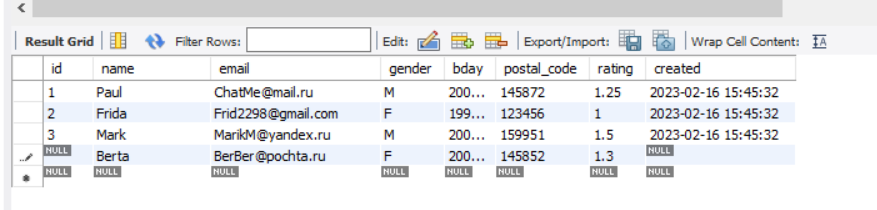
ADD COLUMN `postal\_code` VARCHAR(10) NULL AFTER `bday`,

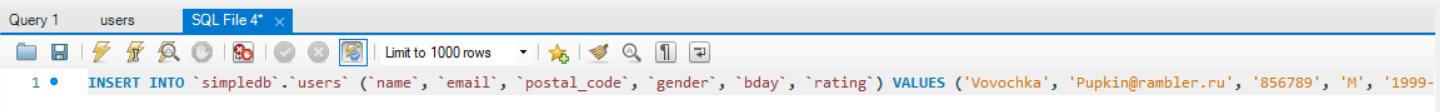
ADD COLUMN `rating` FLOAT NOT NULL AFTER `postal\_code`,

ADD COLUMN `created` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP() AFTER `rating`,

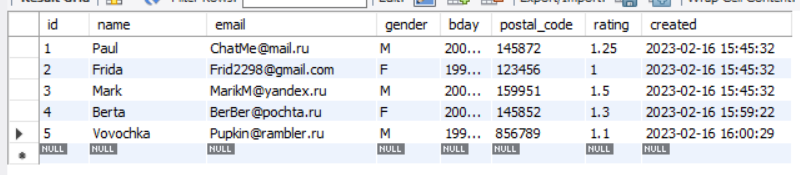
CHANGE COLUMN `name` `name` VARCHAR(50) NOT NULL ;

*Задание 6*



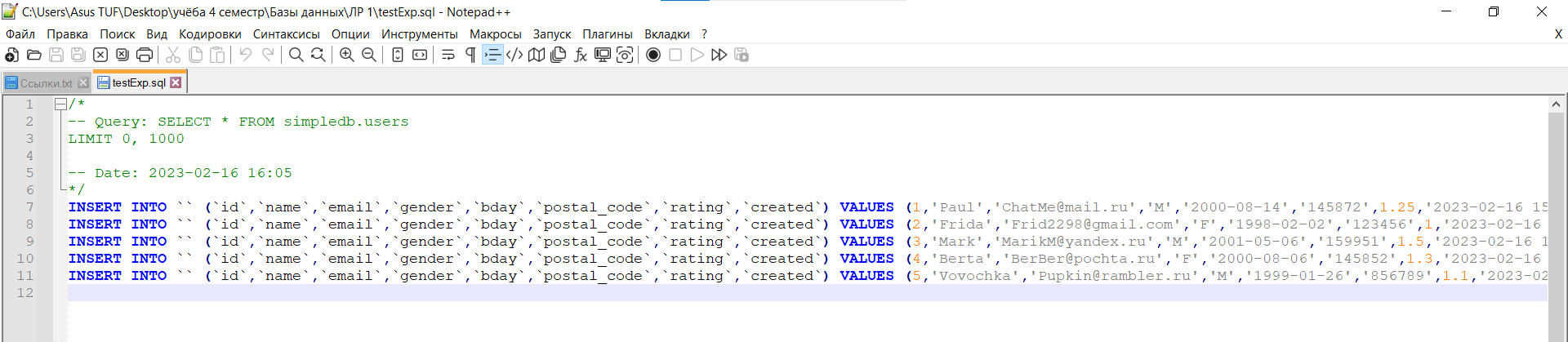


INSERT INTO `simpledb`.`users` (`name`, `email`, `postal\_code`, `gender`, `bday`, `rating`) VALUES ('Vovochka', 'Pupkin@rambler.ru', '856789', 'M', '1999-01-26', '1.1');



*Задание 7*





/\*

-- Query: SELECT \* FROM simpledb.users

LIMIT 0, 1000

-- Date: 2023-02-16 16:05

\*/

INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal\_code`,`rating`,`created`) VALUES (1,'Paul','ChatMe@mail.ru','M','2000-08-14','145872',1.25,'2023-02-16 15:45:32');

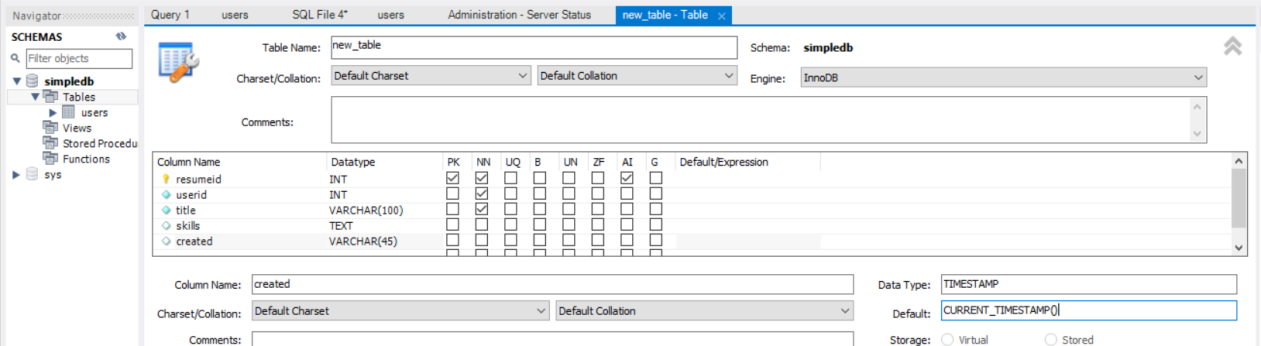
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal\_code`,`rating`,`created`) VALUES (2,'Frida','Frid2298@gmail.com','F','1998-02-02','123456',1,'2023-02-16 15:45:32');

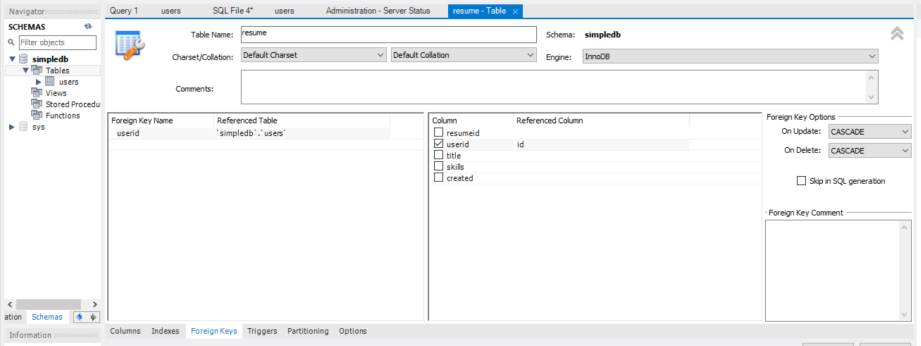
INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal\_code`,`rating`,`created`) VALUES (3,'Mark','MarikM@yandex.ru','M','2001-05-06','159951',1.5,'2023-02-16 15:45:32');

INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal\_code`,`rating`,`created`) VALUES (4,'Berta','BerBer@pochta.ru','F','2000-08-06','145852',1.3,'2023-02-16 15:59:22');

INSERT INTO `` (`id`,`name`,`email`,`gender`,`bday`,`postal\_code`,`rating`,`created`) VALUES (5,'Vovochka','Pupkin@rambler.ru','M','1999-01-26','856789',1.1,'2023-02-16 16:00:29');

*Задание 8*





CREATE TABLE `simpledb`.`resume` (

`resumeid` INT NOT NULL AUTO\_INCREMENT,

`userid` INT NOT NULL,

`title` VARCHAR(100) NOT NULL,

`skills` TEXT NULL,

`created` TIMESTAMP NULL DEFAULT CURRENT\_TIMESTAMP(),

PRIMARY KEY (`resumeid`),

INDEX `userid\_idx` (`userid` ASC) VISIBLE,

CONSTRAINT `userid`

FOREIGN KEY (`userid`)

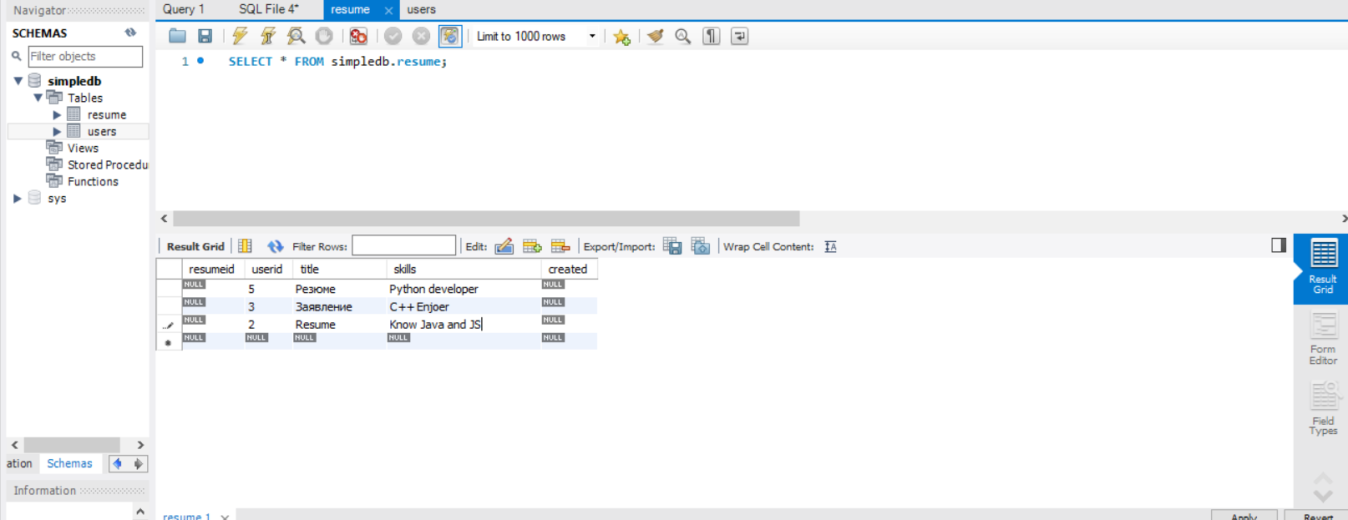
REFERENCES `simpledb`.`users` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE);

Мы можем спокойно менять и удалять любые записи в таблице ‘resume’, однако при удалении пользователя с некоторым id из таблицы ‘users’, будут удалены все сущности связанные с этим id из таблицы ‘resume’.

*Задание 9*

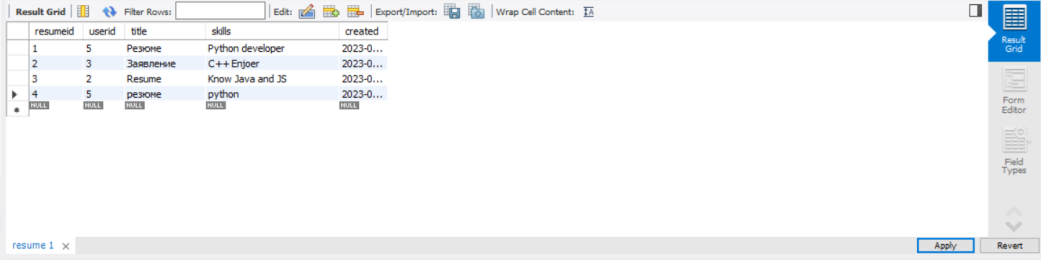


INSERT INTO `simpledb`.`resume` (`userid`, `title`, `skills`) VALUES ('5', 'Резюме', 'Python developer');

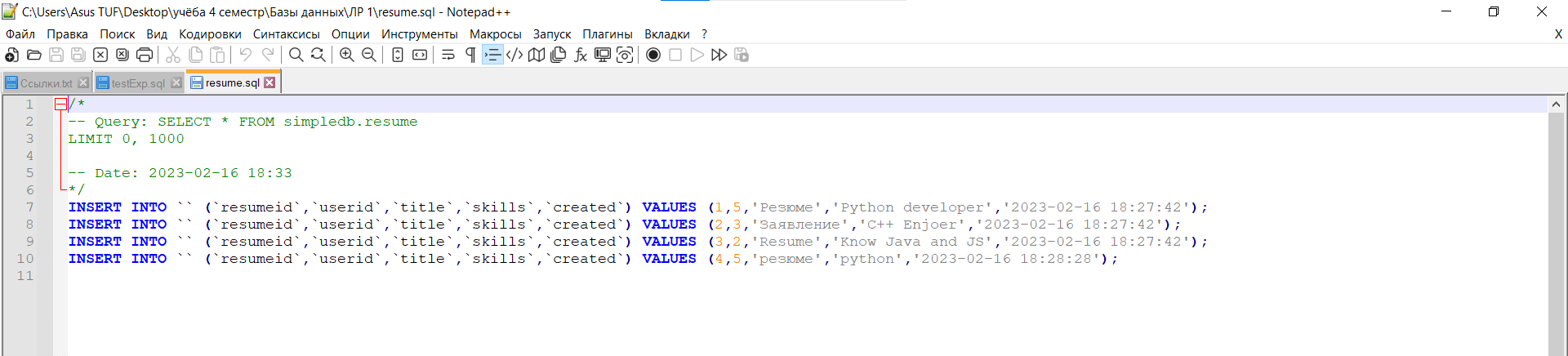
INSERT INTO `simpledb`.`resume` (`userid`, `title`, `skills`) VALUES ('3', 'Заявление', 'С++ Enjoer');

INSERT INTO `simpledb`.`resume` (`userid`, `title`, `skills`) VALUES ('2', 'Resume', 'Know Java and JS');

Каждый пользователь может иметь любое количество резюме, от 0 и до бесконечности, т.к. при написании нового резюме, id пользователя (userid) остаётся прежним, а новое резюме получает новый id (столбец resumeid).







/\*

-- Query: SELECT \* FROM simpledb.resume

LIMIT 0, 1000

-- Date: 2023-02-16 18:33

\*/

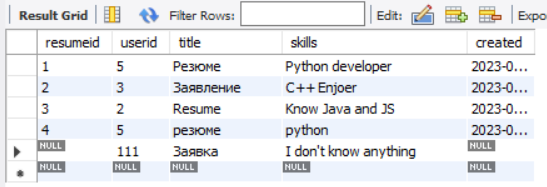
INSERT INTO `` (`resumeid`,`userid`,`title`,`skills`,`created`) VALUES (1,5,'Резюме','Python developer','2023-02-16 18:27:42');

INSERT INTO `` (`resumeid`,`userid`,`title`,`skills`,`created`) VALUES (2,3,'Заявление','С++ Enjoer','2023-02-16 18:27:42');

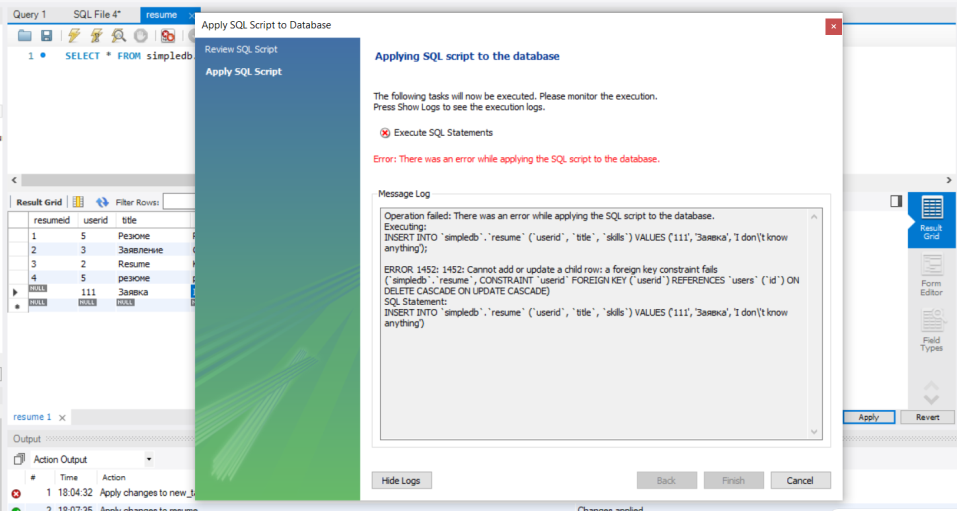
INSERT INTO `` (`resumeid`,`userid`,`title`,`skills`,`created`) VALUES (3,2,'Resume','Know Java and JS','2023-02-16 18:27:42');

INSERT INTO `` (`resumeid`,`userid`,`title`,`skills`,`created`) VALUES (4,5,'резюме','python','2023-02-16 18:28:28');

Попытка добавить несуществующего пользователя:

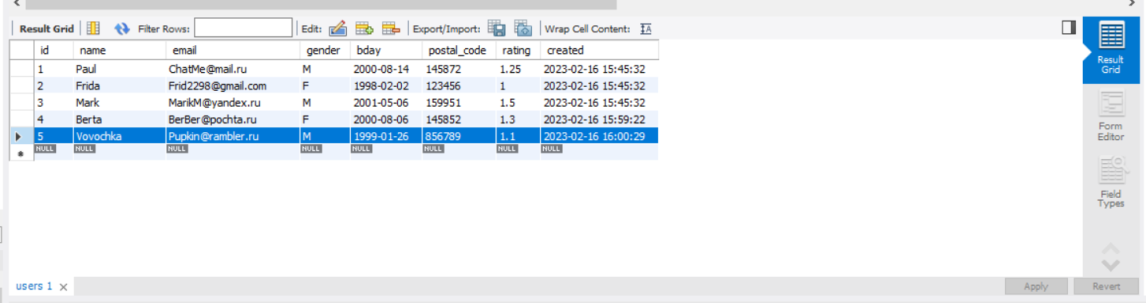


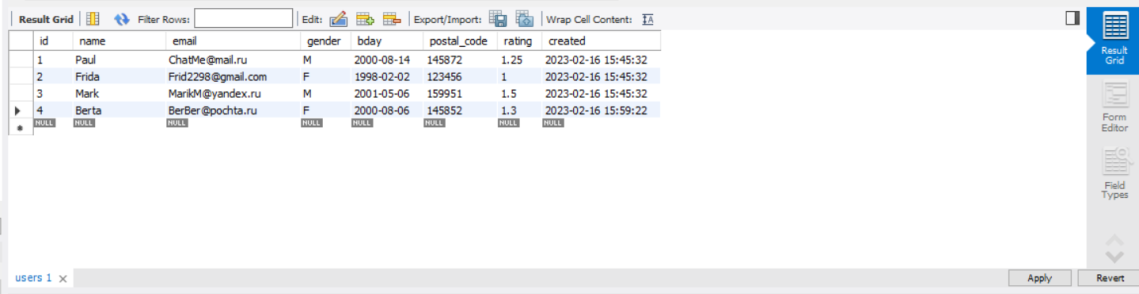
Выдаётся ошибка о не возможности добавления или обновления дочерней строки из-за не выполнения ограничений внешнего ключа.



*Задание 10*

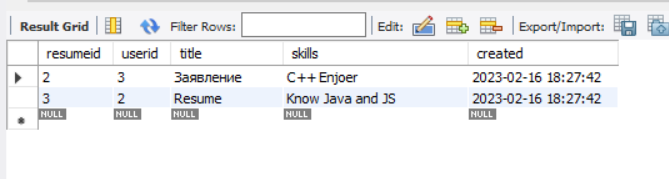
Удаление пользователя Vovochka:





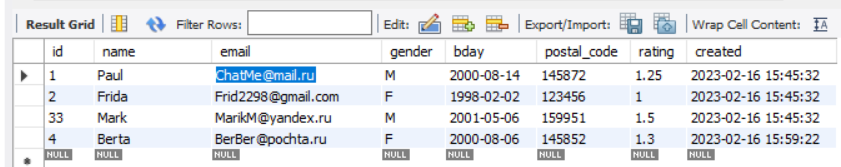
DELETE FROM `simpledb`.`users` WHERE (`id` = '5');

При удалении пользователи из таблицы ‘users’, все связанные с его id сущности в таблице ‘resume’ также удаляются.



Если поменять id у существующего пользователя в таблице ‘users’, то в таблице ‘resume’ у связанных сущностей, также автоматически поменяется id.

Меняем id пользователя Mark:



UPDATE `simpledb`.`users` SET `id` = '33' WHERE (`id` = '3');

Видим, что userid поменялся в таблице ‘resume’, после изменений внесённых в таблицу ‘users’:

