МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Пензенский государственный технологический университет»

(ПензГТУ)

Факультет информационных и образовательных технологий

Кафедра «Информационные технологии и системы»

Дисциплина «Управление информационными ресурсами»

ОТЧЕТ

по лабораторным работам

Выполнил: студент гр. 15ИС2м Терехина Е.А.

Проверил: ст. преподаватель каф. ИТС Володин К.И.

Работа защищена с оценкой: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пенза, 2016

Содержание

[Раздел 1. Создание схемы хранения данных в рамках современной клиент-серверной СУБД с применением визуальных средств проектирования 3](#_Toc452647130)

[1.1. Общие сведения 3](#_Toc452647131)

[1.2. Цель 4](#_Toc452647132)

[1.3.Задачи 4](#_Toc452647133)

[1.4. Реализация 4](#_Toc452647134)

[1.5. Выводы по разделу 11](#_Toc452647135)

[Раздел 2. Разработка и тестирование запросов на языке SQL 12](#_Toc452647136)

[2.1. Общие сведения 12](#_Toc452647137)

[2.2. Цель 12](#_Toc452647138)

[2.3. Задачи 12](#_Toc452647139)

[2.4. Реализация 12](#_Toc452647140)

[2.5. Выводы по разделу 21](#_Toc452647141)

[Раздел 3. Разработка и тестирование запросов по созданию индексов, представлений и других объектов БД на языке *SQL* 22](#_Toc452647142)

[3.1. Общие сведения 22](#_Toc452647143)

[3.2. Цель 23](#_Toc452647144)

[3.3. Задача 23](#_Toc452647145)

[3.4. Реализация 23](#_Toc452647146)

[3.5. Выводы по разделу 28](#_Toc452647147)

[Список литературы 29](#_Toc452647148)

[Приложение 1. Схема БД 30](#_Toc452647149)

[Приложение 2. Схема БД в MySQL Workbench 31](#_Toc452647150)

[Приложение 3. SQL-код создания БД 32](#_Toc452647151)

[Приложение 4. SQL-код создания и наполнения БД Northwind. 37](#_Toc452647152)

### Раздел 1. Создание схемы хранения данных в рамках современной клиент-серверной СУБД с применением визуальных средств проектирования

### 1.1. Общие сведения

Под базами данных (БД) понимаются системы хранения и обработки данных, для доступа к которым используется язык *SQL* (*Structured Query Language)*. Существует множество различных систем управления базами данных (СУБД), но в условиях хостинга как правило применяется СУБД [*MySQL*](http://www.mysql.com/). Причинами тому являются фактическая ориентация этой СУБД на хостинговые задачи, доступность на всех популярных серверных операционных системах, а также простота настройки и администрирования.

*MySQL* - это система управления базами данных.

База данных представляет собой структурированную совокупность данных. Эти данные могут быть любыми - от простого списка предстоящих покупок до перечня экспонатов картинной галереи или огромного количества информации в корпоративной сети. Для записи, выборки и обработки данных, хранящихся в компьютерной базе данных, необходима система управления базой данных, каковой и является ПО *MySQL*. Поскольку компьютеры замечательно справляются с обработкой больших объемов данных, управление базами данных играет центральную роль в вычислениях. Реализовано такое управление может быть по-разному - как в виде отдельных утилит, так и в виде кода, входящего в состав других приложений.

*MySQL* - это система управления реляционными базами данных.

В реляционной базе данных данные хранятся не все скопом, а в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. *SQL* как часть системы *MySQL* можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

### 1.2. Цель

Цель выполнения работы – изучить и получить навыки работы с объектами и элементами языка *MySQL*. Создать схему хранения данных в *MySQL Workbench*

### 1.3.Задачи

* Изучить основные типы данных в БД;
* Изучить структуры БД;
* Создать схему хранения данных.

### 1.4. Реализация

На основе схемы БД (см. Приложение 1), необходимо визуальными средствами программирования создать однотипные таблицы с соответствующими полями и наименованиями в программе *MySQL Workbench*. Для этого необходимо выполнить следующие действия:

Первоначально необходимо создать Модель БД. Для добавления модели необходимо кликнуть значок плюса рядом с заголовком "*Model*s" или выбрать команду меню "File → New Model".

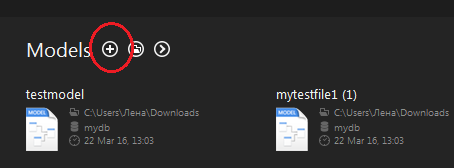


Рисунок 1 − Форма создания модели.

В открывшемся окне необходимо ввести имя базы данных, выбрать кодировку и при необходимости заполнить поле комментария.

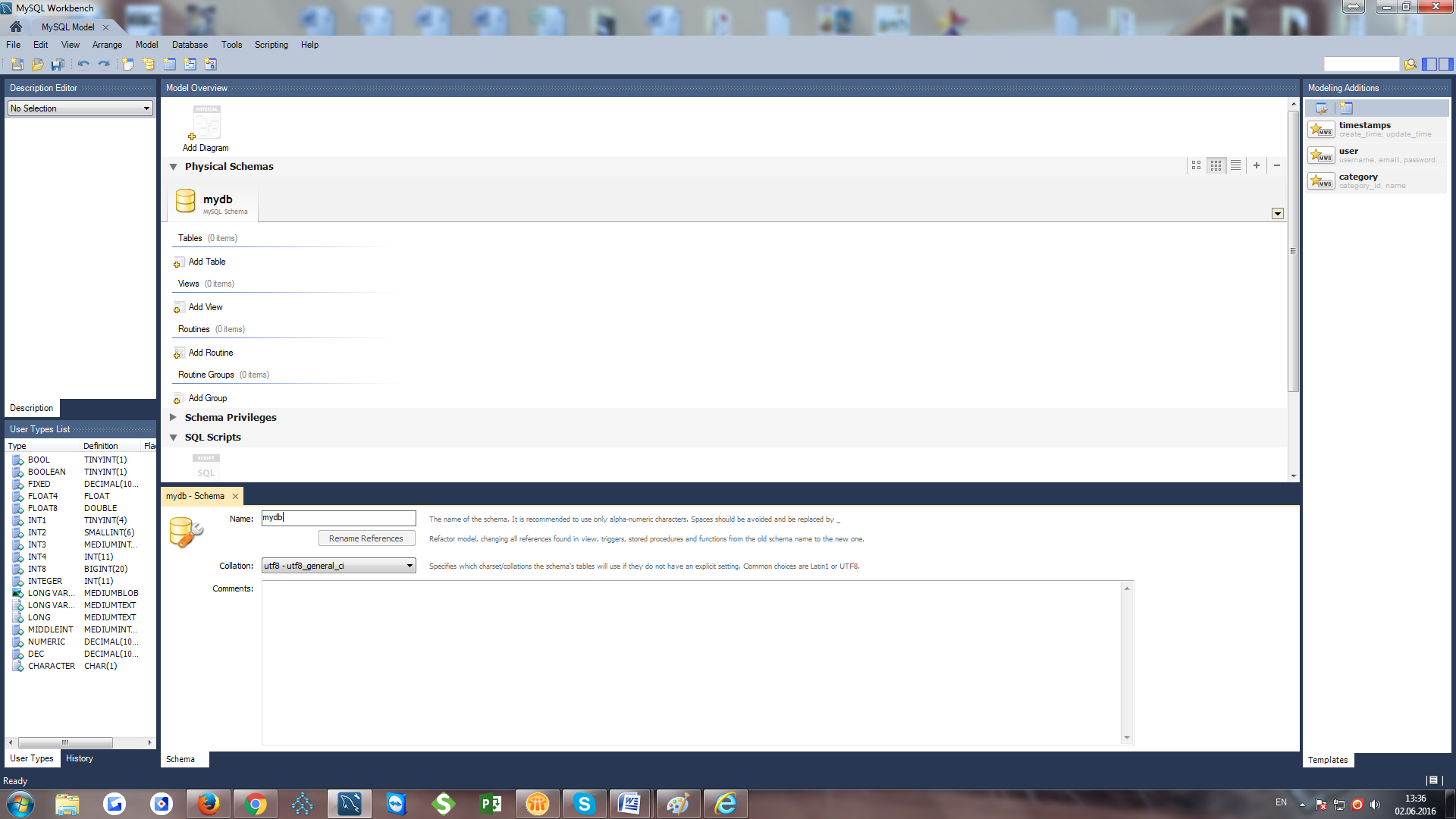


Рисунок 2 − Вкладка создания БД.

Для создания новой таблицы на вкладке «*Physical Schemas*» необходимо кликнуть «*Add Table*» и заполнить поля — название таблицы, поля таблицы, их типы и длину.

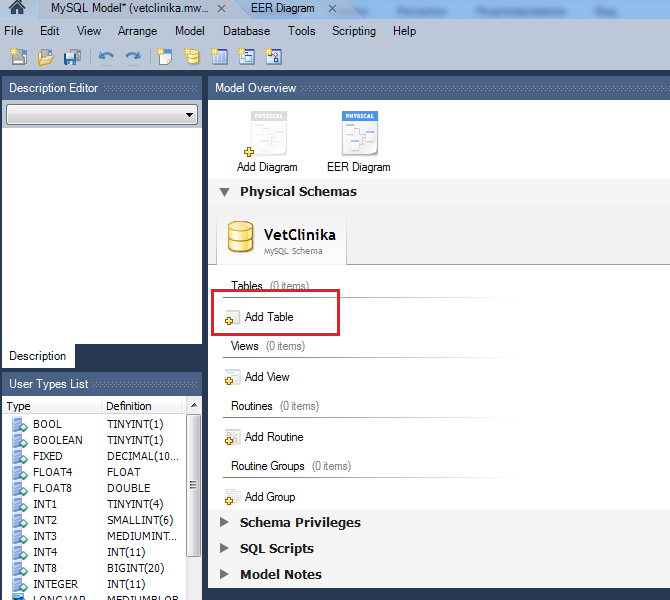


Рисунок 3 − Кнопка создания таблицы.

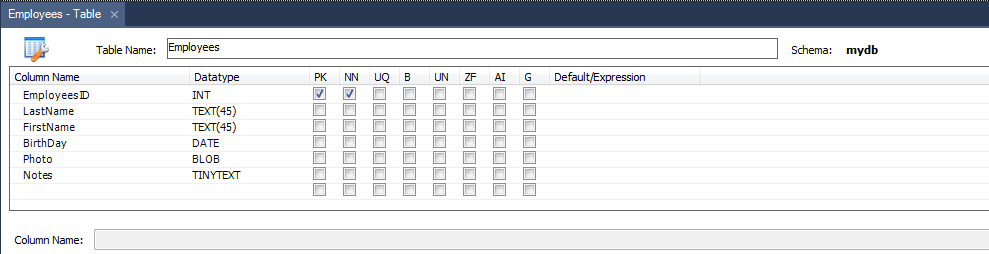


Рисунок 4 − Создание таблицы.

Все созданные таблицы отображаются в разделе «*Tables*»:

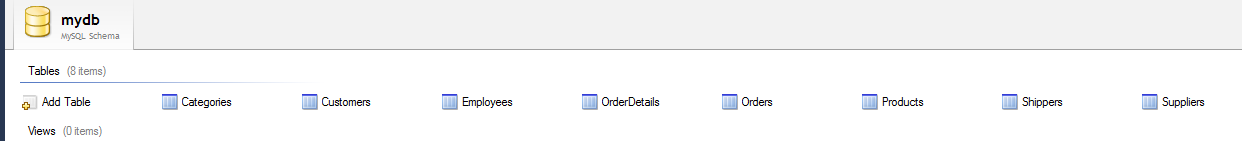


Рисунок 5 − Созданные таблицы.

Базы данных содержат таблицы с различными данными. Каждый столбец данных имеет свой тип в зависимости от информации, которую хранит. В языке *MySQL* типы данных делятся на числовые, строковые и типы данных даты и времени. Например:

* *CHARACTER(n)* или *CHAR(n)* — символьные строки постоянной длины в п символов. При задании данного типа под каждое значение всегда отводится п символов, и если реальное значение занимает менее, чем п символов, то СУБД автоматически дополняет недостающие символы пробелами.
* *NUMERIC[(n,m)]* — точные числа, здесь n — общее количество цифр в числе, m — количество цифр слева от десятичной точки.
* *DECIMAL[(n,m)]* — точные числа, здесь n — общее количество цифр в числе, m — количество цифр слева от десятичной точки.
* *DEC[(n,m)]* - то же, что и *DECIMAL[(n,m)].*
* *INTEGER* или *INT* — целые числа.
* *SMALLINT* — целые числа меньшего диапазона.
* *FLOAT[(n)]* — числа большой точности, хранимые в форме с плавающей точкой. Здесь n — число байтов, резервируемое под хранение одного числа. Диапазон чисел определяется конкретной реализацией.
* *REAL* — вещественный тип чисел, который соответствует числам с плавающей точкой, меньшей точности, чем FLOAT.
* *DOUBLE PRECISION* специфицирует тип данных с определенной в реализации точностью большей, чем определенная в реализации точность для *REAL*.
* *VARCHAR(n)* — строки символов переменной длины.
* *NCHAR(N)* — строки локализованных символов постоянной длины.
* *NCHAR VARYING(n)* — строки локализованных символов переменной длины.
* *BIT(n)* — строка битов постоянной длины.
* *BIT VARYING(n)* — строка битов переменной длины.
* *DATE* — календарная дата.
* *ТIMESТАМР*(точность) — дата и время.
* *INTERVAL* — временной интервал.
* *BLOB, TEXT- BLOB*или *ТЕХТ* с максимальной длиной 65535 (2^16 - 1) символов.

Присвоение типов данных каждой строке БД на примере таблицы *«Employees»* представлено в таблице 1.

Таблица 1− Типы данных таблицы *«Employees».*

|  |  |  |
| --- | --- | --- |
| Строка в таблице | Тип данных | Характеристика |
| EmployeeID | INT | Присвоение определенного номера, поле является ключевым |
| LastName | TEXT(45) | Будет использоваться для введения фамилий клиентов |
| FirstName | TEXT(45) | Будет использоваться для введения имён клиентов |
| BirthDate | DATE | Поле для введения даты рождения клиента |
| Photo | VARCHAR(255) | Путь к фотографиям будет через текстовые значения 9ссылки) |
| Notes | TEXT (255) | Поле для введения дополнительных заметок |

В программе *MySQL Workbench* процесс присвоения типов данных происходит следующим образом: в колонке «*Datatype*» каждой строке выбирается тип данных и необходимый размер строки.

Каждая таблица имеет связь с другой по определенным данным. Связи создаются через вкладку «*Foreign Keys*» у необходимой таблицы, либо непосредственно при рисовании диаграммы базы данных.

Для создания связи таблиц через вкладку «*Foreign Keys*» необходимо перейти на эту вкладку, ввести название ключа и выбрать таблицу для связи. Дальше в правой части формы, где указана таблица с полями у нужного поля выбирается поле указанной ранее таблицы, т.е. указывается связь между полями.

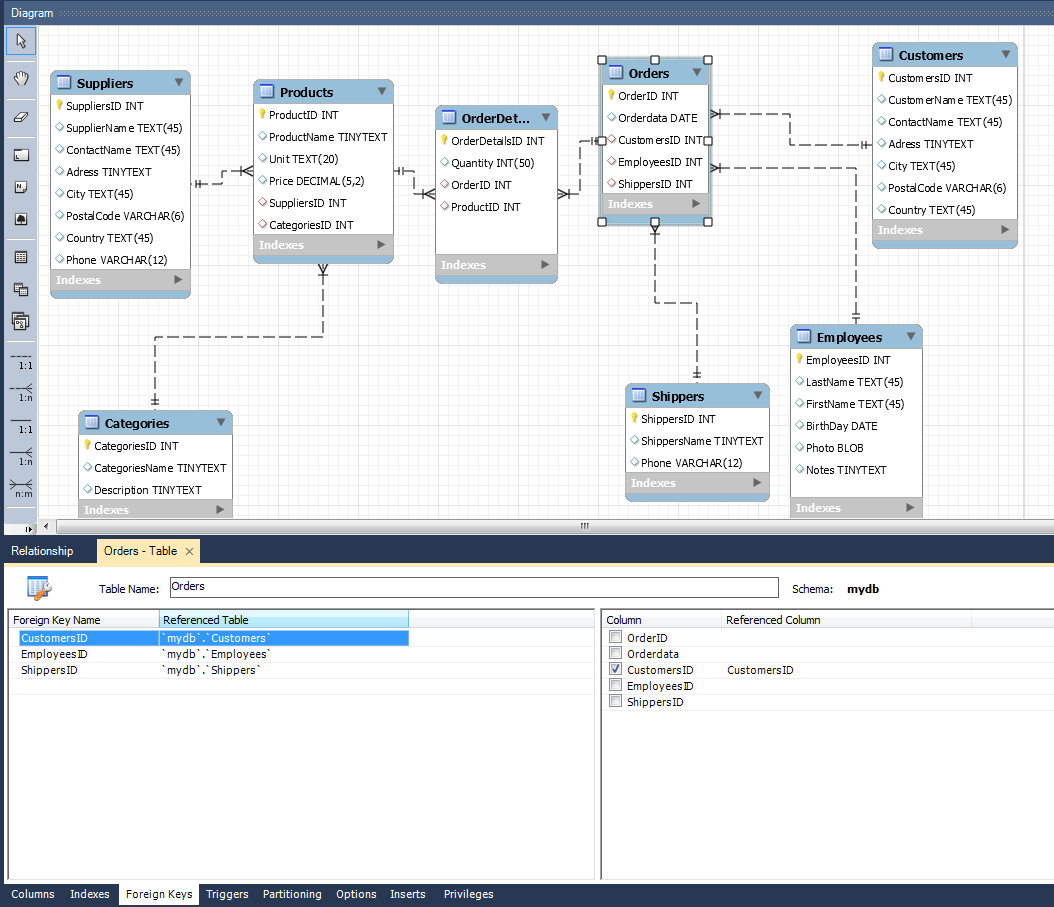


Рисунок 6 - Установление связей между таблицами.

В результате проделанной работы была сделана схема БД с такой же структурой и таблицами, как в исходном примере. Полученная схема данных представлена в приложении 2.

После того как таблица была выполнена, сгенерируем код *SQL,* необходимый для дальнейшей работы. Для этого выполним следующие действия:

Нажимаем кнопку *«Forward Engineer to Database».*

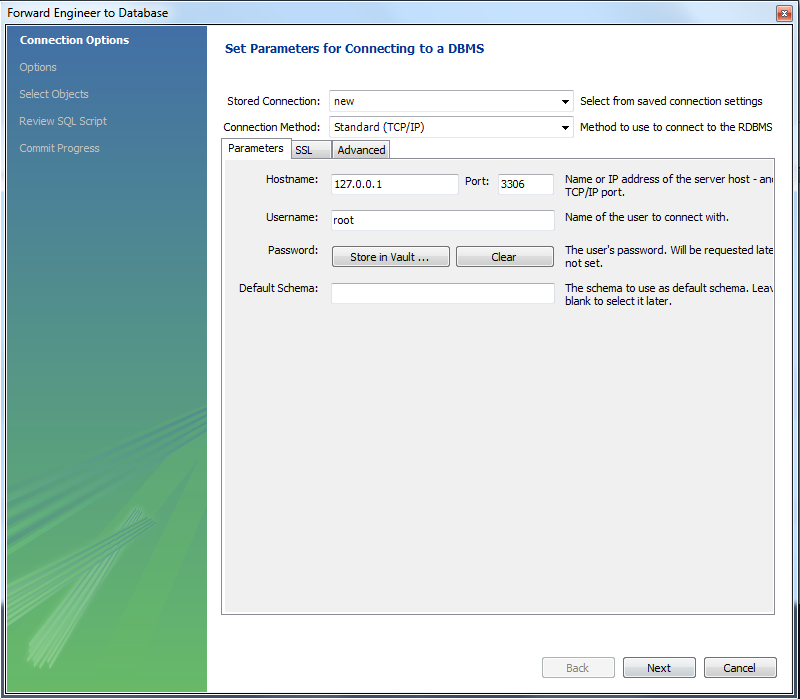


Рисунок 7 – Открытие кода *SQL* базы данных.

Выбираем необходимые параметры и нажимаем *«Next».*

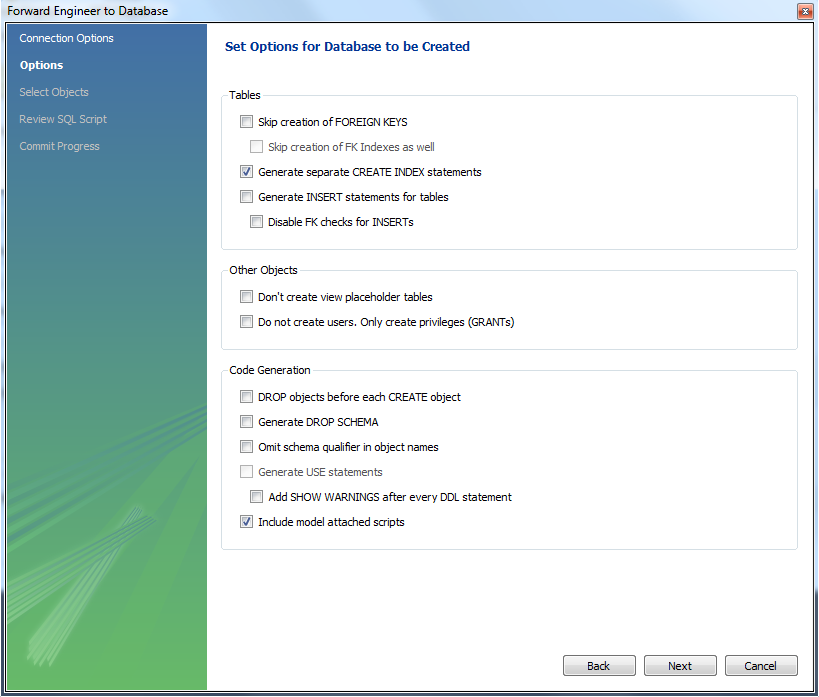


Рисунок 8 – Открытие кода *SQL* базы данных.

Система сообщает, что будет экспортировано 8 таблиц. Нажимаем *«Next».*

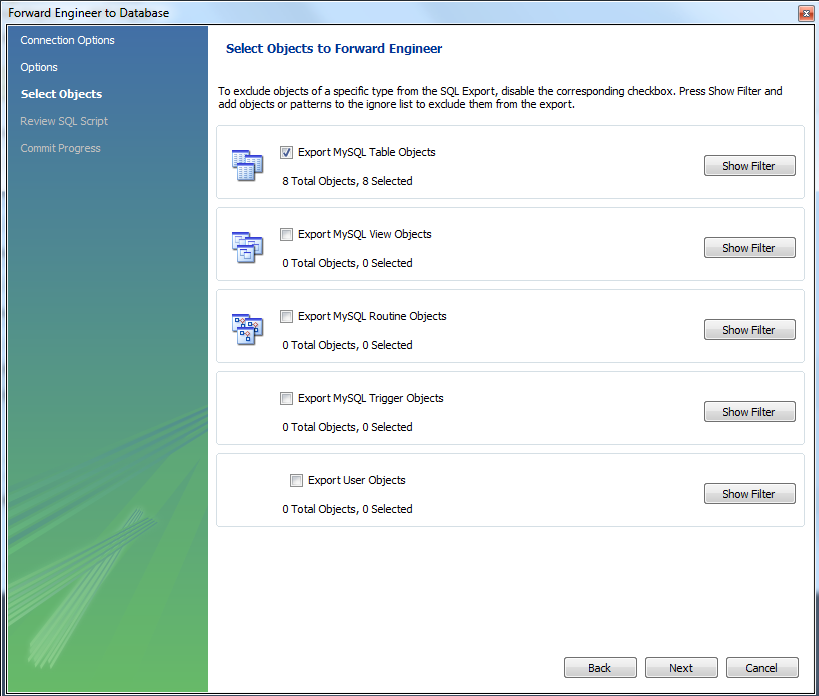


Рисунок 9 – Список экспортируемых объектов.

В итоге выгружается код базы данных (полный код представлен в Приложении 3). Данный код можно копировать и сохранить в формате *sql*.

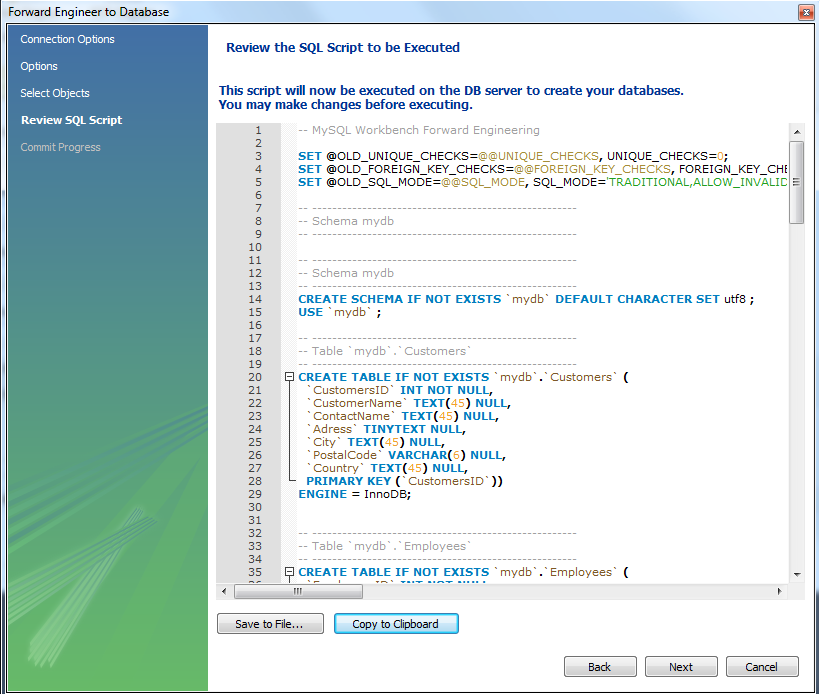


Рисунок 10 – Код *SQL* базы данных.

### 1.5. Выводы по разделу

В результате проделанной работы, была разработана схема БД в программе *MySQL Workbench*, соответствующая БД исходного изображения (представленного в Приложении 1). Строкам таблицы присвоены имена, типы данных, размер. Установлены связи между таблицами. Сгенерирован код *SQL* базы данных (Приложение 3).

### Раздел 2. Разработка и тестирование запросов на языке SQL

### 2.1. Общие сведения

*MySQL* запрос – это обращение к базе данных MySQL, с помощью которого мы можем реализовать: получение, изменение, удаление, сортировку, добавление, и другие манипуляции с данными базы.

Все *MySQL* запросы поделены на простые и сложные запросы. Нами также выделена категория очень простых запросов, структура которых очень понятна и не требует дополнительных разъяснений.

[Простые *MySQL* запросы](http://sitear.ru/material/mysql-zaprosy#mysql1) – запросы в которых участвует одна таблица базы данных.

[Сложные *MySQL* запросы](http://sitear.ru/material/mysql-zaprosy#mysql2) – запросы в которых могут участвовать две и более таблиц БД.

### 2.2. Цель

Цель работы – ознакомиться и получить практические навыки работы с кодом *SQL* базы данных на примере различных видов запросов.

### 2.3. Задачи

* Доработать полученный *SQL* код базы данных;
* Научиться выполнять SQLзапросы.

### 2.4. Реализация

Для выполнения работы необходимо открыть окно *MySQL Workbench для ввода SQL-кода*.

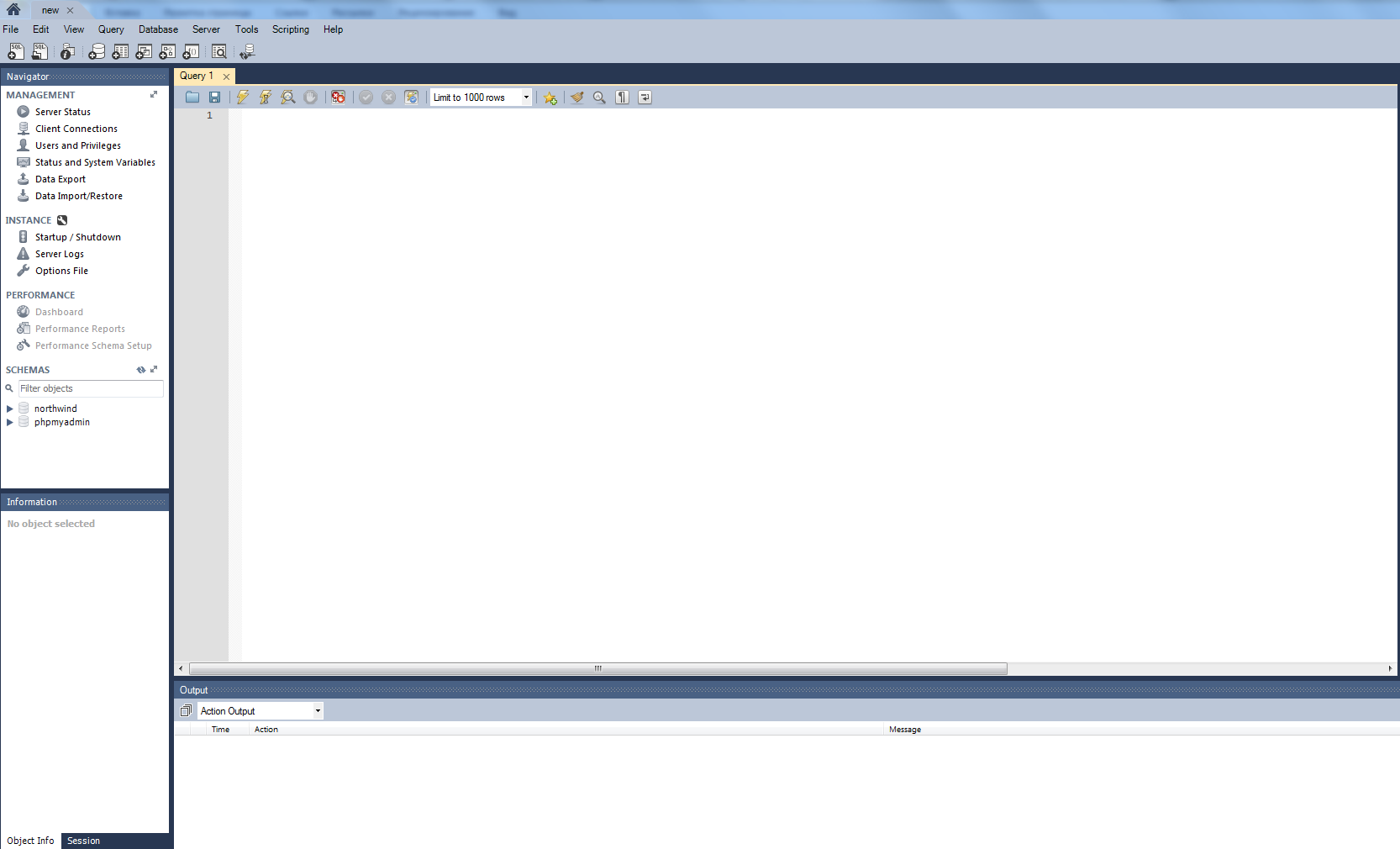


Рисунок 11 – Пустая база данных.

Вставим в вкладку *Query* код *SQL* (Представлен в приложении 4)

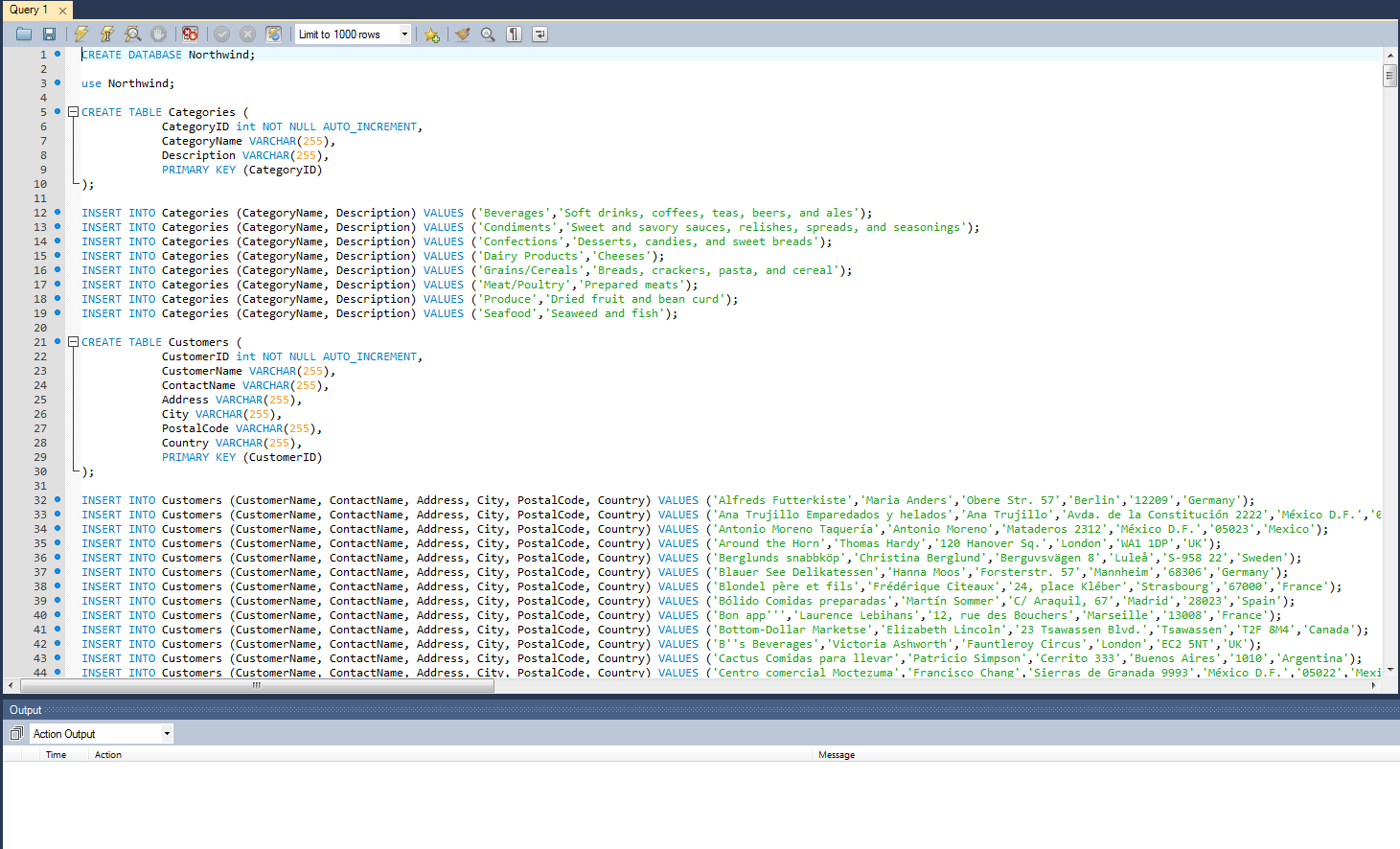


Рисунок 12 – Код базы данных *SQL* в программе *MySQL Workbench.*

Для создания БД необходимо запустить код. В нижней области экрана отобразится информация об успешном создании таблиц и заполнении их данными.

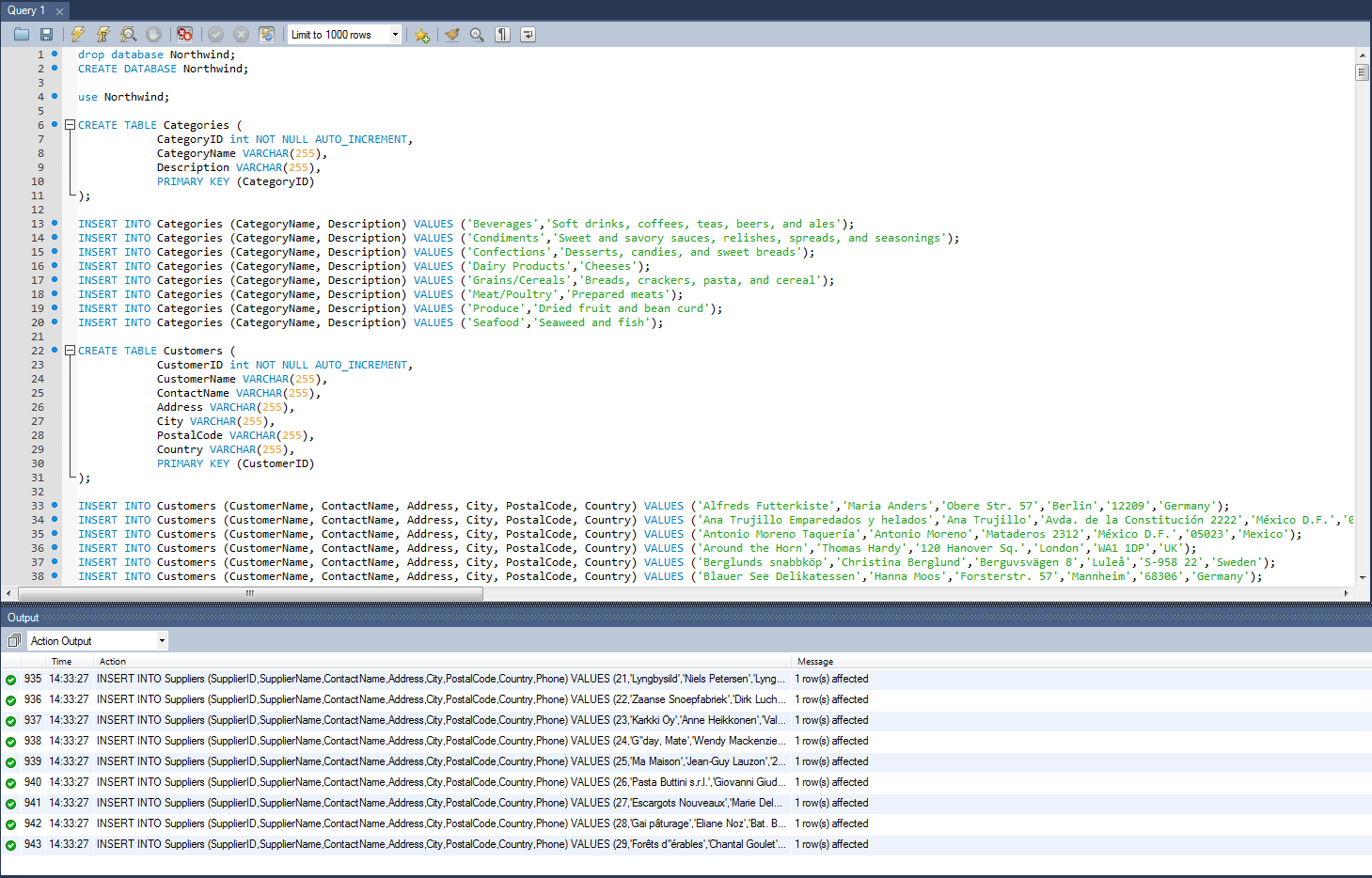


Рисунок 13 – Запущенный код базы данных *SQL* в программе *MySQL Workbench.*

Выполним некоторые виды запросов. Есть четыре основных типа запросов данных в *SQL:*

* *SELECT* – выбор строк из таблиц;
* *INSERT* – добавление строк в таблицу;
* *UPDATE* – изменение строк в таблице;
* *DELETE* – удаление строк из таблицы;

Для того, чтобы выбрать необходимые данные из таблицы, например вывести данные о поставщиках, нам нужно ввести следующий текст запроса: *SELECT \* from Suppliers.*

Текст запроса :

use Northwind;

SELECT \* from Suppliers;

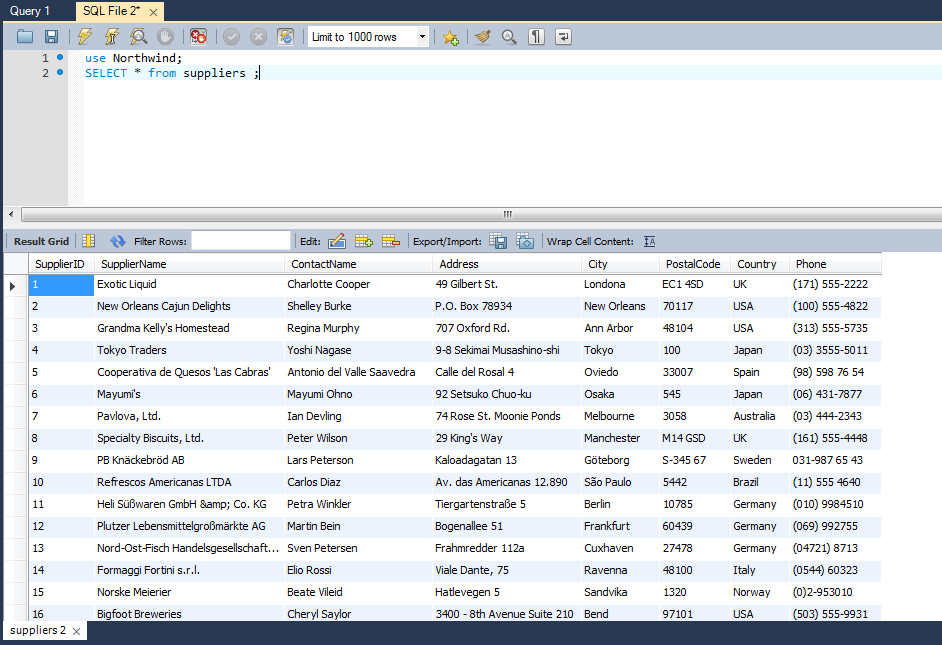


Рисунок 14 – Запрос на выбор данных о поставщиках.

Выберем из таблицы *Suppliers*, клиентов из Токио. Для этого введем следующий текст запроса:

use Northwind;

SELECT \* from suppliers

where suppliers.City like 'Tokyo';

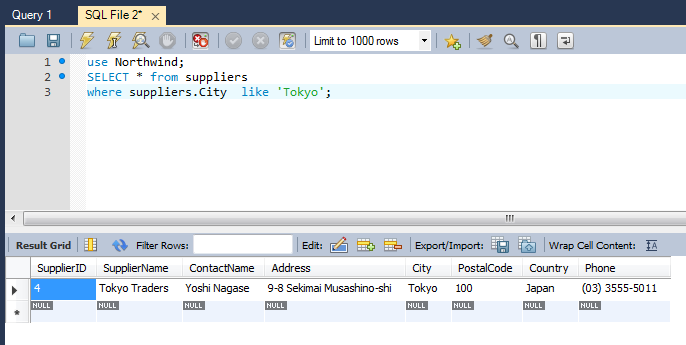


Рисунок 15 – Использование функции *Where и Like.*

Увеличим выборку. Выберем из таблицы Поставщики, клиентов из Токио и Лондона. Для этого введем следующий текст запроса:

use Northwind;

SELECT \* from suppliers

where suppliers.City in ( 'Tokyo', 'London');

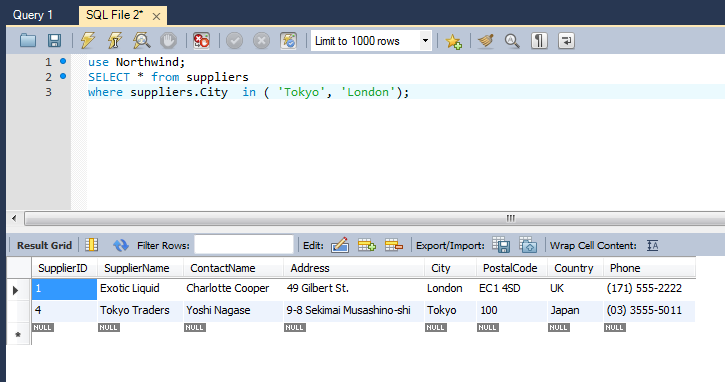


Рисунок 16 – Использование функции *in*

Далее выберем данные только о тех поставщиках, в ячейке «Город» которых стоит значение, не оканчивающееся на *«land».*

use Northwind;

SELECT \* from suppliers

where suppliers.City not like '%land';

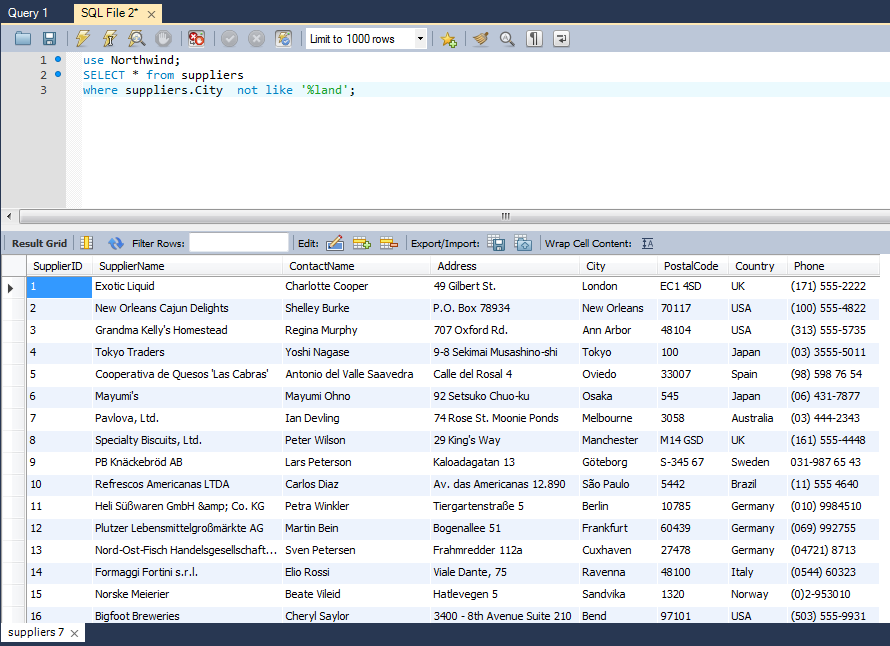


Рисунок 17 – Использование функции *not like*

Теперь выберем из таблицы данные только о тех поставщиках, значение *ID* которых меньше 30.

Далее сортируем полученные значения по контактному имени в обратном алфавитному порядке.

use Northwind;

SELECT \* from suppliers

where suppliers.SupplierID < 30 Order by SupplierName DESC ;

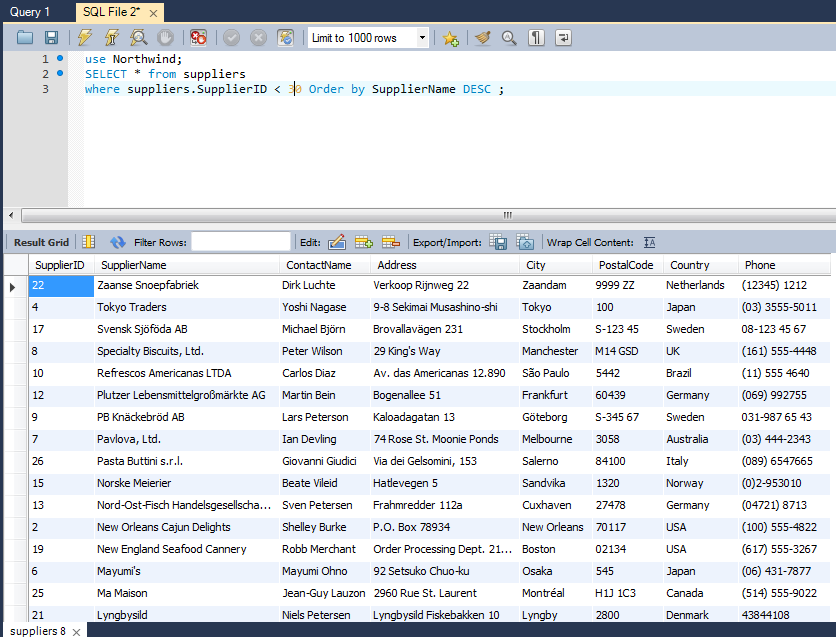


Рисунок 18 – Использование функции *Order by*

Также можно выполнить запрос выборки продуктов по цене равной 10 и 12, используя функцию between.

use Northwind;

SELECT \* from Products where Products.Price between 10 and 12;

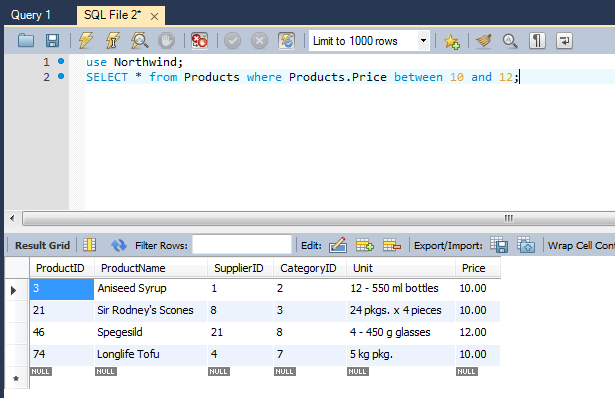


Рисунок 19 – Использование функции *Between*

При помощи функции *Limit,* отобразим первых пять покупателей:

use Northwind;

SELECT \* from Customers Limit 5;

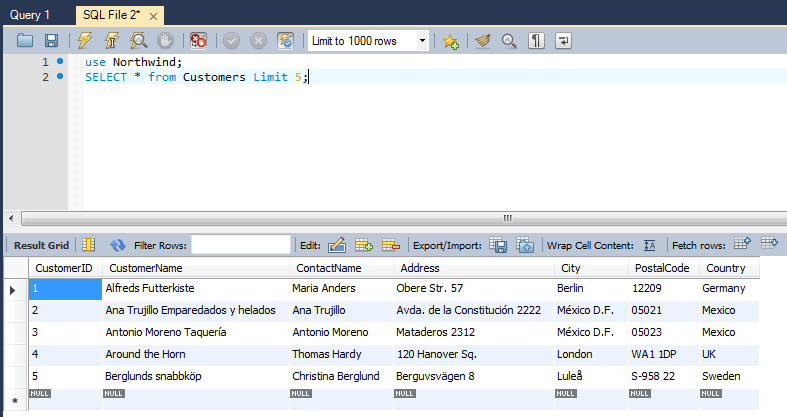


Рисунок 20 – Отображение первых 5 покупателей

Выберем в таблице Продукты максимальную цену продукта:

use Northwind;

SELECT max(Price) from Products;

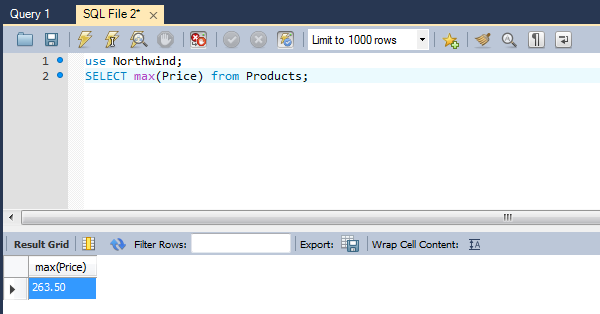


Рисунок 21 – Использование функции *MAX*

Следующий тип запросов − *INSERT* (добавить строки в таблицу).

Добавим в таблицу Employees новые строки:

use Northwind;

Insert into Employees (LastName, FirstName, BirthDate, Notes) VALUES ('Petrov', 'Victor', '1968-12-15', 'Test Employee 1');

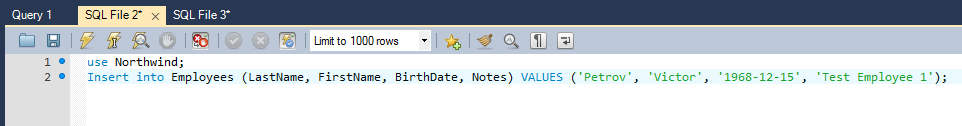


Рисунок 22 – Использование функции *Values*

Отобразим добавленную запись запросом Select:

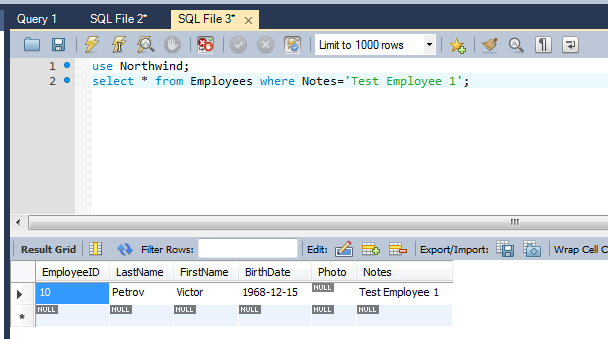


Рисунок 23 – Результат добавления новой строки в таблицу

*DELETE* – удалить строки в таблице, это следующий тип основных запросов.

В таблице Customers удалим первую строку:

use Northwind;

delete from customers where CustomerID = 1;

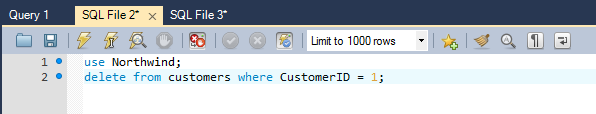


Рисунок 24 – Использование функции *Delete*

Запись удалена из таблицы:

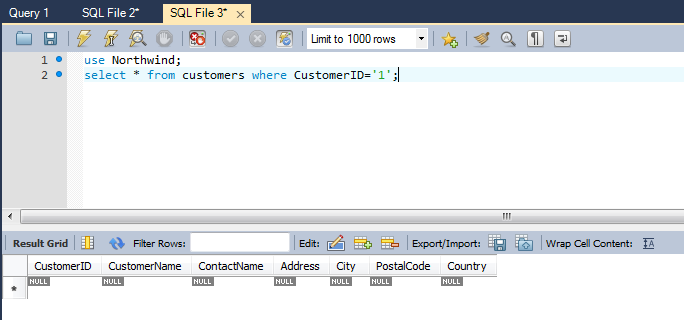


Рисунок 25 – Результат удаления строки

И последний тип запросов *UPDATE* – изменить строки в таблице.

Заменим значение *Employees.Notes в третьей строке* на *Test Note*.

use Northwind;

update employees set Employees.Notes= 'Test Note' where EmployeeID=3 ;

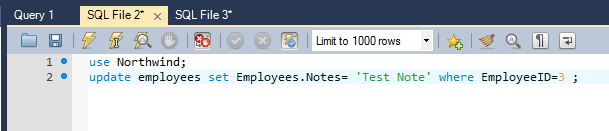


Рисунок 26 – Использование функции *Set*

Запрос выполнен:

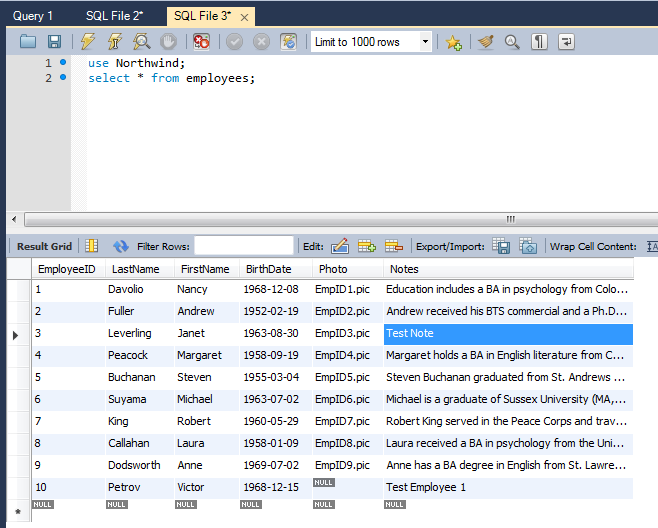


Рисунок 27 – Результат изменения строки

### 2.5. Выводы по разделу

В результате выполнения работы, был доработан код *SQL* базы данных, в который были добавлены данные о клиентах, товарах, поставщиках и тд. на основе БД протестированы основные типы запросов: *SELECT, INCERT, DELETE, UPDATE. в* каждом из которых использовались различные функции, благодаря которым отсортировывалась необходимая информация.

### Раздел 3. Разработка и тестирование запросов по созданию индексов, представлений и других объектов БД на языке *SQL*

### 3.1. Общие сведения

Одним из важнейших путей достижения высокой производительности *SQL Server* является использование индексов. Индекс ускоряет процесс запроса, предоставляя быстрый доступ к строкам данных в таблице, аналогично тому, как указатель в книге помогает быстро найти необходимую информацию. Индексы создаются для столбцов таблиц и представлений. Индексы предоставляют путь для быстрого поиска данных на основе значений в этих столбцах. Индексы применяются для быстрого поиска строк с указанным значением одного столбца. Без индекса чтение таблицы осуществляется по всей таблице начиная с первой записи, пока не будут найдены соответствующие строки. Чем больше таблица, тем больше накладные расходы. Если же таблица содержит индекс по рассматриваемым столбцам, то *MySQL* может быстро определить позицию для поиска в середине файла данных без просмотра всех данных. Для таблицы, содержащей 1000 строк, это будет как минимум в 100 раз быстрее по сравнению с последовательным перебором всех записей. Однако в случае, когда необходим доступ почти ко всем 1000 строкам, быстрее будет последовательное чтение, так как при этом не требуется операций поиска по диску.

Все индексы *MySQL (PRIMARY, UNIQUE, и INDEX)* хранятся в виде B-деревьев. Строки автоматически сжимаются с удалением пробелов в префиксах и оконечных пробелов

Индексы используются для того, чтобы:

* Быстро найти строки, соответствующие выражению *WHERE*.
* Извлечь строки из других таблиц при выполнении объединений.
* Найти величины *MAX*() или *MIN*() для заданного индексированного столбца. Эта операция оптимизируется препроцессором, который проверяет, не используете ли вы *WHERE key\_part\_4* = константа, по всем частям составного ключа< N. В этом случае *MySQL* сделает один просмотр ключа и заменит выражение константой MIN(). Если все выражения заменяются константой, запрос моментально вернет результат:

SELECT MIN(key\_part2),MAX(key\_part2) FROM table\_name where key\_part1=10

* Производить сортировку или группирование в таблице, если эти операции делаются на крайнем слева префиксе используемого ключа (например *ORDER BY key\_part\_1,key\_part\_2*). Если за всеми частями ключа следует *DESC*, то данный ключ читается в обратном порядке
* В некоторых случаях запрос можно оптимизировать для извлечения величин без обращения к файлу данных. Если все используемые столбцы в некоторой таблице являются числовыми и образуют крайний слева префикс для некоторого ключа, то чтобы обеспечить большую скорость, искомые величины могут быть извлечены непосредственно из индексного дерева:

SELECT key\_part3 FROM table\_name WHERE key\_part1=1

### 3.2. Цель

Целью работы – разработать и протестировать запросы по созданию индексов, представлений и других объектов БД на языке *SQL.*

### 3.3. Задача

* Разработать индексы БД на языке *SQL;*
* Разработать представления БД на языке *SQL;*
* Тестирование разработанных индексов и представлений БД на языке *SQL.*

### 3.4. Реализация

Сделаем выборку из таблицы *Products*, отобразив все её категории:

SELECT \* From Products, Categories;

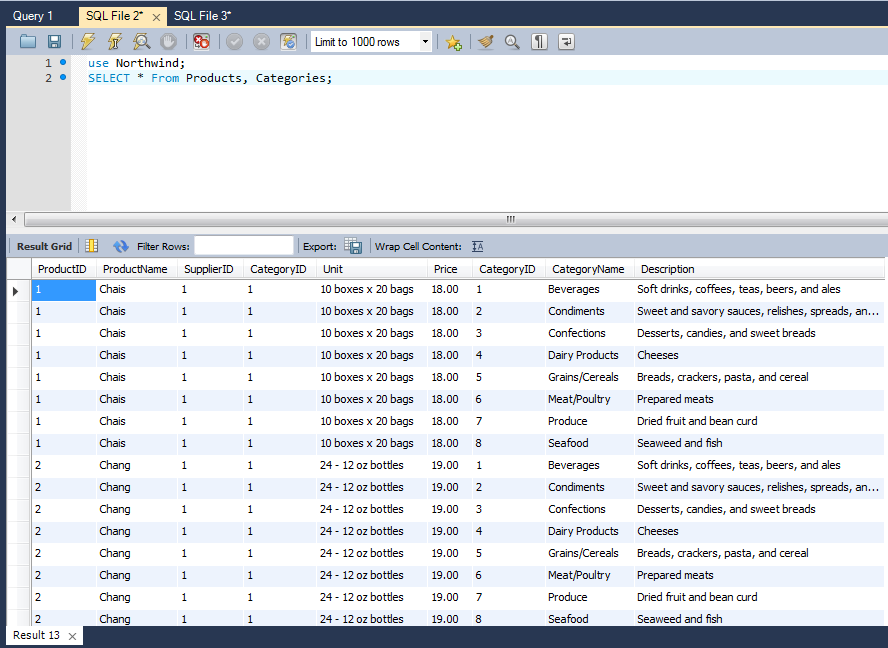


Рисунок 28 – Отображение таблицы *«Products»*

Отобразим конкретные данные из таблицы *«Products»*:

use Northwind;

SELECT Products.ProductID,

Products.ProductName,

Products.Price,

Categories.CategoryName

from Products, Categories where Products.CategoryID=Categories.CategoryID;

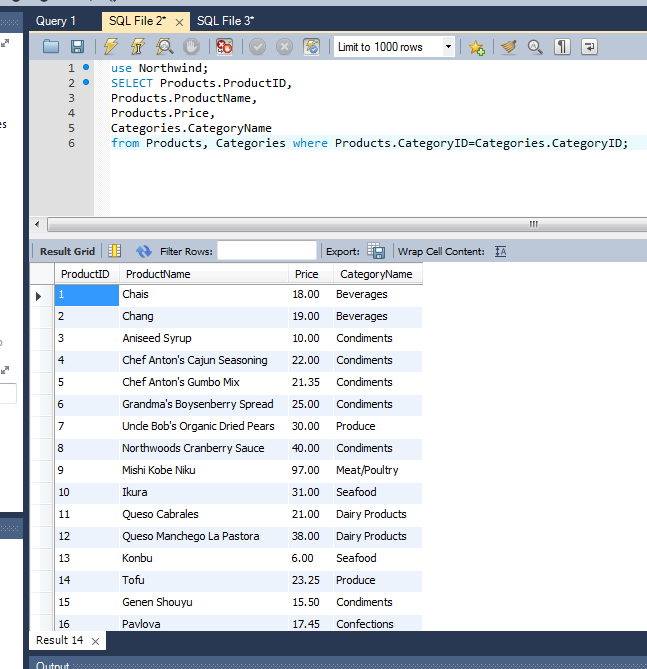


Рисунок 29 – Отображение данных таблицы *«Products»*

Сделаем выборку продуктов цена которых больше 25:

use Northwind;

SELECT Products.ProductID,

Products.ProductName,

Products.Price,

Categories.CategoryName as 'Category'

from Products, Categories where Products.CategoryID=Categories.CategoryID and (Products.Price>25);

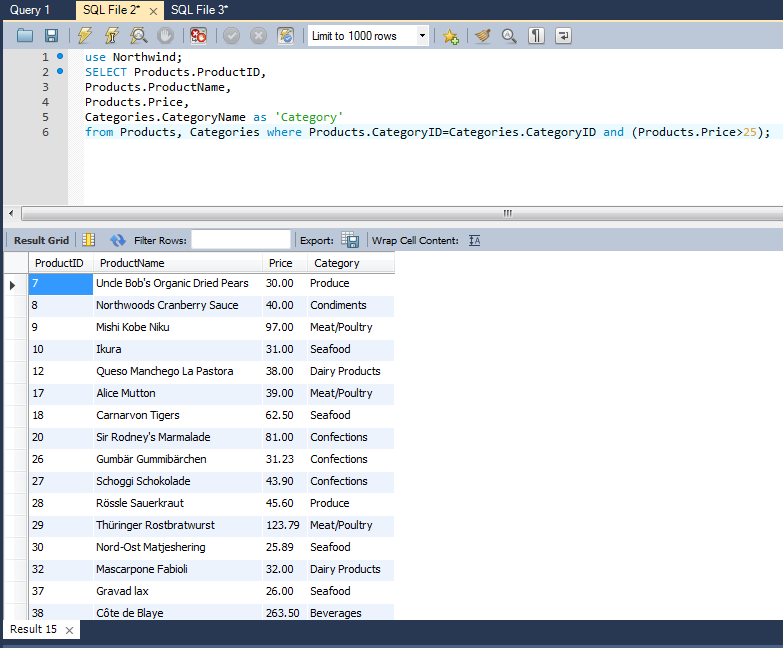


Рисунок 30 – Выборка товаров по цене

В БД *Northwind* используем представление для того, чтобы при дальнейших запросах можно было нагляднее представить хранящуюся информацию:

Create View Northwind.AVGProductPrice as select Products.ProductID, Products.ProductName, Products.Price, Categories.CategoryName as ‘Category’ From Products. Categories where (Products.CategoryID = Categories.CategoryID) and (Products.Price > (select min(Products.Price) from Products));

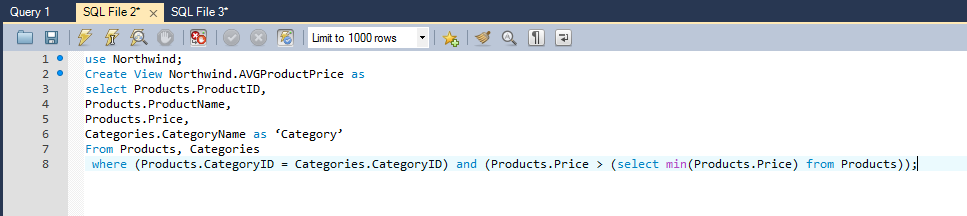


Рисунок 31 – Использование представления

Отобразим данные, которые будут использоваться в представлении:

Select \* From AVGProductPrice;

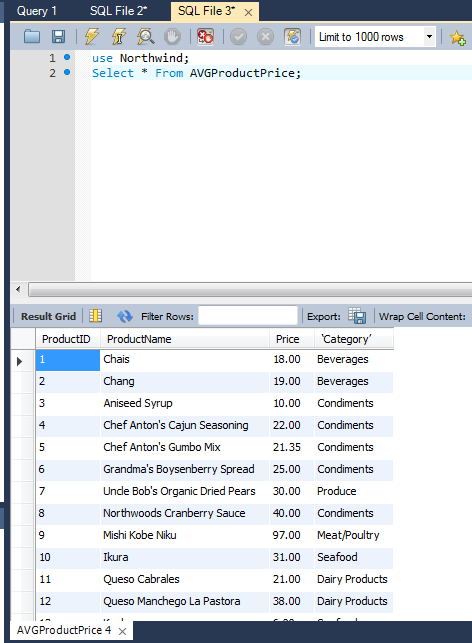


Рисунок 32 – Отображение данных таблицы *AVGProductPrice*

Выполним несколько запросов. В таблице *AVGProductPrice ds* с столбце *Category* выберем те, которые начинаются на *‘Pro’:*

use Northwind;

SELECT \* from AVGProductPrice

where avgproductprice.‘Category’ like 'Pro%';

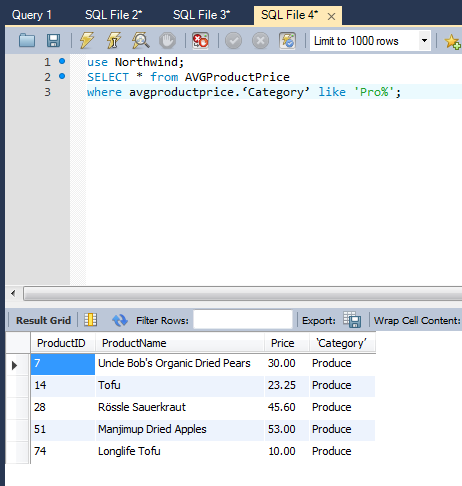


Рисунок 33 – Использование функции *Like*

Попробуем ввести сразу несколько строк запросов, содержащих разные операторы:

use Northwind;

Delimiter //

create procedure country\_new (IN con varchar (255))

Begin select \* from Customers where Customers.Country = con;

end //

Delimiter ;

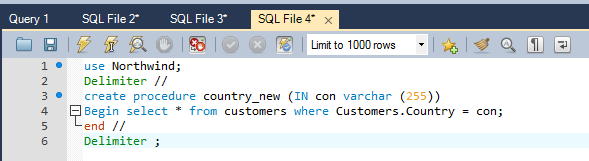


Рисунок 34 – Использование команды *DELIMETER*

### 3.5. Выводы по разделу

В ходе выполнения работы были рассмотрены представления и индексы, а в частности воспроизведены основные действия по их введению в БД, и тестирование в них различных запросов. Также при помощи команды DELIMITER стало возможным посылать строки запросов, содержащие разделитель операторов.

### Список литературы

1. MySQL.Самый популярный в мире базы данных с открытым исходным кодом [Электронный ресурс]. − Режим доступа: [www.mysql.com](http://www.mysql.com/), свободный (23.04.2016).
2. Володин К.А. Репозиторий Управление информационными ресурсами: Материалы курса [Электронный ресурс]. − Режим доступа: <https://github.com/volodink/irm-course> , свободный (23.04.2016).
3. Форум SQL Электронный ресурс]. − Режим доступа: [www.sql.ru](http://www.sql.ru/), свободный (23.04.2016).
4. «Хабрахабр»- ресурс для IT-специалистов [Электронный ресурс] . – Режим доступа:  <https://habrahabr.ru>, свободный (23.04.2016).

### Приложение 1. Схема БД

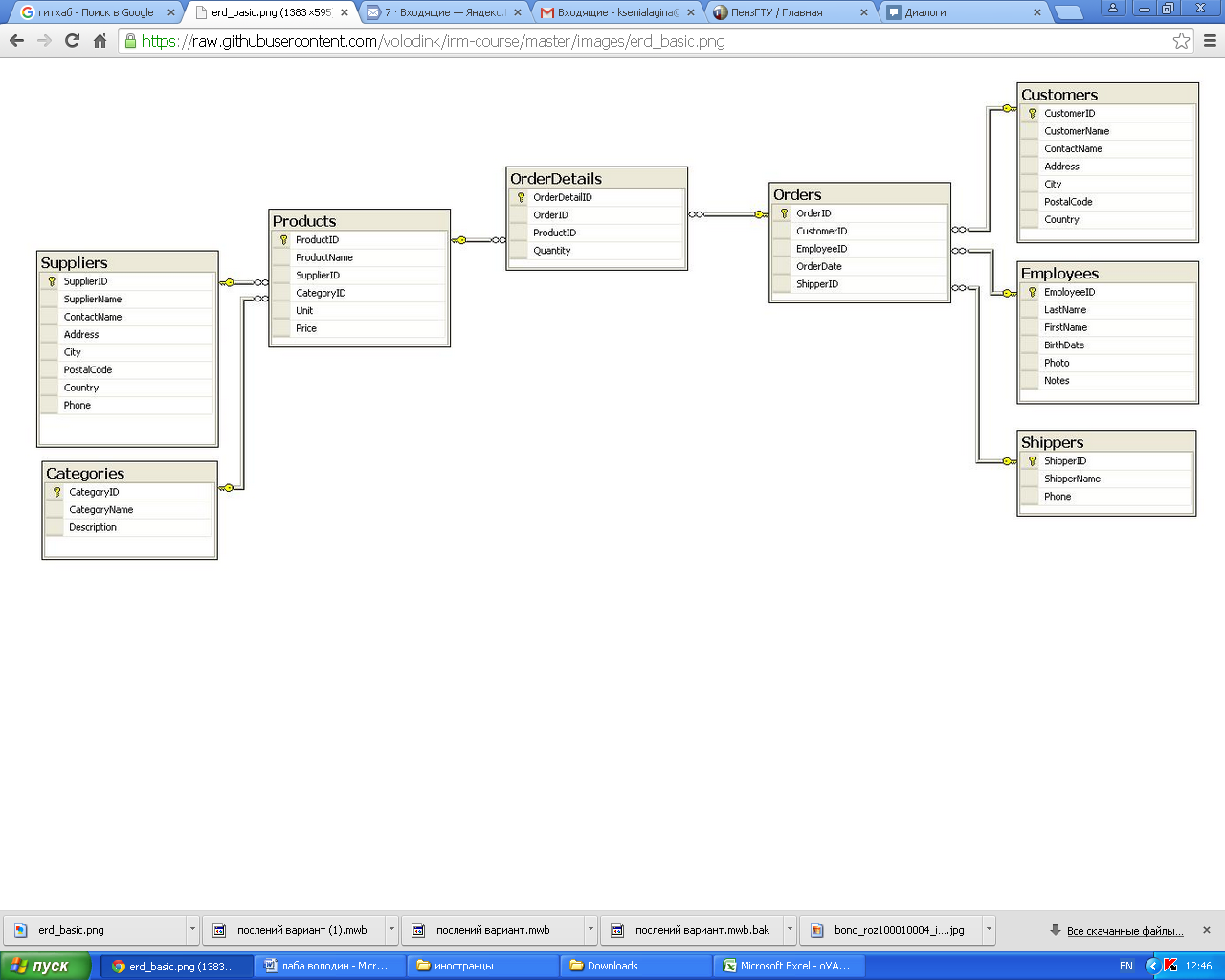


Рисунок 35 – Схема БД

### Приложение 2. Схема БД в MySQL Workbench

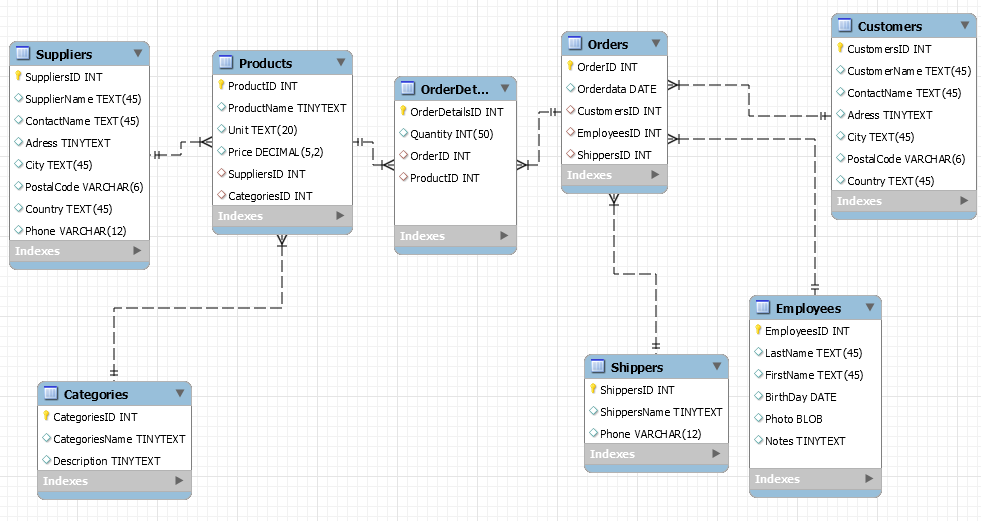


Рисунок 36 – Схема БД в *MySQL Workbench*

### Приложение 3. SQL-код создания БД

-- 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='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema mydb

-------------------------------------------------------

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

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

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`Customers`

-- -----------------------------------------------------

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

`CustomersID` INT NOT NULL,

`CustomerName` TEXT(45) NULL,

`ContactName` TEXT(45) NULL,

`Adress` TINYTEXT NULL,

`City` TEXT(45) NULL,

`PostalCode` VARCHAR(6) NULL,

`Country` TEXT(45) NULL,

PRIMARY KEY (`CustomersID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Employees`

-- -----------------------------------------------------

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

`EmployeesID` INT NOT NULL,

`LastName` TEXT(45) NULL,

`FirstName` TEXT(45) NULL,

`BirthDay` DATE NULL,

`Photo` BLOB NULL,

`Notes` TINYTEXT NULL,

PRIMARY KEY (`EmployeesID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Shippers`

-- -----------------------------------------------------

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

`ShippersID` INT NOT NULL,

`ShippersName` TINYTEXT NULL,

`Phone` VARCHAR(12) NULL,

PRIMARY KEY (`ShippersID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Orders`

-- -----------------------------------------------------

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

`OrderID` INT NOT NULL,

`Orderdata` DATE NULL,

`CustomersID` INT NULL,

`EmployeesID` INT NULL,

`ShippersID` INT NULL,

PRIMARY KEY (`OrderID`),

CONSTRAINT `CustomersID`

FOREIGN KEY (`CustomersID`)

REFERENCES `mydb`.`Customers` (`CustomersID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `EmployeesID`

FOREIGN KEY (`EmployeesID`)

REFERENCES `mydb`.`Employees` (`EmployeesID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `ShippersID`

FOREIGN KEY (`ShippersID`)

REFERENCES `mydb`.`Shippers` (`ShippersID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `CustomersID\_idx` ON `mydb`.`Orders` (`CustomersID` ASC);

CREATE INDEX `EmployeesID\_idx` ON `mydb`.`Orders` (`EmployeesID` ASC);

CREATE INDEX `ShippersID\_idx` ON `mydb`.`Orders` (`ShippersID` ASC);

-- -----------------------------------------------------

-- Table `mydb`.`Suppliers`

-- -----------------------------------------------------

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

`SuppliersID` INT NULL,

`SupplierName` TEXT(45) NULL,

`ContactName` TEXT(45) NULL,

`Adress` TINYTEXT NULL,

`City` TEXT(45) NULL,

`PostalCode` VARCHAR(6) NULL,

`Country` TEXT(45) NULL,

`Phone` VARCHAR(12) NULL,

PRIMARY KEY (`SuppliersID`))

ENGINE = InnoDB;

-----------------------------------------------------

-- Table `mydb`.`Categories`

-- -----------------------------------------------------

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

`CategoriesID` INT NOT NULL,

`CategoriesName` TINYTEXT NULL,

`Description` TINYTEXT NULL,

PRIMARY KEY (`CategoriesID`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Products`

-- -----------------------------------------------------

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

`ProductID` INT NOT NULL,

`ProductName` TINYTEXT NULL,

`Unit` TEXT(20) NULL,

`Price` DECIMAL(5,2) NULL,

`SuppliersID` INT NULL,

`CategoriesID` INT NULL,

PRIMARY KEY (`ProductID`),

CONSTRAINT `SuppliersID`

FOREIGN KEY (`SuppliersID`)

REFERENCES `mydb`.`Suppliers` (`SuppliersID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `CategoriesID`

FOREIGN KEY (`CategoriesID`)

REFERENCES `mydb`.`Categories` (`CategoriesID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `SuppliersID\_idx` ON `mydb`.`Products` (`SuppliersID` ASC);

CREATE INDEX `CategoriesID\_idx` ON `mydb`.`Products` (`CategoriesID` ASC);

-- -----------------------------------------------------

-- Table `mydb`.`OrderDetails`

-- -----------------------------------------------------

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

`OrderDetailsID` INT NOT NULL,

`Quantity` INT(50) NULL,

`OrderID` INT NULL,

`ProductID` INT NULL,

PRIMARY KEY (`OrderDetailsID`),

CONSTRAINT `OrderID`

FOREIGN KEY (`OrderID`)

REFERENCES `mydb`.`Orders` (`OrderID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `PoductID`

FOREIGN KEY (`ProductID`)

REFERENCES `mydb`.`Products` (`ProductID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

CREATE INDEX `OrderID\_idx` ON `mydb`.`OrderDetails` (`OrderID` ASC);

CREATE INDEX `PoductID\_idx` ON `mydb`.`OrderDetails` (`ProductID` ASC);

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

### Приложение 4. SQL-код создания и наполнения БД Northwind.

drop database Northwind;

CREATE DATABASE Northwind;

use Northwind;

CREATE TABLE Categories (

CategoryID int NOT NULL AUTO\_INCREMENT,

CategoryName VARCHAR(255),

Description VARCHAR(255),

PRIMARY KEY (CategoryID)

);

INSERT INTO Categories (CategoryName, Description) VALUES ('Beverages','Soft drinks, coffees, teas, beers, and ales');

INSERT INTO Categories (CategoryName, Description) VALUES ('Condiments','Sweet and savory sauces, relishes, spreads, and seasonings');

INSERT INTO Categories (CategoryName, Description) VALUES ('Confections','Desserts, candies, and sweet breads');

INSERT INTO Categories (CategoryName, Description) VALUES ('Dairy Products','Cheeses');

INSERT INTO Categories (CategoryName, Description) VALUES ('Grains/Cereals','Breads, crackers, pasta, and cereal');

INSERT INTO Categories (CategoryName, Description) VALUES ('Meat/Poultry','Prepared meats');

INSERT INTO Categories (CategoryName, Description) VALUES ('Produce','Dried fruit and bean curd');

INSERT INTO Categories (CategoryName, Description) VALUES ('Seafood','Seaweed and fish');

CREATE TABLE Customers (

CustomerID int NOT NULL AUTO\_INCREMENT,

CustomerName VARCHAR(255),

ContactName VARCHAR(255),

Address VARCHAR(255),

City VARCHAR(255),

PostalCode VARCHAR(255),

Country VARCHAR(255),

PRIMARY KEY (CustomerID)

);

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Alfreds Futterkiste','Maria Anders','Obere Str. 57','Berlin','12209','Germany');

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Ana Trujillo Emparedados y helados','Ana Trujillo','Avda. de la Constitución 2222','México D.F.','05021','Mexico');

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Antonio Moreno Taquería','Antonio Moreno','Mataderos 2312','México D.F.','05023','Mexico');

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Berglunds snabbköp','Christina Berglund','Berguvsvägen 8','Luleå','S-958 22','Sweden');

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Blauer See Delikatessen','Hanna Moos','Forsterstr. 57','Mannheim','68306','Germany');

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country) VALUES ('Blondel père et fils','Frédérique Citeaux','24, place Kléber','Strasbourg','67000','France');

CREATE TABLE Employees (

EmployeeID int NOT NULL AUTO\_INCREMENT,

LastName VARCHAR(255),

FirstName VARCHAR(255),

BirthDate DATE,

Photo VARCHAR(255),

Notes TEXT,

PRIMARY KEY (EmployeeID)

);

INSERT INTO Employees (LastName,FirstName,BirthDate,Photo,Notes) VALUES ('Davolio','Nancy','1968-12-08','EmpID1.pic','Education includes a BA in psychology from Colorado State University. She also completed (The Art of the Cold Call). Nancy is a member of ''Toastmasters International''.');

INSERT INTO Employees (LastName,FirstName,BirthDate,Photo,Notes) VALUES ('Fuller','Andrew','1952-02-19','EmpID2.pic','Andrew received his BTS commercial and a Ph.D. in international marketing from the University of Dallas. He is fluent in French and Italian and reads German. He joined the company as a sales representative, was promoted to sales manager and was then named vice president of sales. Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association.');

INSERT INTO Employees (LastName,FirstName,BirthDate,Photo,Notes) VALUES ('Leverling','Janet','1963-08-30','EmpID3.pic','Janet has a BS degree in chemistry from Boston College). She has also completed a certificate program in food retailing management. Janet was hired as a sales associate and was promoted to sales representative.');

INSERT INTO Employees (LastName,FirstName,BirthDate,Photo,Notes) VALUES ('Peacock','Margaret','1958-09-19','EmpID4.pic','Margaret holds a BA in English literature from Concordia College and an MA from the American Institute of Culinary Arts. She was temporarily assigned to the London office before returning to her permanent post in Seattle.');

CREATE TABLE OrderDetails (

OrderDetailID int NOT NULL AUTO\_INCREMENT,

OrderID int,

ProductID int,

Quantity int,

PRIMARY KEY (OrderDetailID)

);

INSERT INTO OrderDetails (OrderID,ProductID,Quantity) VALUES (10248,11,12);

INSERT INTO OrderDetails (OrderID,ProductID,Quantity) VALUES (10248,42,10);

INSERT INTO OrderDetails (OrderID,ProductID,Quantity) VALUES (10248,72,5);

INSERT INTO OrderDetails (OrderID,ProductID,Quantity) VALUES (10249,14,9);

INSERT INTO OrderDetails (OrderID,ProductID,Quantity) VALUES (10249,51,40);

INSERT INTO OrderDetails (OrderID,ProductID,Quantity) VALUES (10250,41,10);

INSERT INTO OrderDetails (OrderID,ProductID,Quantity) VALUES (10250,51,35);

INSERT INTO OrderDetails (OrderID,ProductID,Quantity) VALUES (10250,65,15);

CREATE TABLE Orders (

OrderID int NOT NULL AUTO\_INCREMENT,

CustomerID int,

EmployeeID int,

OrderDate DATE,

ShipperID int,

PRIMARY KEY (OrderID)

); INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10248,90,5,'1996-07-04',3);

INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10249,81,6,'1996-07-05',1);

INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10250,34,4,'1996-07-08',2);

INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10251,84,3,'1996-07-08',1);

INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10252,76,4,'1996-07-09',2);

INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10253,34,3,'1996-07-10',2);

INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10254,14,5,'1996-07-11',2);

INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10255,68,9,'1996-07-12',3);

INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10256,88,3,'1996-07-15',2);

INSERT INTO Orders (OrderID,CustomerID,EmployeeID,OrderDate,ShipperID) VALUES (10305,55,8,'1996-09-13',3);

CREATE TABLE Products (

ProductID int NOT NULL AUTO\_INCREMENT,

ProductName VARCHAR(255),

SupplierID int,

CategoryID int,

Unit VARCHAR(255),

Price numeric(15,2),

PRIMARY KEY (ProductID)

);

INSERT INTO Products (ProductID,ProductName,SupplierID,CategoryID,Unit,Price) VALUES (1,'Chais',1,1,'10 boxes x 20 bags',18);

INSERT INTO Products (ProductID,ProductName,SupplierID,CategoryID,Unit,Price) VALUES (2,'Chang',1,1,'24 - 12 oz bottles',19);

INSERT INTO Products (ProductID,ProductName,SupplierID,CategoryID,Unit,Price) VALUES (5,'Chef Anton''s Gumbo Mix',2,2,'36 boxes',21.35);

INSERT INTO Products (ProductID,ProductName,SupplierID,CategoryID,Unit,Price) VALUES (6,'Grandma''s Boysenberry Spread',3,2,'12 - 8 oz jars',25);

INSERT INTO Products (ProductID,ProductName,SupplierID,CategoryID,Unit,Price) VALUES (31,'Gorgonzola Telino',14,4,'12 - 100 g pkgs',12.5);

INSERT INTO Products (ProductID,ProductName,SupplierID,CategoryID,Unit,Price) VALUES (32,'Mascarpone Fabioli',14,4,'24 - 200 g pkgs.',32);

INSERT INTO Products (ProductID,ProductName,SupplierID,CategoryID,Unit,Price) VALUES (33,'Geitost',15,4,'500 g',2.5);

CREATE TABLE Shippers (

ShipperID int NOT NULL AUTO\_INCREMENT,

ShipperName VARCHAR(255),

Phone VARCHAR(255),

PRIMARY KEY (ShipperID)

);

INSERT INTO Shippers (ShipperName, Phone) VALUES ('Speedy Express', '(503) 555-9831');

INSERT INTO Shippers (ShipperName, Phone) VALUES ('United Package', '(503) 555-3199');

INSERT INTO Shippers (ShipperName, Phone) VALUES ('Federal Shipping', '(503) 555-9931');

CREATE TABLE Suppliers (

SupplierID int NOT NULL AUTO\_INCREMENT,

SupplierName VARCHAR(255),

ContactName VARCHAR(255),

Address VARCHAR(255),

City VARCHAR(255),

PostalCode VARCHAR(255),

Country VARCHAR(255),

Phone VARCHAR(255),

PRIMARY KEY (SupplierID)

);

INSERT INTO Suppliers (SupplierID,SupplierName,ContactName,Address,City,PostalCode,Country,Phone) VALUES (1,'Exotic Liquid','Charlotte Cooper','49 Gilbert St.','Londona','EC1 4SD','UK','(171) 555-2222');

INSERT INTO Suppliers (SupplierID,SupplierName,ContactName,Address,City,PostalCode,Country,Phone) VALUES (2,'New Orleans Cajun Delights','Shelley Burke','P.O. Box 78934','New Orleans','70117','USA','(100) 555-4822');

INSERT INTO Suppliers (SupplierID,SupplierName,ContactName,Address,City,PostalCode,Country,Phone) VALUES (3,'Grandma Kelly''s Homestead','Regina Murphy','707 Oxford Rd.','Ann