**Базы данных**

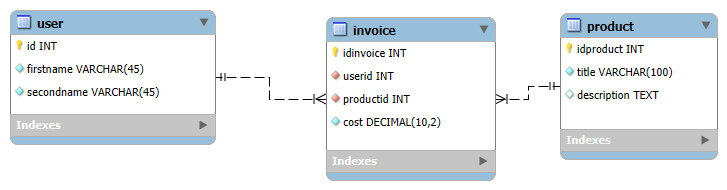
**Лабораторная работа 2**

**Схема данных. EER-диаграмма**

**Отчет**

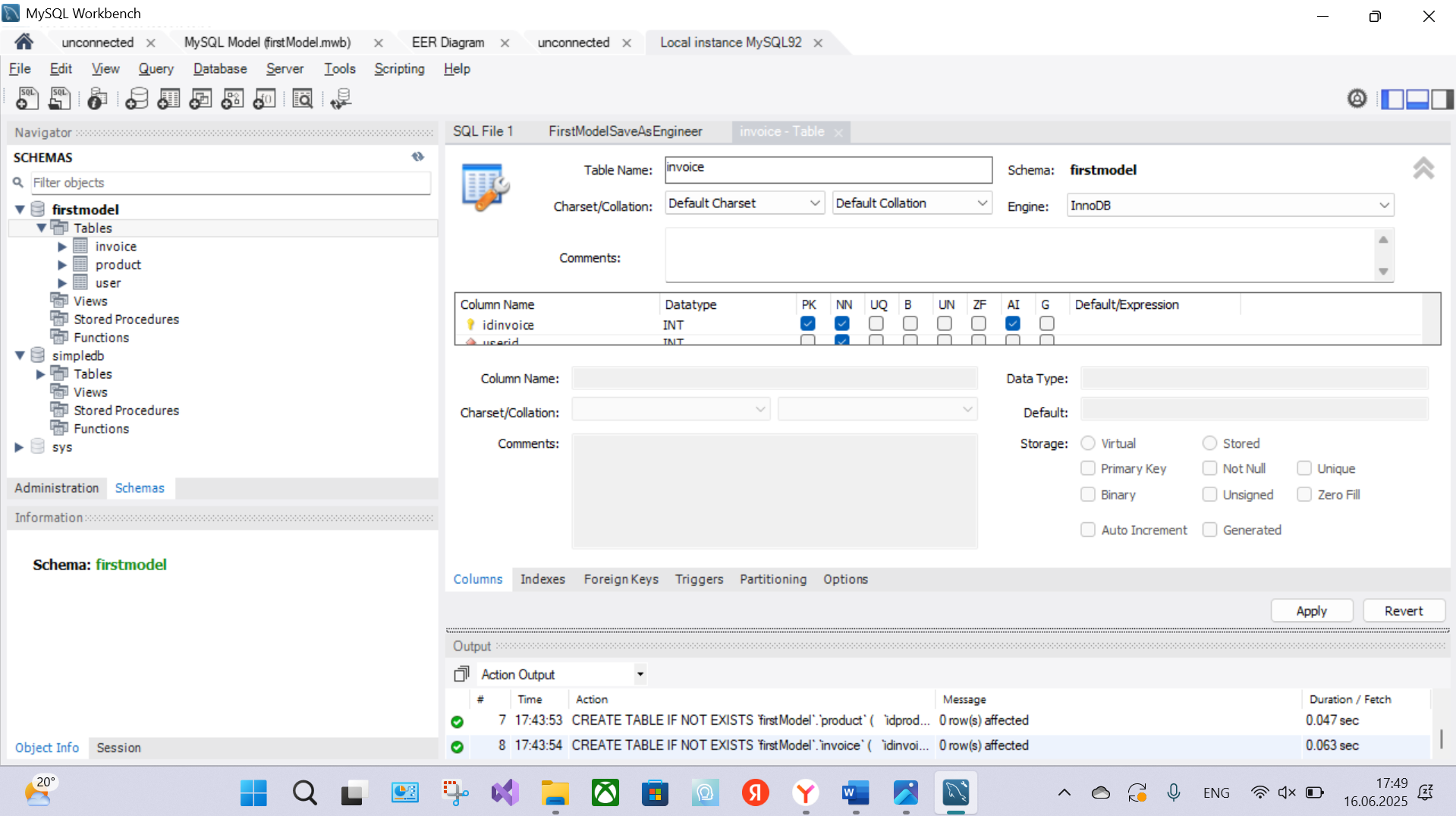
**Задание 1**. Повторите действия, демонстрируемые в ролике и создайте диаграмму (модель), созданную во второй половине ролика. Экспортируйте модель в виде изображения, экспортируйте модель в виде SQL-скрипта.

Cхема в виде изображения:



Скопированный запрос, соответствующий созданию этой базы данных:

<https://gist.github.com/Olljux/d66a19d5331ef36f60341e52372d4fd0>



Фрагмент запроса, касающийся создания и настройки таблицы invoice:

CREATE TABLE IF NOT EXISTS `firstModel`.`invoice` (

`idinvoice` INT NOT NULL AUTO\_INCREMENT,

`userid` INT NOT NULL,

`productid` INT NOT NULL,

`cost` DECIMAL(10,2) NOT NULL,

PRIMARY KEY (`idinvoice`),

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

INDEX `prod\_idx` (`productid` ASC) VISIBLE,

CONSTRAINT `user`

FOREIGN KEY (`userid`)

REFERENCES `firstModel`.`user` (`id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `prod`

FOREIGN KEY (`productid`)

REFERENCES `firstModel`.`product` (`idproduct`)

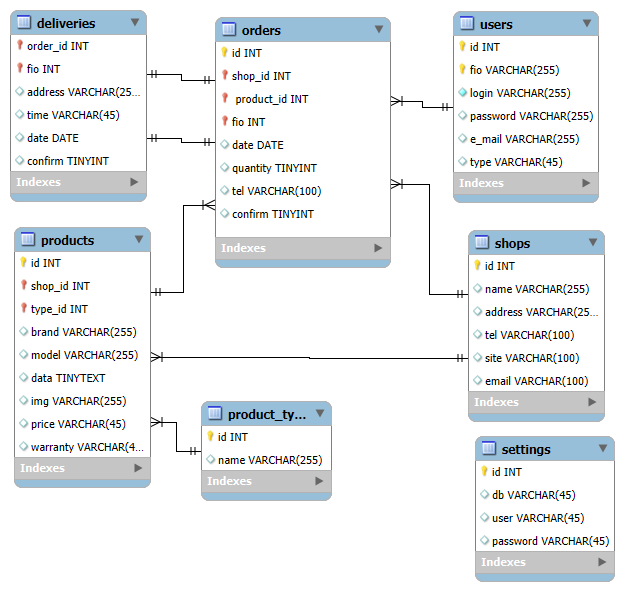
ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB

**Задание 2.** Создайте собственную EER-диаграмму и спроектируйте БД с параметрами на основе текста, опубликованного по ссылке: <https://habr.com/ru/post/175985/>

Cхема в виде изображения:



Скопированный запрос, соответствующий созданию этой базы данных:

<https://gist.github.com/Olljux/090d366590f3d921423f69ea1020e12f>

Фрагмент запроса, касающийся создания и настройки таблицы *Orders*:

CREATE TABLE IF NOT EXISTS `mydb`.`orders` (

`id` INT NOT NULL AUTO\_INCREMENT,

`shop\_id` INT NOT NULL,

` product\_id` INT NOT NULL,

`fio` INT NOT NULL,

`date` DATE NULL,

`quantity` TINYINT NULL,

`tel` VARCHAR(100) NULL,

`confirm` TINYINT NULL,

PRIMARY KEY (`id`, `shop\_id`, ` product\_id`, `fio`),

UNIQUE INDEX `id\_UNIQUE` (`id` ASC) VISIBLE,

INDEX `orders\_to\_users\_fio\_idx` (`fio` ASC) VISIBLE,

INDEX `orders\_to\_products\_idx` (` product\_id` ASC) VISIBLE,

INDEX `orders\_to\_shops\_idx` (`shop\_id` ASC) VISIBLE,

CONSTRAINT `orders\_to\_users`

FOREIGN KEY (`fio`)

REFERENCES `mydb`.`users` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `orders\_to\_products`

FOREIGN KEY (` product\_id`)

REFERENCES `mydb`.`products` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `orders\_to\_shops`

FOREIGN KEY (`shop\_id`)

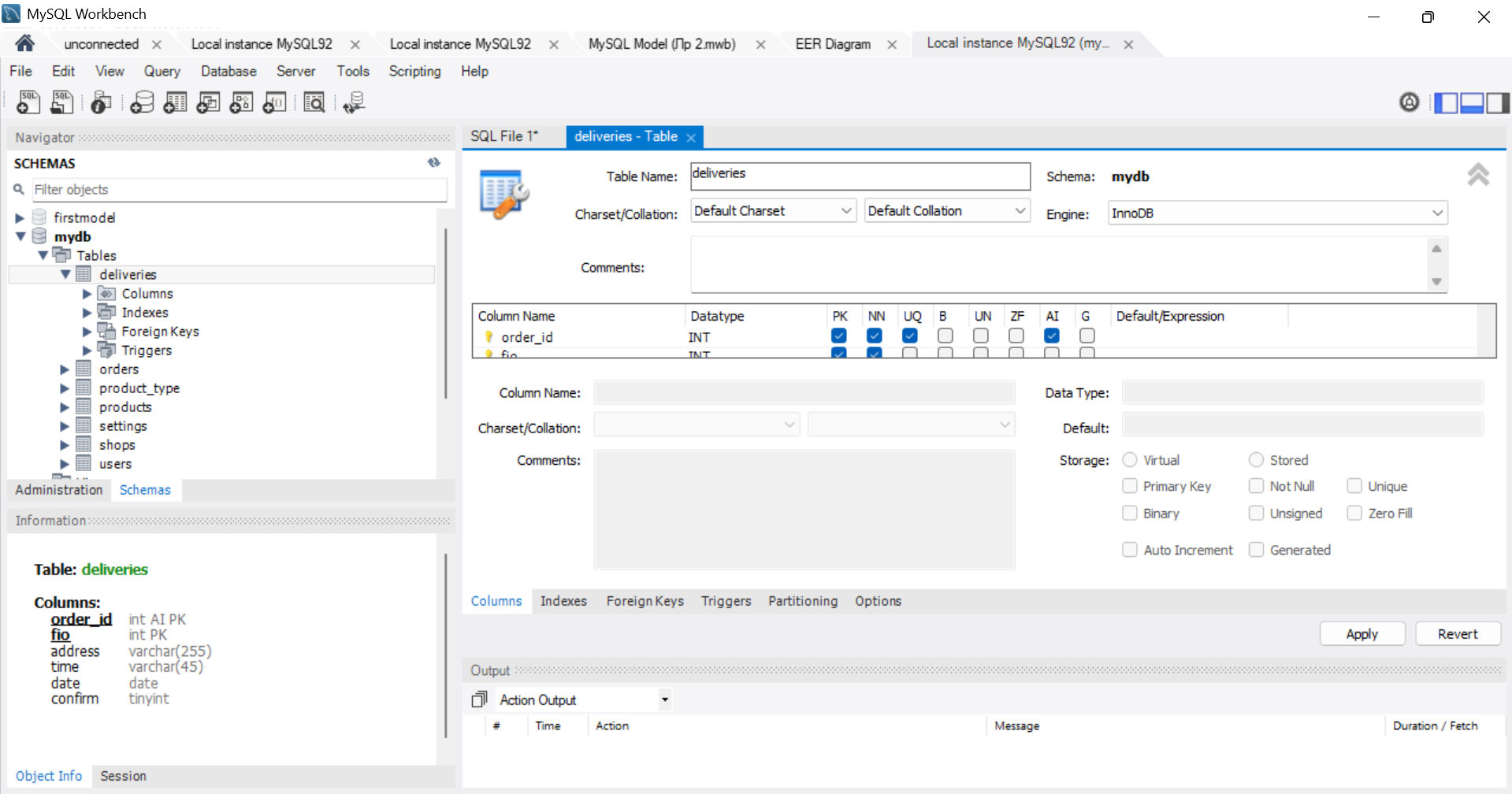
REFERENCES `mydb`.`shops` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

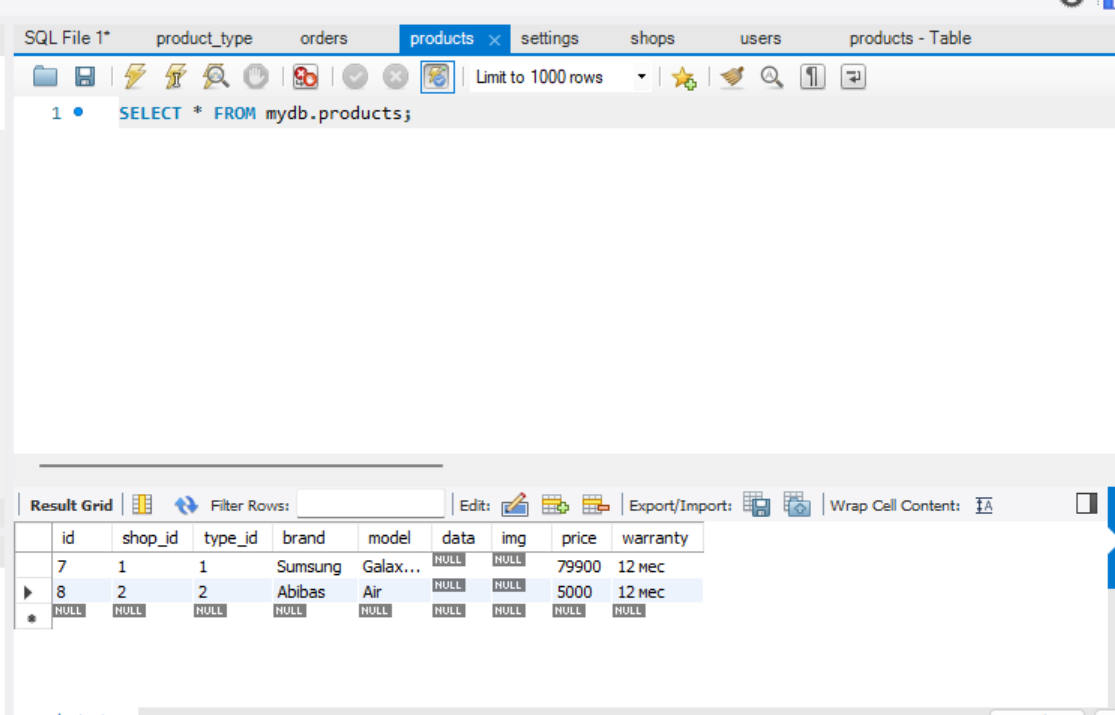
ENGINE = InnoDB

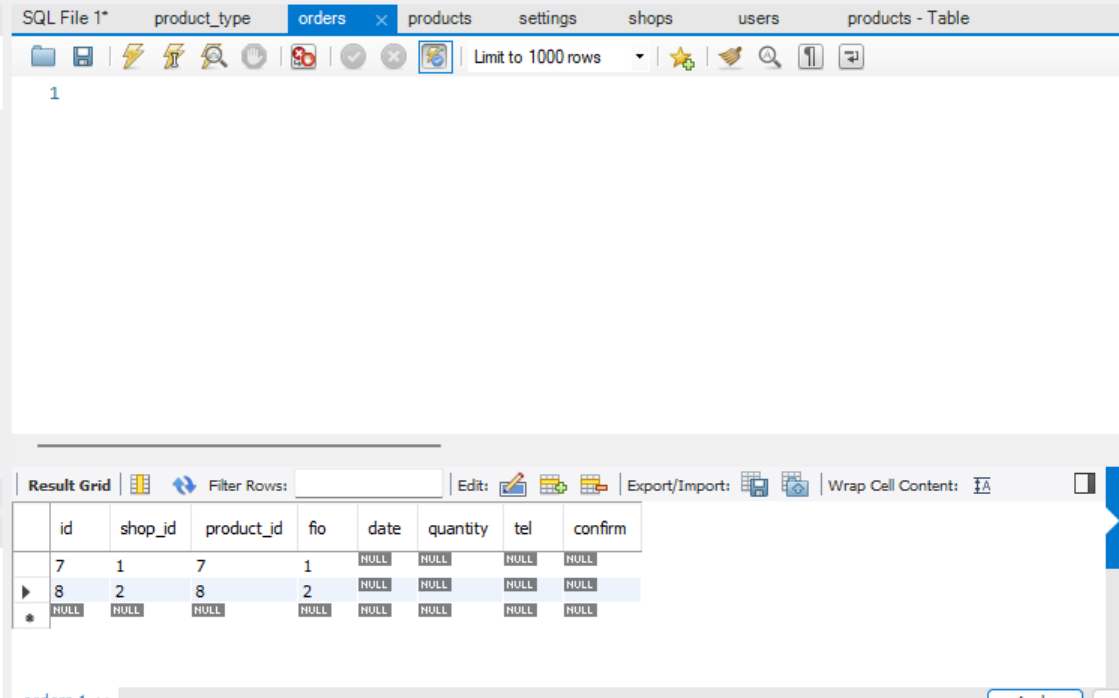
**Задание 3.** Выполните операцию Database - Forward Engineer и создайте базу данных на вашем сервере. Сделайте скриншот с успешным выполнением этого процесса и вставьте его в отчет.



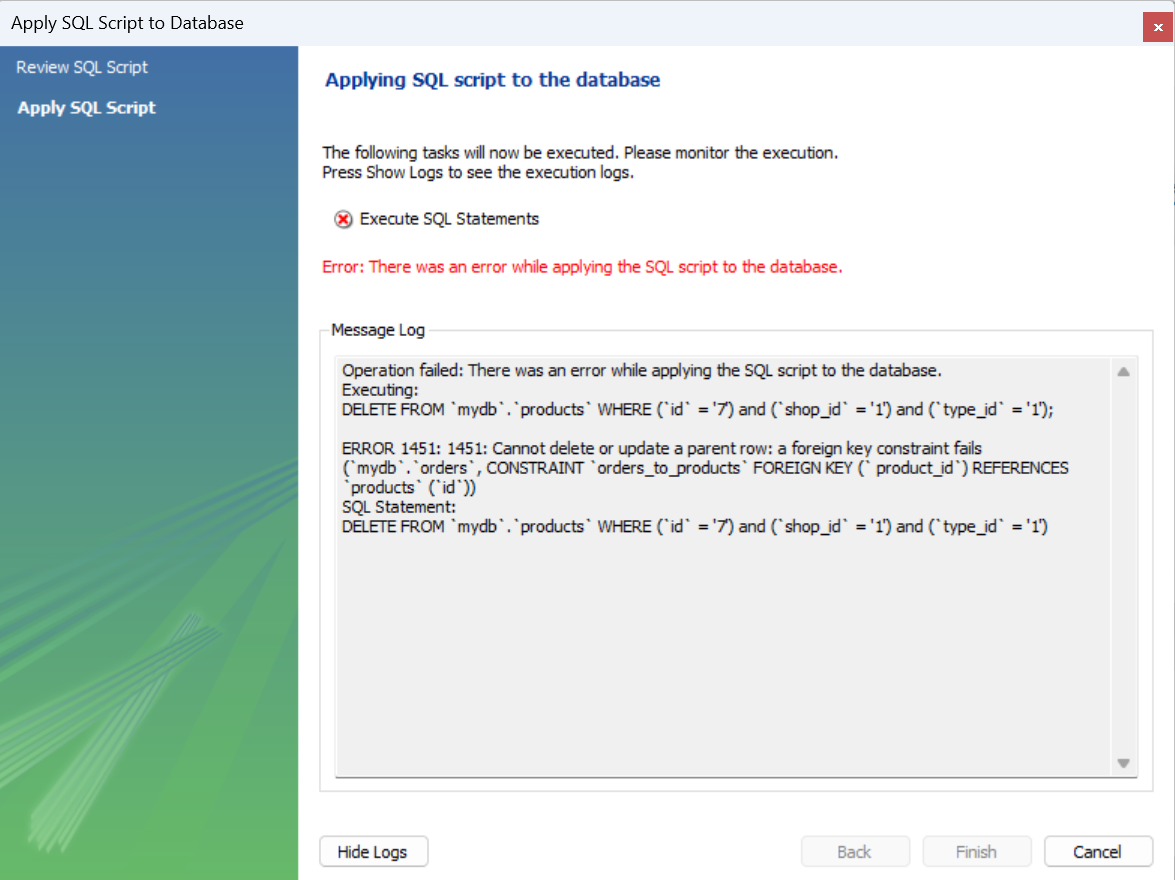
**Задание 4.** Добавьте несколько строк и новых атрибутов в каждую таблицу созданной базы данных. Попробуйте удалить связанные в нескольких таблицах данные, зафиксируйте, что произошло и опишите текстом (и по возможности дополните скриншотами) в отчёте.

Заполним таблицы, подробнее рассмотрим таблицу products и orders. Добавим в таблицу products несколько товаров, в таблицу orders несколько заказов:





Попробуем удалить товар с id 7 из products.



Видим, что этого сделать не удается, так как система выдает ошибку «Cannot delete or update a parent row: a foreign key constraint fails». Это значит, что система не даёт удалить товар, так как на него ссылается заказ.