

МИНОБРНАУКИ РОССИИ

РГУ НЕФТИ И ГАЗА (НИУ) ИМЕНИ И.М. ГУБКИНА

Факультет Автоматики и вычислительной техники
Кафедра Автоматизированных систем управления

Оценка комиссии: _____ Рейтинг: _____
Подписи членов комиссии:

_____	<u>Папилина Т.М.</u>
(подпись)	(фамилия, имя, отчество)
_____	<u>Волков Д.А.</u>
(подпись)	(фамилия, имя, отчество)

(дата)	

КУРСОВАЯ РАБОТА

по дисциплине _____ Базы данных

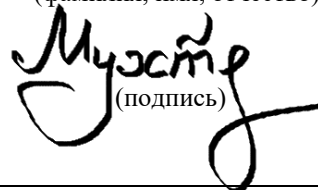
на тему _____ Проектирование реляционной базы данных

«К ЗАЩИТЕ»

ВЫПОЛНИЛ:

Студент группы АС-20-04
(номер группы)

Мухтаров Тимерлан Тахирович
(фамилия, имя, отчество)


(подпись)

(должность, ученая степень; фамилия, и.о.)

(подпись)

(дата)

(дата)

Москва, 20 22

МИНОБРНАУКИ РОССИИ

РГУ НЕФТИ И ГАЗА (НИУ) ИМЕНИ И.М. ГУБКИНА

Факультет Автоматики и вычислительной техники
Кафедра Автоматизированных систем управления

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

по дисциплине _____ Базы данных

на тему _____ Проектирование реляционной базы данных

ДАНО студенту Мухтарову Тимерлану Тахировичу группы АС-20-04
(фамилия, имя, отчество в дательном падеже) (номер группы)

Содержание работы:

1. Концептуальное проектирование
2. Логическое проектирование
3. Физическое проектирование
4. Создание представлений для работы с БД

Исходные данные для выполнения работы:

1. _____

Рекомендуемая литература:

1. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем : учеб. пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 368 с.

Графическая часть:

1. ER-диаграмма базы данных

Руководитель: _____
(уч. степень) (должность) (подпись) (фамилия, имя, отчество)

Задание принял к исполнению: студент _____ Мухтаров Т.Т.
(подпись) (фамилия, имя, отчество)

Оглавление

ВВЕДЕНИЕ	4
КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ	5
ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ	8
ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ	10
СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ	15
ЗАКЛЮЧЕНИЕ.....	20
СПИСОК ЛИТЕРАТУРЫ	21
ПРИЛОЖЕНИЕ 1	22

ВВЕДЕНИЕ

Целью данной работы является проектирование реляционной базы данных, через прохождение 4 последовательных этапов: концептуальное проектирование, логическое проектирование, физическое проектирование и создание представлений.

В качестве тематики для проектирования мной была выбрана модель сети шаурмичных под брендом «Waupmust».

При выполнении данной работы я буду использовать реляционную СУБД – MySQL, в частности инструмент для визуального проектирования баз данных – MySQL Workbench [4].

В результате выполнения задания мы получим реляционную базу данных сети шаурмичных.

MySQL – свободная реляционная система управления базами данных, разработку и поддержку которой осуществляет компания Oracle.

База данных – это совокупность взаимосвязанных, хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

СУБД (Система Управления Базами Данных) – программа поддержки интегрированной совокупности данных, предназначенная для создания, ведения и использования базы данных многими пользователями.

Реляционная модель базы данных характеризуется тем, что организация данных производится в виде двумерных таблиц отношений, где каждая таблица обладает свойствами:

- 1) все столбцы однородны;
- 2) каждый столбец имеет уникальное имя;
- 3) отсутствуют одинаковые строки;
- 4) порядок строк и столбцов неважен.

КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Концептуальное проектирование заключается в создании абстрактной (концептуальной) модели базы данных. Эта модель данных создаётся на основе информации, записанной в спецификациях требований пользователей.

Концептуальное проектирование базы данных абсолютно не зависит от таких подробностей её реализации, как выбранная целевая СУБД, набор создаваемых прикладных программ, используемые языки программирования, тип выбранной вычислительной платформы, а также от любых других особенностей физической реализации [1].

На данном этапе выбирается тема для проектируемой базы данных. В моём случае – сеть шаурмичных. Выделяются сущности: сеть магазинов, персонал, поставщики, меню, продукты, поставщики-продукты, проданные блюда, купленные продукты, денежная статистика по месяцам и магазинам. Всё это отображается на бумаге и называется концептом баз данных (Рисунок 1 Концепт сети кафе).

Помимо прочего прописываются атрибуты каждой сущности:

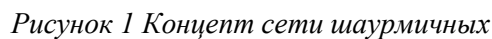
- Сеть Магазинов: id, адрес, окупаемость в месяцах, дата открытия, затраты на открытие
- Персонал: id, ФИО, серия и номер паспорта, наличие мед книжки и гражданства РФ, должность, телефонный номер, зарплата, id рабочего «кафе»
- Поставщики: id, телефон и название компании поставщика
- Меню: id, название блюда, стоимость.
- Продукты: название продукта.
- Поставщики-продукты: id, id поставщика, название продукта, цена за кг/шт
- Денежная статистика по месяцам и магазинам: id, id «кафе», месяц.год продаж, стоимость аренды, стоимость ЖКХ за данный период

- Проданные блюда: id, id проданного блюда, его количество, id привязки к «кафе» и дате
- Купленные продукты: id, id купленного продукта у поставщика, его количество, id привязки к «кафе» и дате

Между собой данные сущности связаны следующим образом:

- Сеть и персонал: один ко многим (идентифицирующая связь) – в каждом магазине работает несколько людей
- Сеть и денеж.статистика по месяцам: один ко многим (идентифицирующая связь) – деятельность кафе разбивается на месяцы. 1 кафе -> 1 кафе в разные месяцы.
- Меню и продукты: многие ко многим (идентифицирующая связь) – одно блюдо может состоять из нескольких продуктов, точно также как один продукт может быть частью нескольких блюд
- Поставщики и поставщики-продукты: один ко многим (неидентифицирующая связь) – 1 поставщик может поставлять разные продукты по разным ценам
- Продукты и поставщики продукты: один ко многим (неидентифицирующая связь) – однозначное определение наименований продуктов в зависимой таблице
- Поставщики-продукты и купленные продукты: один ко многим (идентифицирующая связь) – одна и та же поставка по той же цене может быть в течение n-го количества месяцев.
- Меню и проданные блюда: один ко многим (идентифицирующая связь) - однозначная идентификация проданных блюд с передачей их цен.
- Денеж.статистика по месяцам и кафе и проданные блюда: один ко многим (идентифицирующая связь) – разное количество проданных наименований, привязанных к определенному кафе в

- Денеж.статистика по месяцам и кафе и купленные продукты: один ко многим (идентифицирующая связь) – разное количество купленных наименований, привязанных к определенному кафе в определенный месяц



ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

На втором этапе – в логическом проектировании – создаётся схема базы данных, основанная на концептуальной модели данных, спроектированной на первом этапе. Учитываются особенности выбранной модели данных, в данном случае – реляционной модели данных. Создаём схему отношений сущностей, которые мы выявили на первом этапе, с указанием их первичных ключей.

Этот этап целиком осуществляется с помощью инструмента MySQL Workbench. В этой программе строим схему нашей базы данных, которая называется EER-диаграмма (Рисунок 2 EER-диаграмма базы данных).

На этом этапе будет добавлена промежуточная сущность – Меню_Продукты – поскольку в рамках нормализации отношений мы должны избавиться от связи «многие ко многим». Благодаря промежуточной сущности наша связь «многие ко многим» не исчезает, а реализуется как 2 связи «один ко многим» и дополнительную сущность, содержащую id из меню и название продукта из продуктов.

Первичный ключ – поле, или группа полей, позволяющие однозначно определить каждую запись [2].

Первичными ключами у каждой сущности являются поля id, описанные выше на предыдущем этапе.

Внешний ключ – поле, по которому происходит связь двух сущностей.

Внешними ключами у сущностей являются:

- Personal (персонал) – idshopwork.
- Sold_Bluda (Проданные блюда) – idbluda, idinoutcome
- Bought_products (купленные продукты) – idinoutcome, Suppliers_Products_id.
- Suppliers_Products (Поставщики-продукты) – idPostavshika

■ IN_OUT_COME (денеж.статистика по месяцам и кафе) – idshop

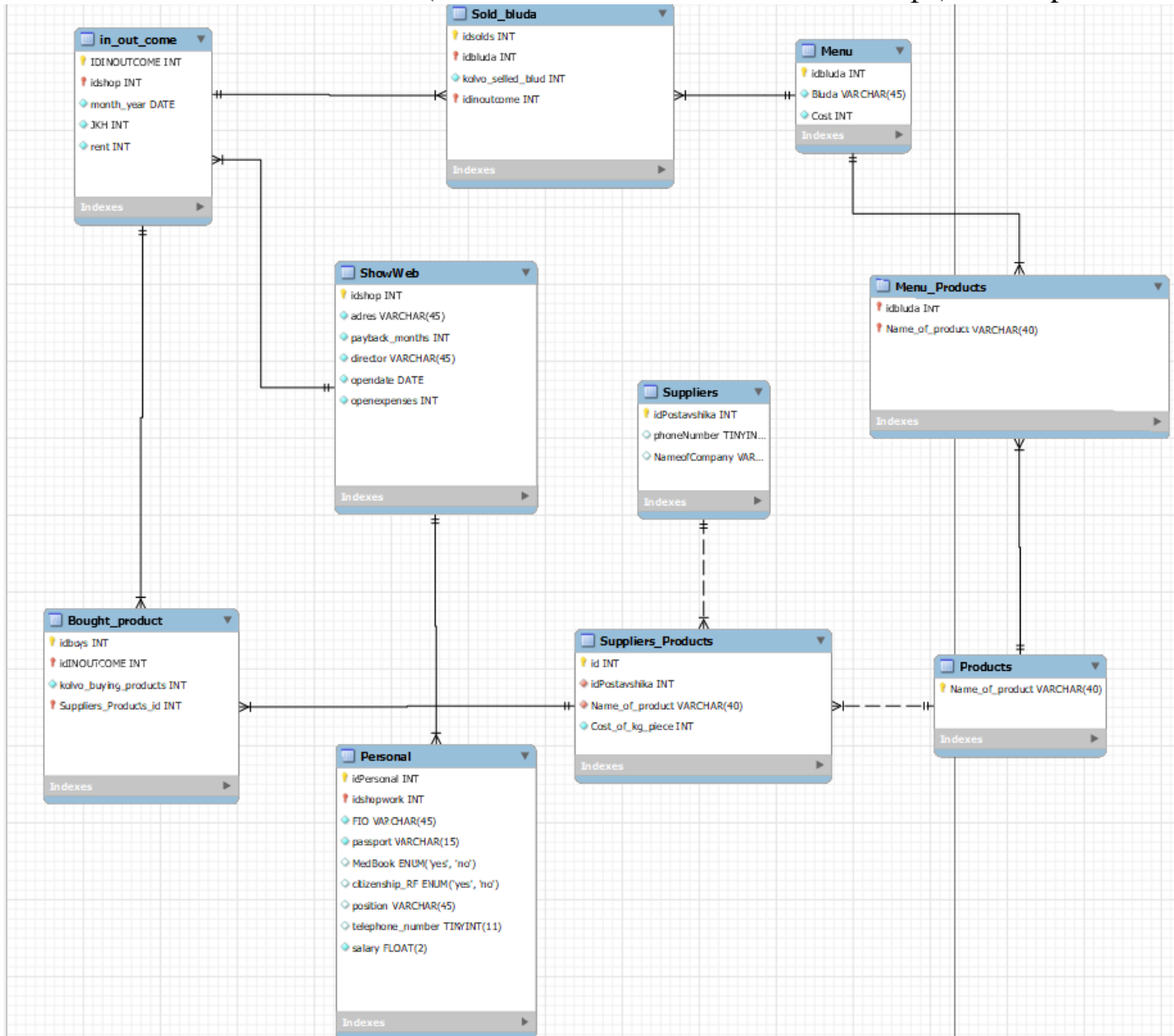


Рисунок 2 EER-диаграмма базы данных

ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Физическое проектирование базы данных – процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах. На этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также связанные с этим ограничения целостности.

Под целостностью базы данных подразумевается соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам. Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется ограничением целостности [3].

Основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

В случае реляционной модели данных под этим подразумевается следующее:

- Создание набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной логической модели данных.
- Определение конкретных структур хранения данных и методов доступа к ним, обеспечивающих оптимальную производительность СУБД.
- Разработка средств защиты создаваемой системы.

Задача этого этапа – создание скрипта на языке SQL, а также обеспечение целостности связей между сущностями в базе данных.

Перед тем, как генерировать скрипт, заполним в MySQL Workbench базу данных информацией, а также проставим условия ON DELETE и ON UPDATE.

После этого на вкладке File верхнего меню выбираем Export, затем нажимаем Forward Engineer SQL Create script (Рисунок 3 Начало формирования скрипта).

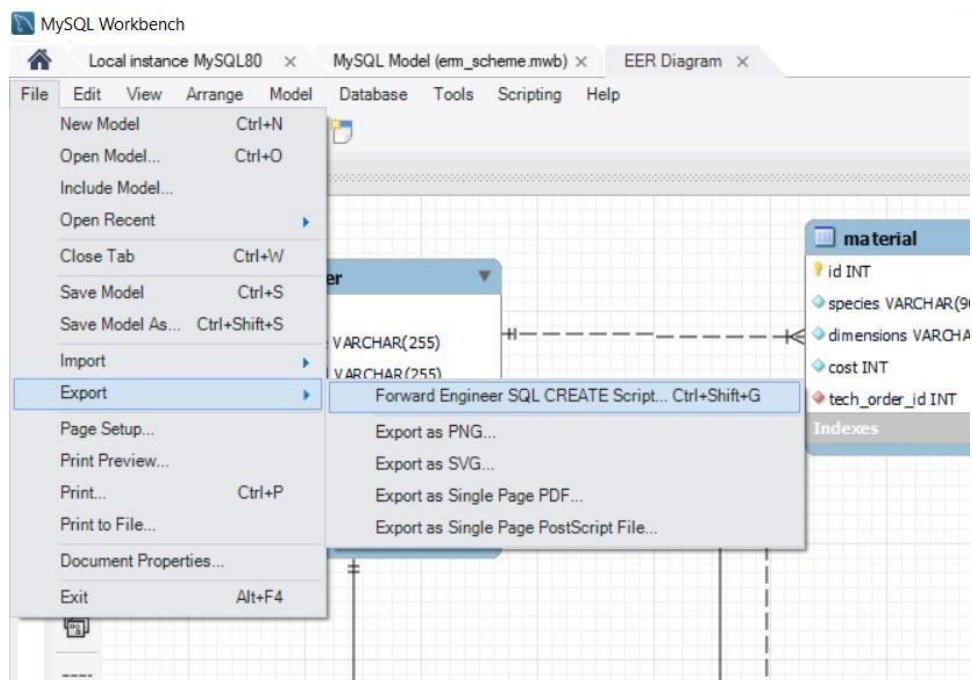


Рисунок 3 Начало формирования скрипта

Затем в открывшемся окне выбираем путь, куда мы сохраним наш будущий скрипт, нажимаем на условие «Generate INSERT statements for tables», чтобы данные, внесённые нами в MySQL Workbench, перенеслись на реальную базу данных и начинаем формирование скрипта (Рисунок 4 Формирование скрипта, Рисунок 5 Формирование скрипта, Рисунок 6 Формирование скрипта).

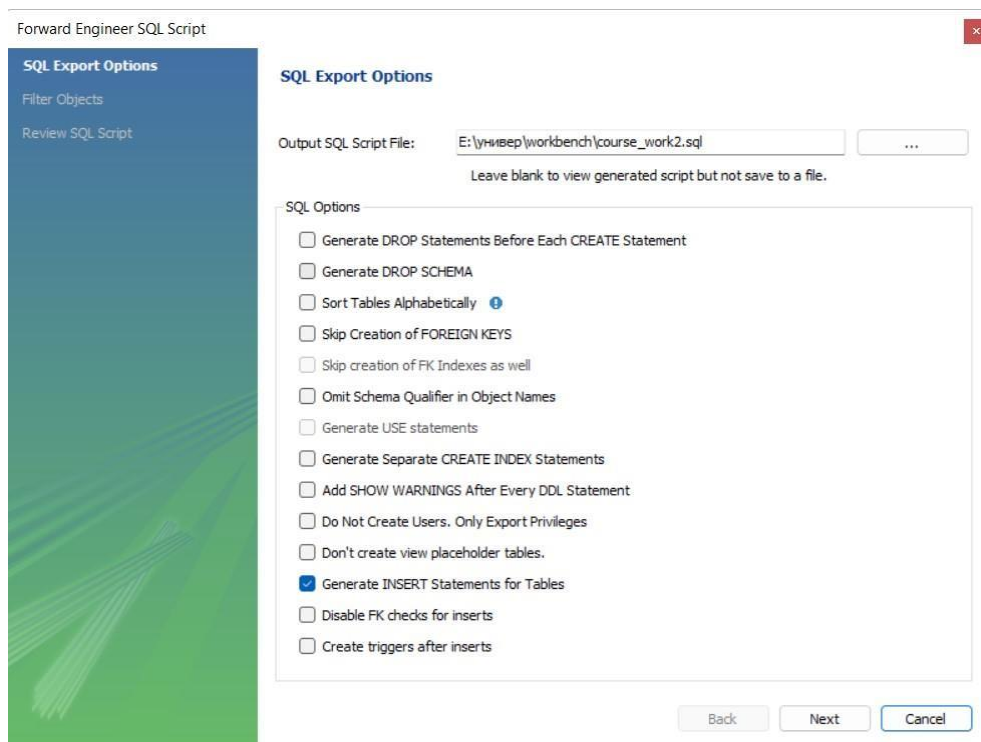


Рисунок 4 Формирование скрипта

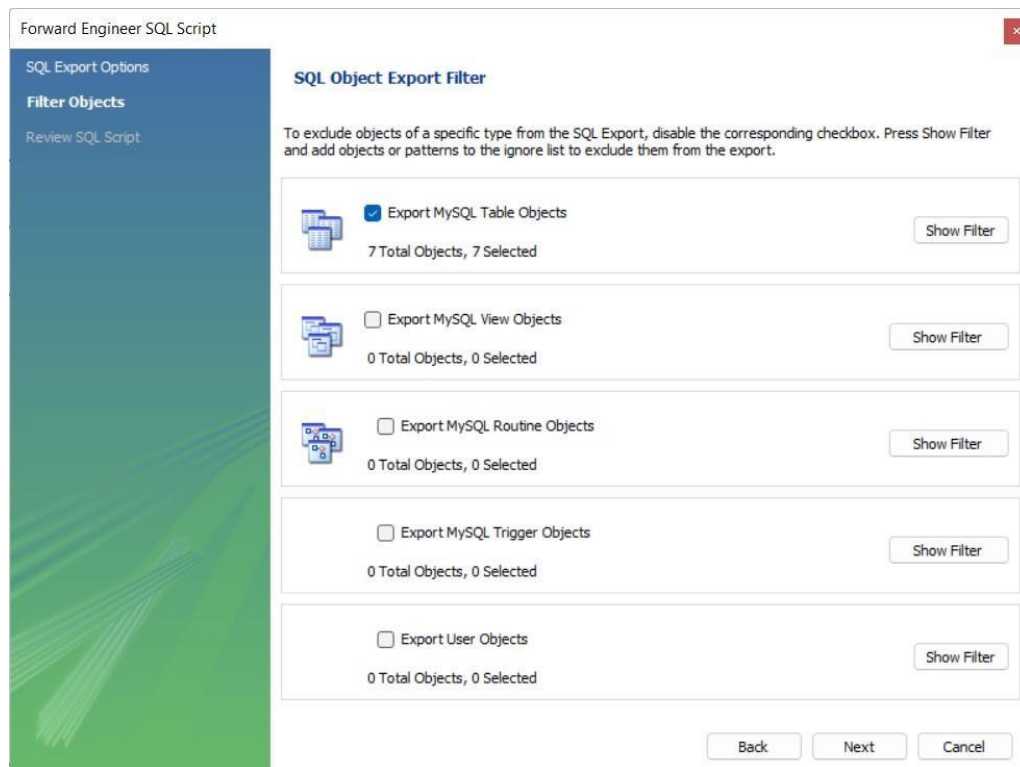


Рисунок 5 Формирование скрипта



Рисунок 6 Формирование скрипта

После того как SQL скрипт сформировался, необходимо запустить его, чтобы база данных смонтировалась на устройство. Для этого выполняем локальное подключение, открываем и запускаем скрипт (Рисунок 7 Демонстрация запуска скрипта, Рисунок 8 Демонстрация запуска скрипта, Рисунок 9 Демонстрация запуска скрипта).

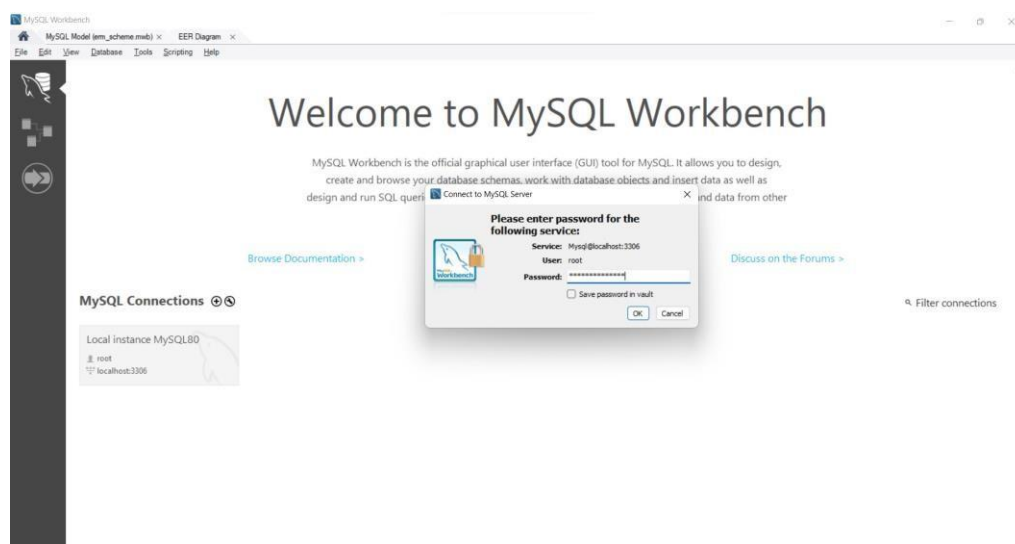


Рисунок 7 Демонстрация запуска скрипта

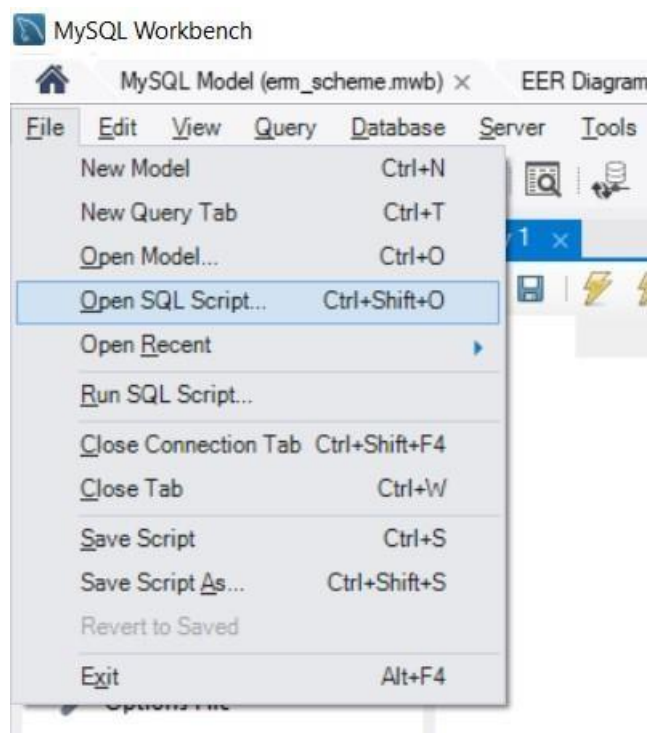


Рисунок 8 Демонстрация запуска скрипта

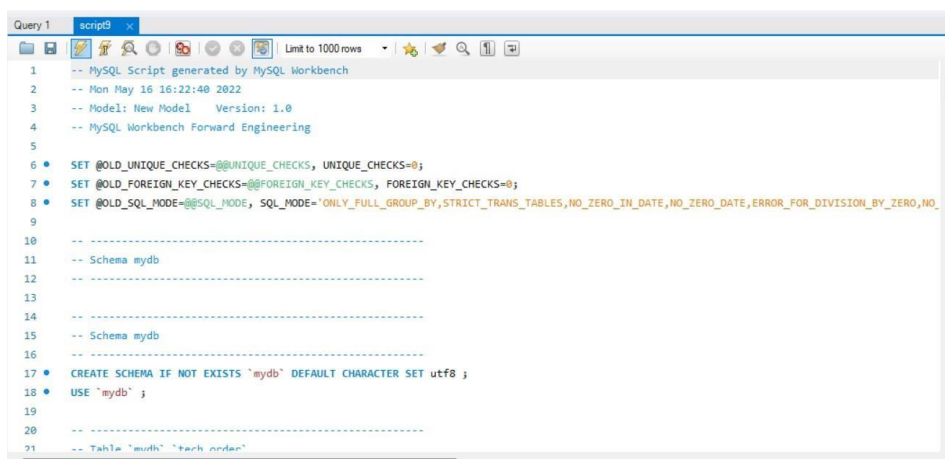


Рисунок 9 Демонстрация запуска скрипта

После нажатия на выделенную на последнем скриншоте жёлтую молнию, база данных будет смонтирована на устройство. Я выполнять скрипт не стал, так как у меня уже смонтирована финальная версия базы данных.

Листинг скрипта приведён в Приложении 1.

СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ

Представление или view – это виртуальная таблица, созданная запросом, которая обычно объединяет несколько таблиц.

Создаётся представление с помощью команды «create view view_name as select ...».

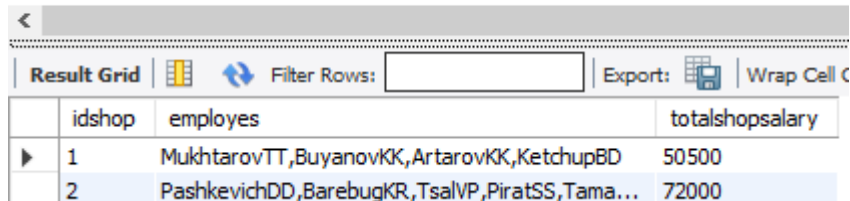
Для данной работы было создано 7 представлений:

1. Показывает, какой персонал работает в магазинах и суммарную зарплату, выделяемую персоналу (Рисунок 10 Демонстрация работы 1-го представления):

```
«CREATE VIEW `view2` AS SELECT idshopwork as idshop, group_concat(FIO) as employes, sum(salary) as totalshopsalary from personal group by idshopwork;»
```

Результат:

520 • `select * from view2;`



	idshop	employes	totalshopsalary
1		MukhtarovTT,BuyanovKK,ArtarovKK,KetchupBD	50500
2		PashkevichDD,BarebugKR,TsalVP,PiratSS,Tama...	72000

Рисунок 10 Демонстрация работы 1-го представления

2. Показывает, прибыль по наименованиям продуктов, привязанную к дате и «кафе» (Рисунок 11 Демонстрация работы 2-го представления):

- ```
CREATE VIEW `view3` AS select sold_bluda.idinoutcome, sold_bluda.idbluda,Menu.Bluda,sold_bluda.kolvo_selled_blud,Menu.cost, Menu.cost*sold_bluda.kolvo_selled_blud as Gain from sold_bluda inner join Menu on sold_bluda.idbluda=Menu.idbluda;
```

Результат:

521 • `select * from view3;`

|   | idinoutcome | idbluda | Bluda     | kolvo_selled_blud | cost | Gain   |
|---|-------------|---------|-----------|-------------------|------|--------|
| ▶ | 1           | 1       | shaverma1 | 600               | 180  | 108000 |
|   | 2           | 1       | shaverma1 | 380               | 180  | 68400  |
|   | 3           | 1       | shaverma1 | 500               | 180  | 90000  |
|   | 1           | 2       | shaverma2 | 1500              | 220  | 330000 |
|   | 2           | 2       | shaverma2 | 700               | 220  | 154000 |
|   | 3           | 2       | shaverma2 | 900               | 220  | 198000 |
|   | 1           | 3       | shaverma3 | 1800              | 350  | 630000 |

Рисунок 11 Демонстрация работы 2-го представления

3. Показывает суммарную выручку с проданных блюд кафе по месяцам (Рисунок 12 Демонстрация работы 3-го представления):

```
CREATE VIEW `view4` AS select view3.idinoutcome, in_out_come.idshop,
in_out_come.month_year, SUM(Gain) as TotalGain
from in_out_come,view3 where in_out_come.idinoutcome=view3.idinoutcome
group by view3.idinoutcome, in_out_come.idshop;
```

Результат:

522 • `select * from view4;`

|   | idinoutcome | idshop | month_year | TotalGain |
|---|-------------|--------|------------|-----------|
| ▶ | 1           | 1      | 2001-01-20 | 1130000   |
|   | 2           | 1      | 2001-02-20 | 918300    |
|   | 3           | 2      | 2001-01-20 | 1040000   |

Рисунок 12 Демонстрация работы 3-го представления

4. Показывает, затраты на продукты по наименованиям (Рисунок 13 Демонстрация работы 4-го представления):



```
CREATE VIEW `view1` AS select
bought_product.idinoutcome,bought_product.Suppliers_Products_id,Suppliers_Pr
oducts.Name_of_product,
bought_product.kolvo_buying_products,Suppliers_Products.Cost_of_kg_piece,
Suppliers_Products.Cost_of_kg_piece*bought_product.kolvo_buying_products as
Losts
from bought_product inner join Suppliers_Products on
bought_product.Suppliers_Products_id=Suppliers_Products.id;
```

Результат:

523 • `select * from view1;`

Result Grid | Filter Rows: | Export: | Wrap Cell Content: `⌂`

|   | idinoutcome | Suppliers_Products_id | Name_of_product | kolvo_buying_products | Cost_of_kg_piece | Losts |
|---|-------------|-----------------------|-----------------|-----------------------|------------------|-------|
| ▶ | 1           | 1                     | morkov          | 300                   | 45               | 13500 |
|   | 2           | 1                     | morkov          | 250                   | 45               | 11250 |
|   | 3           | 1                     | morkov          | 300                   | 45               | 13500 |
|   | 1           | 2                     | kapusta         | 150                   | 150              | 22500 |
|   | 2           | 2                     | kapusta         | 125                   | 150              | 18750 |
|   | 3           | 2                     | kapusta         | 140                   | 150              | 21000 |
|   | .           | .                     | .               | ...                   | ..               | ..... |

Рисунок 13 Демонстрация работы 4-го представления



- Показывает, суммарные затраты кафе на продукты по месяцам

(Рисунок 14 Демонстрация работы 5-го представления):

```
CREATE VIEW `view5` AS select view1.idinoutcome, in_out_come.idshop,
in_out_come.month_year, SUM(Losts) as TotalLosts
from in_out_come,view1 where in_out_come.idinoutcome=view1.idinoutcome
group by view1.idinoutcome, in_out_come.idshop;Результат:
```

524 • `select * from view5;`

<

Result Grid   Filter Rows:

|     | idinoutcome | idshop | month_year | TotalLosses |
|-----|-------------|--------|------------|-------------|
| ▶ 1 | 1           | 1      | 2001-01-20 | 278500      |
| 2   | 1           | 1      | 2001-02-20 | 228000      |
| 3   | 2           | 2      | 2001-01-20 | 222500      |

Рисунок 14 Демонстрация работы 5-го представления

6. Показывает чистую прибыль кафе по месяцам с учетом аренды, коммунальных услуг, прибыли за месяц, затрат за месяц, суммарной зарплаты персонала кафе, окупаемости кафе(если с момента открытия кафе до рассматриваемого месяца работы кафе не прошло кол-во месяцев окупаемости то в учет чистый прибыли идет «-затраты на открытие/кол-во месяцев окупаемости»)(Рисунок 15 Демонстрация работы 6-го представления):

```
CREATE VIEW `view7` AS SELECT
IN_out_come.idinoutcome,IN_out_come.idshop,IN_out_come.month_year,show
web.opendate, view4.TotalGain, view5.TotalLosses,
IN_out_come.rent,IN_out_come.JKH,view2.totalshopsalary,
case
when showweb.opendate > (in_out_come.month_year - interval
showweb.payback_months month)
then view4.TotalGain-view5.TotalLosses-IN_out_come.rent-IN_out_come.JKH-
view2.totalshopsalary-showweb.openexpenses/showweb.payback_months
else view4.TotalGain-view5.TotalLosses-IN_out_come.rent-IN_out_come.JKH-
view2.totalshopsalary
end ClearExpense
from IN_out_come
inner join view4 on view4.idinoutcome=IN_out_come.idinoutcome
inner join view5 on view5.idinoutcome=IN_out_come.idinoutcome
inner join view2 on view2.idshop=IN_out_come.idshop
```

inner join showweb on showweb.idshop=IN\_out\_come.idshop;

Результат:

525 • `select * from view7;`

|   | idinoutcome | idshop | month_year | opendate   | TotalGain | TotalLosses | rent  | JKH  | totalshopsalary | ClearExpense     |
|---|-------------|--------|------------|------------|-----------|-------------|-------|------|-----------------|------------------|
| ▶ | 1           | 1      | 2001-01-20 | 2001-01-20 | 1130000   | 278500      | 15000 | 2000 | 50500           | 775666.666666667 |
|   | 2           | 1      | 2001-02-20 | 2001-01-20 | 918300    | 228000      | 15000 | 2200 | 50500           | 614266.666666667 |
|   | 3           | 2      | 2001-01-20 | 2001-02-20 | 1040000   | 222500      | 13000 | 2140 | 72000           | 724110           |

Рисунок 15 Демонстрация работы 6-го представления

7. Показывает чистую прибыль «кафе» за все время работы (Рисунок 16 Демонстрация работы 6-го представления):

```
CREATE VIEW `view6` AS select idshop, sum(ClearExpense) as
Total_Clear_Expense_of_All_Time from view7 group by idshop;
```

Результат:

526 • `select * from view6;`  
527

|   | idshop | Total_Clear_Expense_of_All_Time |
|---|--------|---------------------------------|
| ▶ | 1      | 1389933.333333334               |
|   | 2      | 724110                          |

Рисунок 16 Демонстрация работы 7-го представления

## **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсовой работы была создана реляционная база данных по тематике сети шаурмичных. Были последовательно пройдены три основные этапа проектирования баз данных: концептуальное проектирование, логическое проектирование и физическое проектирование. Также, в ходе работы были созданы представления, которые могут быть полезны пользователям для работы с базой данных.

Я ознакомился с процессом проектирования базы данных на языке SQL, а также изучил инструмент для визуального проектирования баз данных – MySQL Workbench.

## СПИСОК ЛИТЕРАТУРЫ

- [1] Базы данных [Электронный ресурс]: учебное пособие /. — Саратов: Научная книга, 2012. — 158 с. — Режим доступа:  
<http://www.iprbookshop.ru/6261.html>
- [2] Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем: учеб. пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. — Москва: ИД «ФОРУМ»: ИНФРА-М, 2019. — 368 с.
- [3] Мартишин, С.А., Симонов В.Л., Храпченко М.В.. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: Методы и средства проектирования информационных систем и технологий - М.: Форум, 2018. - 61 с.
- [4] MySQL Workbench Reference Manual – Режим доступа:  
<https://dev.mysql.com/doc/workbench/en/>

## ПРИЛОЖЕНИЕ 1

```
-- MySQL Script generated by MySQL Workbench
-- Fri May 27 23:01:18 2022
-- Model: New Model Version: 1.0
-- MySQL Workbench Forward Engineering
```

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DAT
E,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';
```

```
-- -----
-- Schema mydb
-- -----
```

```
DROP SCHEMA IF EXISTS `mydb` ;
```

```
-- -----
-- Schema mydb
-- -----
```

```
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;
```

```
-- -----
-- Table `mydb`.`ShowWeb`
-- -----
```

```
DROP TABLE IF EXISTS `mydb`.`ShowWeb` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`ShowWeb` (
 `idshop` INT NOT NULL AUTO_INCREMENT,
 `adres` VARCHAR(45) NOT NULL,
 `payback_months` INT NOT NULL,
 `director` VARCHAR(45) NOT NULL,
 `opendate` DATE NOT NULL,
 `openexpenses` INT NOT NULL,
 PRIMARY KEY (`idshop`))
ENGINE = InnoDB;
```

```
-- -----
-- Table `mydb`.`in_out_come`
-- -----
```

```
DROP TABLE IF EXISTS `mydb`.`in_out_come` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`in_out_come` (
 `IDINOUTCOME` INT NOT NULL AUTO_INCREMENT,
 `idshop` INT NOT NULL,
 `month_year` DATE NOT NULL,
 `JKH` INT NOT NULL,
 `rent` INT NOT NULL,
 PRIMARY KEY (`IDINOUTCOME`, `idshop`),
 INDEX `fk_showweb_inuotcome_idx` (`idshop` ASC) VISIBLE,
 CONSTRAINT `fk_showweb_inuotcome`
 FOREIGN KEY (`idshop`)
 REFERENCES `mydb`.`ShowWeb` (`idshop`)
 ON DELETE RESTRICT
```

```
 ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```

-- Table `mydb`.`Menu`

```

```
DROP TABLE IF EXISTS `mydb`.`Menu` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Menu` (
 `idbluda` INT NOT NULL AUTO_INCREMENT,
 `Bluda` VARCHAR(45) NOT NULL,
 `Cost` INT NOT NULL,
 PRIMARY KEY (`idbluda`))
ENGINE = InnoDB;
```

```

-- Table `mydb`.`Products`

```

```
DROP TABLE IF EXISTS `mydb`.`Products` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Products` (
 `Name_of_product` VARCHAR(40) NOT NULL,
 PRIMARY KEY (`Name_of_product`))
ENGINE = InnoDB;
```

```

-- Table `mydb`.`Personal`

```

```
DROP TABLE IF EXISTS `mydb`.`Personal` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Personal` (
 `idPersonal` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `idshopwork` INT NOT NULL,
 `FIO` VARCHAR(45) NOT NULL,
 `passport` VARCHAR(15) NOT NULL,
 `MedBook` ENUM('yes', 'no') NULL,
 `citizenship_RF` ENUM('yes', 'no') NULL,
 `position` VARCHAR(45) NULL,
 `telephone_number` TINYINT(11) NULL,
 `salary` FLOAT(2) NOT NULL,
 PRIMARY KEY (`idPersonal`, `idshopwork`),
 INDEX `fk_shop_idx` (`idshopwork` ASC) VISIBLE,
 CONSTRAINT `fk_shop`
 FOREIGN KEY (`idshopwork`)
 REFERENCES `mydb`.`ShowWeb` (`idshop`)
 ON DELETE CASCADE
 ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```

-- Table `mydb`.`Suppliers`

```

```
DROP TABLE IF EXISTS `mydb`.`Suppliers` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Suppliers` (
 `idPostavshika` INT NOT NULL AUTO_INCREMENT,
 `phoneNumber` TINYINT(20) NULL,
 `NameofCompany` VARCHAR(45) NULL,
 PRIMARY KEY (`idPostavshika`))
ENGINE = InnoDB;
```

```
-- Table `mydb`.`Sold_bluda`
```

```
DROP TABLE IF EXISTS `mydb`.`Sold_bluda` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Sold_bluda` (
 `idsolds` INT NOT NULL AUTO_INCREMENT,
 `idbluda` INT NOT NULL,
 `kolvo_solded_blud` INT NOT NULL,
 `idinoutcome` INT NOT NULL,
 INDEX `fk_svyazmenu_sellbluda_idx` (`idbluda` ASC) VISIBLE,
 PRIMARY KEY (`idsolds`, `idinoutcome`, `idbluda`),
 INDEX `fk_bluda_inoutcome_idx` (`idinoutcome` ASC) VISIBLE,
 CONSTRAINT `fk_svyazmenu_sellbluda`
 FOREIGN KEY (`idbluda`)
 REFERENCES `mydb`.`Menu` (`idbluda`)
 ON DELETE CASCADE
 ON UPDATE CASCADE,
 CONSTRAINT `fk_bluda_inoutcome`
 FOREIGN KEY (`idinoutcome`)
 REFERENCES `mydb`.`in_out_come` (`IDINOUTCOME`)
 ON DELETE CASCADE
 ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- Table `mydb`.`Поставщики_has_Поставщики`
```

```
DROP TABLE IF EXISTS `mydb`.`Поставщики_has_Поставщики` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Поставщики_has_Поставщики` (
 `Поставщики_idPostavshika` INT NOT NULL,
 `Поставщики_idPostavshika1` INT NOT NULL,
 PRIMARY KEY (`Поставщики_idPostavshika`, `Поставщики_idPostavshika1`),
 INDEX `fk_Поставщики_has_Поставщики_Поста_idx` (`Поставщики_idPostavshika1` ASC) VISIBLE,
 INDEX `fk_Поставщики_has_Поставщики_Поста_idx1` (`Поставщики_idPostavshika` ASC) VISIBLE,
 CONSTRAINT `fk_Поставщики_has_Поставщики_Постав`
 FOREIGN KEY (`Поставщики_idPostavshika`)
 REFERENCES `mydb`.`Suppliers` (`idPostavshika`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION,
 CONSTRAINT `fk_Поставщики_has_Поставщики_Постав1`
 FOREIGN KEY (`Поставщики_idPostavshika1`)
 REFERENCES `mydb`.`Suppliers` (`idPostavshika`)
 ON DELETE NO ACTION
 ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



-- Table `mydb`.`Suppliers\_Products`  
-----

DROP TABLE IF EXISTS `mydb`.`Suppliers\_Products` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Suppliers\_Products` (  
 `id` INT NOT NULL,  
 `idPostavshika` INT NOT NULL,  
 `Name\_of\_product` VARCHAR(40) NOT NULL,  
 `Cost\_of\_kg\_piece` INT NOT NULL,  
 INDEX `fk\_Поставщики\_has\_Продукты\_Продукт\_idx` (`Name\_of\_product` ASC) VISIBLE,  
 INDEX `fk\_Поставщики\_has\_Продукты\_Поставщ\_idx` (`idPostavshika` ASC) VISIBLE,  
 PRIMARY KEY (`id`),  
 CONSTRAINT `fk\_Поставщики\_has\_Продукты\_Поставщи1`  
 FOREIGN KEY (`idPostavshika`)  
 REFERENCES `mydb`.`Suppliers` (`idPostavshika`)  
 ON DELETE RESTRICT  
 ON UPDATE CASCADE,  
 CONSTRAINT `fk\_Поставщики\_has\_Продукты\_Продукты1`  
 FOREIGN KEY (`Name\_of\_product`)  
 REFERENCES `mydb`.`Products` (`Name\_of\_product`)  
 ON DELETE RESTRICT  
 ON UPDATE CASCADE)  
ENGINE = InnoDB;

-- Table `mydb`.`Menu\_Products`  
-----

DROP TABLE IF EXISTS `mydb`.`Menu\_Products` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Menu\_Products` (  
 `idbluda` INT NOT NULL,  
 `Name\_of\_product` VARCHAR(40) NOT NULL,  
 PRIMARY KEY (`idbluda`, `Name\_of\_product`),  
 INDEX `fk\_Меню\_has\_Продукты\_Продукты1\_idx` (`Name\_of\_product` ASC) VISIBLE,  
 CONSTRAINT `fk\_menu\_has\_products\_menu`  
 FOREIGN KEY (`idbluda`)  
 REFERENCES `mydb`.`Menu` (`idbluda`)  
 ON DELETE CASCADE  
 ON UPDATE CASCADE,  
 CONSTRAINT `fk\_menu\_menu\_products`  
 FOREIGN KEY (`Name\_of\_product`)  
 REFERENCES `mydb`.`Products` (`Name\_of\_product`)  
 ON DELETE CASCADE  
 ON UPDATE CASCADE)  
ENGINE = InnoDB;

-- Table `mydb`.`Bought\_product`  
-----

DROP TABLE IF EXISTS `mydb`.`Bought\_product` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Bought\_product` (  
 `idbuys` INT NOT NULL,  
 `idINOUTCOME` INT NOT NULL,  
 `kolvo\_buying\_products` INT NOT NULL,

```

`Suppliers_Products_id` INT NOT NULL,
PRIMARY KEY (`idbuys`, `Suppliers_Products_id`, `idINOUTCOME`),
INDEX `fk_Buying products_Suppliers_Products1_idx` (`Suppliers_Products_id` ASC) VISIBLE,
CONSTRAINT `fk_Buying products_Suppliers_Products1`
 FOREIGN KEY (`Suppliers_Products_id`)
 REFERENCES `mydb`.`Suppliers_Products` (`id`)
 ON DELETE RESTRICT
 ON UPDATE CASCADE,
CONSTRAINT `fk_buyprod_inoutcome`
 FOREIGN KEY (`idINOUTCOME`)
 REFERENCES `mydb`.`in_out_come` (`IDINOUTCOME`)
 ON DELETE CASCADE
 ON UPDATE CASCADE)
ENGINE = InnoDB;

```

```
USE `mydb` ;
```

```

-- Placeholder table for view `mydb`.`view2`

```

```
CREATE TABLE IF NOT EXISTS `mydb`.`view2` (`idshop` INT, `employees` INT, `totalshopsalary` INT);
```

```

-- Placeholder table for view `mydb`.`view3`

```

```
CREATE TABLE IF NOT EXISTS `mydb`.`view3` (`idinoutcome` INT, `idbluda` INT, `Bluda` INT,
`kolvo_selled_blud` INT, `cost` INT, `Gain` INT);
```

```

-- Placeholder table for view `mydb`.`view4`

```

```
CREATE TABLE IF NOT EXISTS `mydb`.`view4` (`idinoutcome` INT, `idshop` INT, `month_year` INT,
`TotalGain` INT);
```

```

-- Placeholder table for view `mydb`.`view1`

```

```
CREATE TABLE IF NOT EXISTS `mydb`.`view1` (`idinoutcome` INT, `Suppliers_Products_id` INT,
`Name_of_product` INT, `kolvo_buying_products` INT, `Cost_of_kg_piece` INT, `Losses` INT);
```

```

-- Placeholder table for view `mydb`.`view5`

```

```
CREATE TABLE IF NOT EXISTS `mydb`.`view5` (`idinoutcome` INT, `idshop` INT, `month_year` INT,
`TotalLosses` INT);
```

```

-- Placeholder table for view `mydb`.`view6`

```

```
CREATE TABLE IF NOT EXISTS `mydb`.`view6` (`idshop` INT, `Total_Clear_Expense_of_All_Time` INT);
```

```

-- Placeholder table for view `mydb`.`view7`

```

```
CREATE TABLE IF NOT EXISTS `mydb`.`view7` (`idinoutcome` INT, `idshop` INT, `month_year` INT,
`opendate` INT, `TotalGain` INT, `TotalLosses` INT, `rent` INT, `JKH` INT, `totalshopsalary` INT, `ClearExpense`
INT);
```

-----  
-- View `mydb`.`view2`  
-----

```
DROP TABLE IF EXISTS `mydb`.`view2`;
DROP VIEW IF EXISTS `mydb`.`view2` ;
USE `mydb`;
CREATE OR REPLACE VIEW `view2` AS SELECT idshopwork as idshop, group_concat(FIO) as employes,
sum(salary) as totalshopsalary from personal group by idshopwork;
```

-----  
-- View `mydb`.`view3`  
-----

```
DROP TABLE IF EXISTS `mydb`.`view3`;
DROP VIEW IF EXISTS `mydb`.`view3` ;
USE `mydb`;
CREATE OR REPLACE VIEW `view3` AS select
sold_bluda.idinoutcome,sold_bluda.idbluda,Menu.Bluda,sold_bluda.kolvo_selled_blud,Menu.cost,
Menu.cost*sold_bluda.kolvo_selled_blud as Gain
from sold_bluda inner join Menu on sold_bluda.idbluda=Menu.idbluda;
```

-----  
-- View `mydb`.`view4`  
-----

```
DROP TABLE IF EXISTS `mydb`.`view4`;
DROP VIEW IF EXISTS `mydb`.`view4` ;
USE `mydb`;
CREATE OR REPLACE VIEW `view4` AS select view3.idinoutcome, in_out_come.idshop,
in_out_come.month_year, SUM(Gain) as TotalGain
from in_out_come,view3 where in_out_come.idinoutcome=view3.idinoutcome
group by view3.idinoutcome, in_out_come.idshop;
```

-----  
-- View `mydb`.`view1`  
-----

```
DROP TABLE IF EXISTS `mydb`.`view1`;
DROP VIEW IF EXISTS `mydb`.`view1` ;
USE `mydb`;
CREATE OR REPLACE VIEW `view1` AS select
bought_product.idinoutcome,bought_product.Suppliers_Products_id,Suppliers_Products.Name_of_product,
bought_product.kolvo_buying_products,Suppliers_Products.Cost_of_kg_piece,
Suppliers_Products.Cost_of_kg_piece*bought_product.kolvo_buying_products as Losses
from bought_product inner join Suppliers_Products on
bought_product.Suppliers_Products_id=Suppliers_Products.id;
```

-----  
-- View `mydb`.`view5`  
-----

```
DROP TABLE IF EXISTS `mydb`.`view5`;
DROP VIEW IF EXISTS `mydb`.`view5` ;
USE `mydb`;
CREATE OR REPLACE VIEW `view5` AS select view1.idinoutcome, in_out_come.idshop,
in_out_come.month_year, SUM(Losses) as TotalLosses
from in_out_come,view1 where in_out_come.idinoutcome=view1.idinoutcome
group by view1.idinoutcome, in_out_come.idshop;
```

-----  
-- View `mydb`.`view6`  
-----

```

DROP TABLE IF EXISTS `mydb`.`view6`;
DROP VIEW IF EXISTS `mydb`.`view6` ;
USE `mydb`;
CREATE OR REPLACE VIEW `view6` AS select idshop, sum(ClearExpense) as
Total_Clear_Expense_of_All_Time from view7 group by idshop;

-- View `mydb`.`view7`

DROP TABLE IF EXISTS `mydb`.`view7`;
DROP VIEW IF EXISTS `mydb`.`view7` ;
USE `mydb`;
CREATE OR REPLACE VIEW `view7` AS SELECT
IN_out_come.idinoutcome,IN_out_come.idshop,IN_out_come.month_year,showweb.opendate, view4.TotalGain,
view5.TotalLosses, IN_out_come.rent,IN_out_come.JKH,view2.totalshopsalary,
case
when showweb.opendate > (in_out_come.month_year - interval showweb.payback_months month)
then view4.TotalGain-view5.TotalLosses-IN_out_come.rent-IN_out_come.JKH-view2.totalshopsalary-
showweb.openexpenses/showweb.payback_months
else view4.TotalGain-view5.TotalLosses-IN_out_come.rent-IN_out_come.JKH-view2.totalshopsalary
end ClearExpense
from IN_out_come
inner join view4 on view4.idinoutcome=IN_out_come.idinoutcome
inner join view5 on view5.idinoutcome=IN_out_come.idinoutcome
inner join view2 on view2.idshop=IN_out_come.idshop
inner join showweb on showweb.idshop=IN_out_come.idshop;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;

-- Data for table `mydb`.`ShowWeb`

START TRANSACTION;
USE `mydb`;
INSERT INTO `mydb`.`ShowWeb` (`idshop`, `adres`, `payback_months`, `director`, `opendate`, `openexpenses`)
VALUES (1, 'ул.Академика Волгина, 2А', 24, 'Абдурашидов Абдулах Акбарович', '01.01.2002', 200000);
INSERT INTO `mydb`.`ShowWeb` (`idshop`, `adres`, `payback_months`, `director`, `opendate`, `openexpenses`)
VALUES (2, 'ул.Бутлерова,3А', 24, 'Атаков Ашан Шавермович', '01.02.2000', 150000);

COMMIT;

-- Data for table `mydb`.`in_out_come`

START TRANSACTION;
USE `mydb`;
INSERT INTO `mydb`.`in_out_come` (`IDINOUTCOME`, `idshop`, `month_year`, `JKH`, `rent`) VALUES (1, 1,
'01.01.2002', 2000, 15000);
INSERT INTO `mydb`.`in_out_come` (`IDINOUTCOME`, `idshop`, `month_year`, `JKH`, `rent`) VALUES (2, 1,
'01.02.2002', 2200, 15000);
INSERT INTO `mydb`.`in_out_come` (`IDINOUTCOME`, `idshop`, `month_year`, `JKH`, `rent`) VALUES (3, 2,
'01.01.2002', 2140, 13000);

COMMIT;

```

-----  
-- Data for table `mydb`.`Menu`  
-----

```
START TRANSACTION;
USE `mydb`;
INSERT INTO `mydb`.`Menu` (`idbluda`, `Bluda`, `Cost`) VALUES (1, 'shaverma1', 180);
INSERT INTO `mydb`.`Menu` (`idbluda`, `Bluda`, `Cost`) VALUES (2, 'shaverma2', 220);
INSERT INTO `mydb`.`Menu` (`idbluda`, `Bluda`, `Cost`) VALUES (3, 'shaverma3', 350);
INSERT INTO `mydb`.`Menu` (`idbluda`, `Bluda`, `Cost`) VALUES (4, 'shaverma4', 120);
INSERT INTO `mydb`.`Menu` (`idbluda`, `Bluda`, `Cost`) VALUES (5, 'cocacola0_5', 100);

COMMIT;
```

-----  
-- Data for table `mydb`.`Products`  
-----

```
START TRANSACTION;
USE `mydb`;
INSERT INTO `mydb`.`Products` (`Name_of_product`) VALUES ('kapusta');
INSERT INTO `mydb`.`Products` (`Name_of_product`) VALUES ('morkov');
INSERT INTO `mydb`.`Products` (`Name_of_product`) VALUES ('myaso');
INSERT INTO `mydb`.`Products` (`Name_of_product`) VALUES ('lavash');
INSERT INTO `mydb`.`Products` (`Name_of_product`) VALUES ('cocacola0_5');

COMMIT;
```

-----  
-- Data for table `mydb`.`Personal`  
-----

```
START TRANSACTION;
USE `mydb`;
INSERT INTO `mydb`.`Personal` (`idPersonal`, `idshopwork`, `FIO`, `passport`, `MedBook`, `citizenship_RF`, `position`, `telephone_number`, `salary`) VALUES (1, 1, 'MukhtarovTT', '1212555555', NULL, NULL, NULL, NULL, 10000);
INSERT INTO `mydb`.`Personal` (`idPersonal`, `idshopwork`, `FIO`, `passport`, `MedBook`, `citizenship_RF`, `position`, `telephone_number`, `salary`) VALUES (2, 1, 'BuyanovKK', '1616545454', NULL, NULL, NULL, NULL, 15000);
INSERT INTO `mydb`.`Personal` (`idPersonal`, `idshopwork`, `FIO`, `passport`, `MedBook`, `citizenship_RF`, `position`, `telephone_number`, `salary`) VALUES (3, 1, 'ArtarovKK', '1245353535', NULL, NULL, NULL, NULL, 12500);
INSERT INTO `mydb`.`Personal` (`idPersonal`, `idshopwork`, `FIO`, `passport`, `MedBook`, `citizenship_RF`, `position`, `telephone_number`, `salary`) VALUES (4, 1, 'KetchupBD', '1243567890', NULL, NULL, NULL, NULL, 13000);
INSERT INTO `mydb`.`Personal` (`idPersonal`, `idshopwork`, `FIO`, `passport`, `MedBook`, `citizenship_RF`, `position`, `telephone_number`, `salary`) VALUES (5, 2, 'PashkevichDD', '1111111111', NULL, NULL, NULL, NULL, 15000);
INSERT INTO `mydb`.`Personal` (`idPersonal`, `idshopwork`, `FIO`, `passport`, `MedBook`, `citizenship_RF`, `position`, `telephone_number`, `salary`) VALUES (6, 2, 'BarebugKR', '4755789654', NULL, NULL, NULL, NULL, 17000);
INSERT INTO `mydb`.`Personal` (`idPersonal`, `idshopwork`, `FIO`, `passport`, `MedBook`, `citizenship_RF`, `position`, `telephone_number`, `salary`) VALUES (7, 2, 'TsalVP', '7189486579', NULL, NULL, NULL, NULL, 15000);
INSERT INTO `mydb`.`Personal` (`idPersonal`, `idshopwork`, `FIO`, `passport`, `MedBook`, `citizenship_RF`, `position`, `telephone_number`, `salary`) VALUES (8, 2, 'PiratSS', '3214153426', NULL, NULL, NULL, NULL, 14000);
```

```
INSERT INTO `mydb`.`Personal` (`idPersonal`, `idshopwork`, `FIO`, `passport`, `MedBook`, `citizenship_RF`,
`position`, `telephone_number`, `salary`) VALUES (9, 2, 'TamaevAA', '9731497628', NULL, NULL, NULL,
NULL, 11000);
```

```
COMMIT;
```

```

-- Data for table `mydb`.`Suppliers`

```

```
START TRANSACTION;
```

```
USE `mydb`;
```

```
INSERT INTO `mydb`.`Suppliers` (`idPostavshika`, `phoneNumber`, `NameofCompany`) VALUES (1, NULL,
NULL);
```

```
INSERT INTO `mydb`.`Suppliers` (`idPostavshika`, `phoneNumber`, `NameofCompany`) VALUES (2, NULL,
NULL);
```

```
INSERT INTO `mydb`.`Suppliers` (`idPostavshika`, `phoneNumber`, `NameofCompany`) VALUES (3, NULL,
NULL);
```

```
INSERT INTO `mydb`.`Suppliers` (`idPostavshika`, `phoneNumber`, `NameofCompany`) VALUES (4, NULL,
NULL);
```

```
COMMIT;
```

```

-- Data for table `mydb`.`Sold_bluda`

```

```
START TRANSACTION;
```

```
USE `mydb`;
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (1, 1,
600, 1);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (2, 2,
1500, 1);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (3, 3,
1800, 1);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (4, 4,
100, 1);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (5, 5,
500, 1);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (6, 1,
380, 2);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (7, 2,
700, 2);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (8, 3,
1850, 2);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (9, 4, 70,
2);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (10, 5,
400, 2);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (11, 1,
500, 3);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (12, 2,
900, 3);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (13, 3,
2000, 3);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (14, 4,
100, 3);
```

```
INSERT INTO `mydb`.`Sold_bluda` (`idsolds`, `idbluda`, `kolvo_solded_blud`, `idinoutcome`) VALUES (15, 5,
```

400, 3);

COMMIT;

-----  
-- Data for table `mydb`.`Suppliers\_Products`  
-----

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Suppliers\_Products` (`id`, `idPostavshika`, `Name\_of\_product`, `Cost\_of\_kg\_piece`)  
VALUES (1, 1, 'morkov', 45);

INSERT INTO `mydb`.`Suppliers\_Products` (`id`, `idPostavshika`, `Name\_of\_product`, `Cost\_of\_kg\_piece`)  
VALUES (2, 1, 'kapusta', 150);

INSERT INTO `mydb`.`Suppliers\_Products` (`id`, `idPostavshika`, `Name\_of\_product`, `Cost\_of\_kg\_piece`)  
VALUES (3, 2, 'lavash', 10);

INSERT INTO `mydb`.`Suppliers\_Products` (`id`, `idPostavshika`, `Name\_of\_product`, `Cost\_of\_kg\_piece`)  
VALUES (4, 3, 'myaso', 300);

INSERT INTO `mydb`.`Suppliers\_Products` (`id`, `idPostavshika`, `Name\_of\_product`, `Cost\_of\_kg\_piece`)  
VALUES (5, 4, 'cocacola0\_5', 45);

COMMIT;

-----  
-- Data for table `mydb`.`Menu\_Products`  
-----

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (1, 'kapusta');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (1, 'morkov');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (2, 'myaso');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (2, 'kapusta');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (3, 'myaso');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (4, 'kapusta');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (1, 'lavash');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (2, 'lavash');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (3, 'lavash');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (4, 'lavash');

INSERT INTO `mydb`.`Menu\_Products` (`idbluda`, `Name\_of\_product`) VALUES (5, 'cocacola0\_5');

COMMIT;

-----  
-- Data for table `mydb`.`Bought\_product`  
-----

START TRANSACTION;

USE `mydb`;

INSERT INTO `mydb`.`Bought\_product` (`idbuys`, `idINOUTCOME`, `kolvo\_buying\_products`,  
`Suppliers\_Products\_id`) VALUES (1, 1, 300, 1);

INSERT INTO `mydb`.`Bought\_product` (`idbuys`, `idINOUTCOME`, `kolvo\_buying\_products`,  
`Suppliers\_Products\_id`) VALUES (2, 1, 150, 2);

INSERT INTO `mydb`.`Bought\_product` (`idbuys`, `idINOUTCOME`, `kolvo\_buying\_products`,  
`Suppliers\_Products\_id`) VALUES (3, 1, 4000, 3);

INSERT INTO `mydb`.`Bought\_product` (`idbuys`, `idINOUTCOME`, `kolvo\_buying\_products`,  
`Suppliers\_Products\_id`) VALUES (4, 1, 600, 4);

INSERT INTO `mydb`.`Bought\_product` (`idbuys`, `idINOUTCOME`, `kolvo\_buying\_products`,

```

`Suppliers_Products_id`) VALUES (5, 1, 500, 5);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (6, 2, 250, 1);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (7, 2, 125, 2);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (8, 2, 3000, 3);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (9, 2, 500, 4);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (10, 2, 400, 5);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (11, 3, 300, 1);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (12, 3, 140, 2);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (13, 3, 3500, 3);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (14, 3, 450, 4);
INSERT INTO `mydb`.`Bought_product` (`idbuys`, `idINOUTCOME`, `kolvo_buying_products`,
`Suppliers_Products_id`) VALUES (15, 3, 400, 5);

```