|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра промышленной информатики (ПИ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ**

по дисциплине «Разработка баз данных»

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | ИКБО-04-21 Даурбеков М.И. | (подпись) | |
| Преподаватель | Чучаева С.М. | (подпись) | |
| Отчет представлен | «22» ноября 2023 г. | |  | |

Москва 2023 г.

**Оглавление**

[**Задание** 3](#_Toc153639655)

[**Глава 1** 4](#_Toc153639656)

[**Глава 2** 7](#_Toc153639657)

[**Глава 3** 13](#_Toc153639658)

[**Глава 4** 18](#_Toc153639659)

[Часть 1: Основные команды для работы с данными таблицы 18](#_Toc153639660)

[Часть 2: Ссылочная целостность 26](#_Toc153639661)

[Часть 3: Импорт и экспорт ER-диаграммы 36](#_Toc153639662)

[Часть 4: Сложные запросы 37](#_Toc153639663)

[**Глава 5** 47](#_Toc153639664)

[Процедуры 47](#_Toc153639665)

[Функции 52](#_Toc153639666)

[Триггеры 55](#_Toc153639667)

[**Глава 6** 58](#_Toc153639668)

[Синтаксис оконных функций 58](#_Toc153639669)

[Агрегатные функции 61](#_Toc153639670)

[Ранжирующие функции 64](#_Toc153639671)

[Функции смещения 68](#_Toc153639672)

# **Задание**

1. Создание логической модели базы данных. Описание ER-модели, генерация на ее основе реляционной модели данных.
2. Реализация модели в СУБД. В качестве СУБД могут быть выбраны: MS SQL Server, MySQL или PostgreSQL или иное серверное СУБД.
3. Заполнение базы данных.
4. Создание различных запросов на получение данных (для формирования навыков работы с реализацией различных операций реляционной алгебры). Для каждой из операций (исключая деление) нужно показать минимум три запроса (хотя один и тот же запрос может демонстрировать выполнение нескольких операций).
5. Создание хранимых процедур и триггеров для обеспечения серверной части работы с данными.

# **Глава 1**

Создадим логическую модель БД «Кондитерская фабрика»:

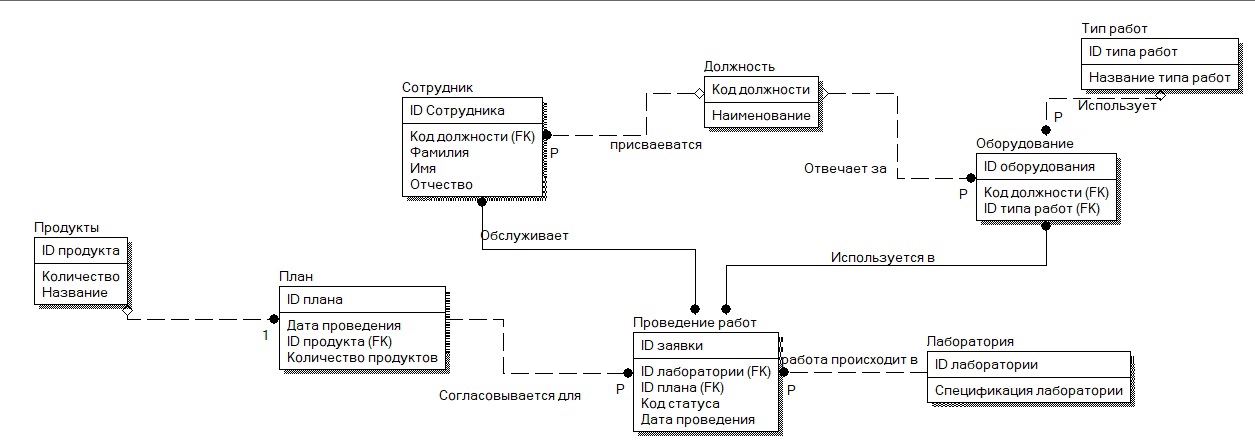


Рисунок 1 – Логическая модель

Опишем каждую сущность БД:

Сущность «Продукты»:

Таблица 1

|  |  |
| --- | --- |
| Поле | Описание |
| ID продукта | Первичный ключ таблицы «Продукты» |
| Количество | Количество продуктов |
| Название | Название продукта |

Сущность «План»:

Таблица 2

|  |  |
| --- | --- |
| Поле | Описание |
| ID плана | Первичный ключ таблицы «Тип работ». |
| Дата проведения | Дата запланированного проведения работ |
| Количество продуктов | Код, по которому можно определить взятые продукты для работы |
| ID продуктов(FK) | Id, по которому можно определить продукты, которые используются |

Сущность «Должность»:

Таблица 3

|  |  |
| --- | --- |
| Поле | Описание |
| Код должности | Код должности состоящий из набора чисел |
| Наименование | Название должности |

Сущность «Сотрудник»:

Таблица 4

|  |  |
| --- | --- |
| Поле | Описание |
| ID Сотрудника | Первичный ключ таблицы «Сотрудник». |
| Код должности(FK) | Код должности, по которому можно определить должность из определенной таблицы |
| Фамилия | Фамилия сотрудника |
| Имя | Имя сотрудника |
| Отчество | Отчество сотрудника |

Сущность «Оборудование»:

Таблица 5

|  |  |
| --- | --- |
| Поле | Описание |
| ID оборудования | Первичный ключ таблицы «Оборудование». |
| Тип работ(FK) | Тип работ на этом оборудовании, обозначенный кодовым словом. |
| Ранг(FK) | Ответственный по работе с этим оборудованием |

Сущность «Лаборатория»:

Таблица 6

|  |  |
| --- | --- |
| Поле | Описание |
| ID лаборатории | Первичный ключ таблицы «Лаборатория». |
| Спецификация лаборатории | Предназначение лаборатории, обозначенный кодовым словом. |

Сущность «Проведение работ»:

Таблица 7

|  |  |
| --- | --- |
| Поле | Описание |
| ID заявки | Первичный ключ таблицы «Проведение работ». |
| ID лаборатории (FK) | Id, по которому можно определить лабораторию, в которой проходят работы |
| ID использованных продуктов(FK) | Id, по которому можно определить взятые продукты и запланированное время работы |
| Код статуса | Код статуса работы |
| Дата проведения | Дата фактического проведения работ |

# **Глава 2**

Создадим физическую модель БД «Кондитерская фабрика»:

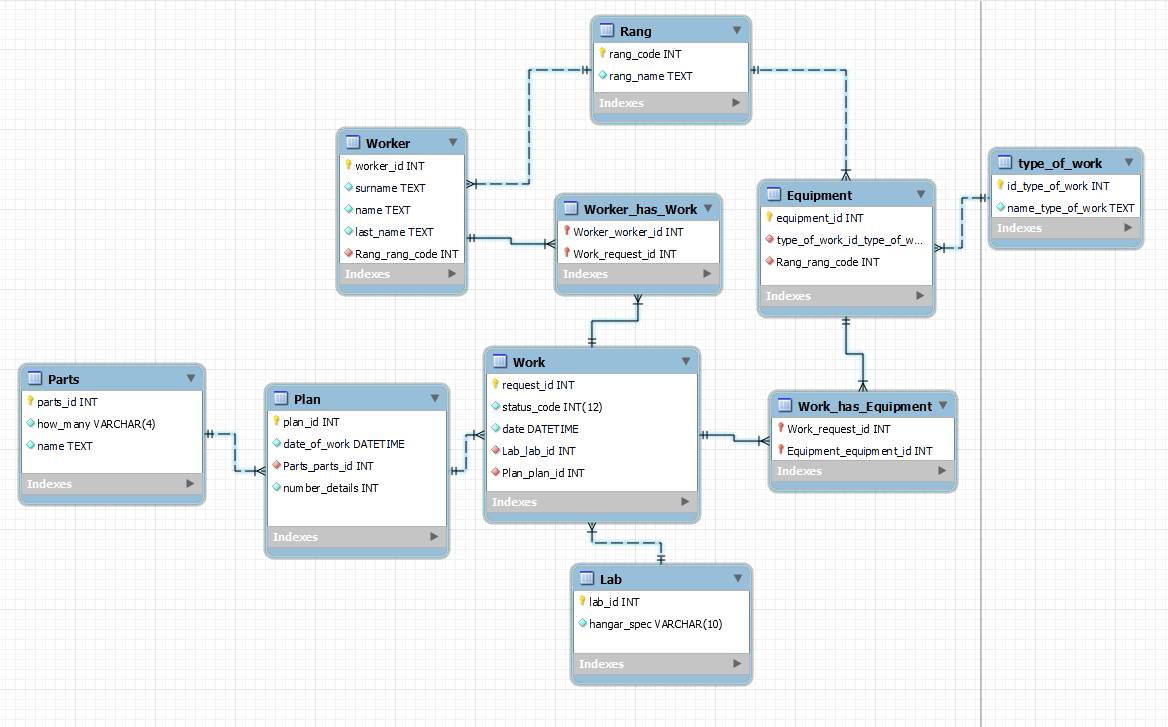


Рисунок 2 – Физическая модель БД

Опишем все сущности БД с указанием полей и их типов данных:

Сущность «Rang»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| rang\_code | Int | Auto Increment |
| rang\_name | Text | Not null |

Сущность «Parts»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| parts\_id | Int | Auto Increment |
| how\_many | Varchar(4) | Not null |
| name | Text | Not null |

Сущность «Plan»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| plan\_id | Int | Auto Increment |
| date\_of\_work | Datetame | Not null |
| parts\_parts\_id | Int | Not null |
| number\_details | Int | Not null |

Сущность «Worker»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| worker\_id | Int | Auto Increment |
| surname | Text | Not null |
| name | Text | Not null |
| last\_name | Text | Not null |
| rang\_rang\_code | Int | Not null |

Сущность «Equipment»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| equipment\_id | Int | Auto Increment |
| type\_of\_work\_type\_of\_work\_id | Int | Not null |
| rang\_rang\_code | Int | Not null |

Сущность «Lab»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| lab\_id | Int | Auto Increment |
| hangar\_spec | Varchar(10) | Not null |

Сущность «Work»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| request\_id | Int | Auto Increment |
| status\_code | Int(12) | Not null |
| date | Datetime | Not null |
| plan\_plan\_id | Int | Not null |
| lab\_lab\_id | Int | Not null |

Сущность «Type\_of\_work»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| type\_of\_work\_id | Int | Auto Increment |
| name\_type\_of\_work | text | Not null |

Сущность «Worker\_has\_work»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| worker\_worker\_id | Int | Not null |
| work\_request\_id | Int | Not null |

Сущность «Work\_has\_equipment»:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Null/Not null |
| equipment\_equipment\_id | Int | Not null |
| work\_request\_id | Int | Not null |

Создадим базу данных электронной библиотеки cFactory, и начнем работать с ней:

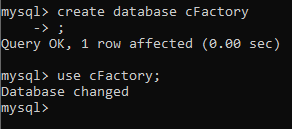


Рисунок 3 – Создание и начало работы с БД

Создадим таблицу Parts:

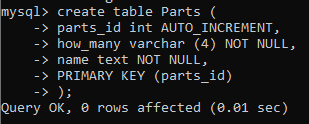


Рисунок 4 – Создание таблицы

Создадим таблицу Plan:

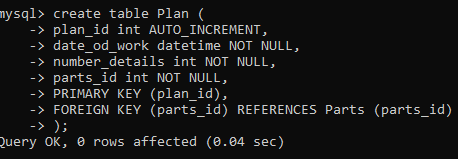


Рисунок 5 – Создание таблицы

Создадим таблицу Rang:

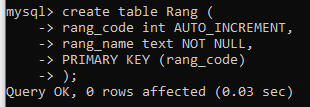


Рисунок 6 – Создание таблицы

Создадим таблицу Type\_of\_work:

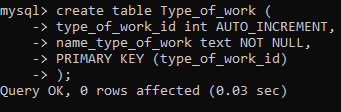


Рисунок 7 – Создание таблицы

Создадим таблицу Worker:

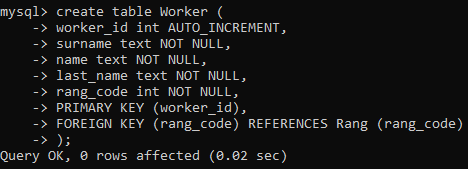


Рисунок 8 – Создание таблицы

Создадим таблицу Equipment:

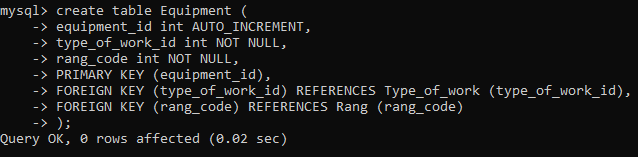


Рисунок 9 – Создание таблицы

Создадим таблицу Lab:

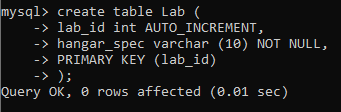


Рисунок 10 – Создание таблицы

Создадим таблицу Work:

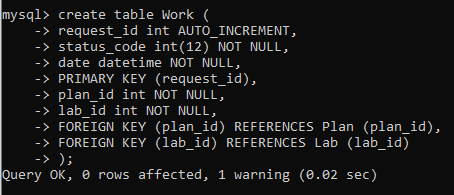


Рисунок 11 – Создание таблицы

Создадим таблицу Worker\_has\_work:

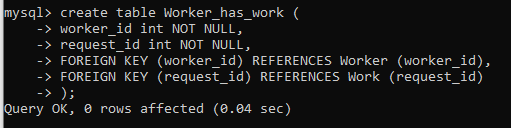


Рисунок 12 – Создание таблицы

Создадим таблицу Work\_has\_equipment:

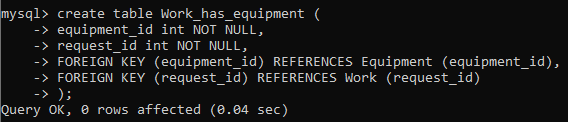


Рисунок 13 – Создание таблицы

# **Глава 3**

Заполним созданные в прошлой работе таблицы:

Таблица Parts

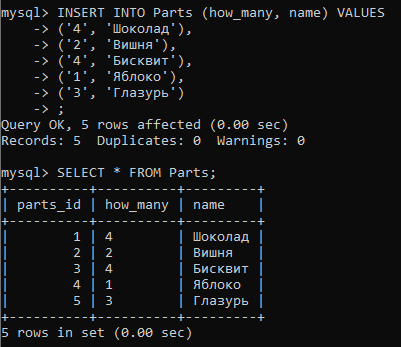


Рисунок 14 – Создание и заполнение таблицы

Таблица Plan:

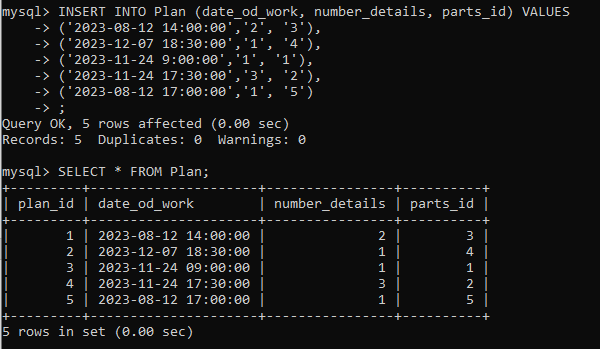


Рисунок 15 – Создание и заполнение таблицы

Сущность Rang:

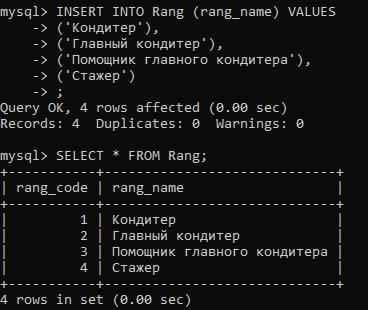


Рисунок 16 – Создание и заполнение таблицы

Таблица Worker:

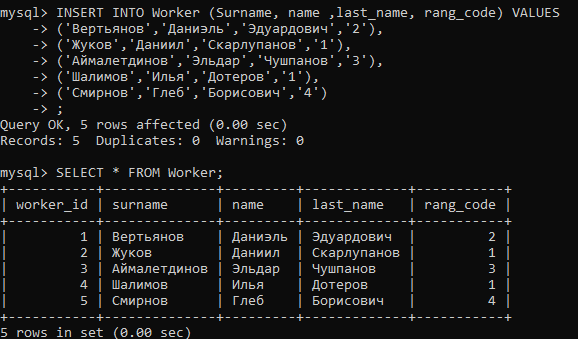


Рисунок 17 – Создание и заполнение таблицы

Таблица Equipment:

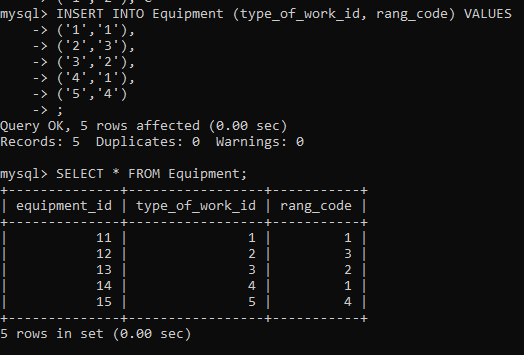


Рисунок 18 – Создание и заполнение таблицы

Таблица Lab:

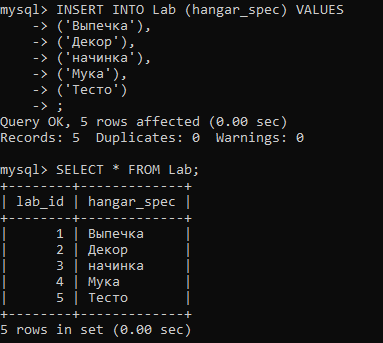


Рисунок 19– Создание и заполнение таблицы

Таблица Type\_of\_work:

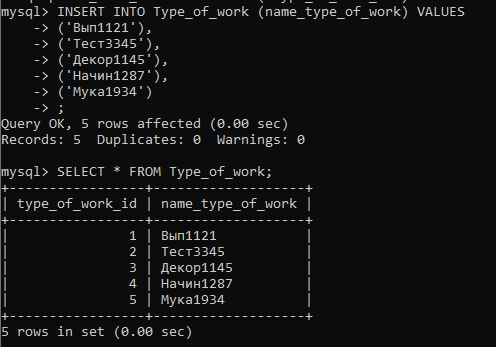


Рисунок 20– Создание и заполнение таблицы

Таблица Work:

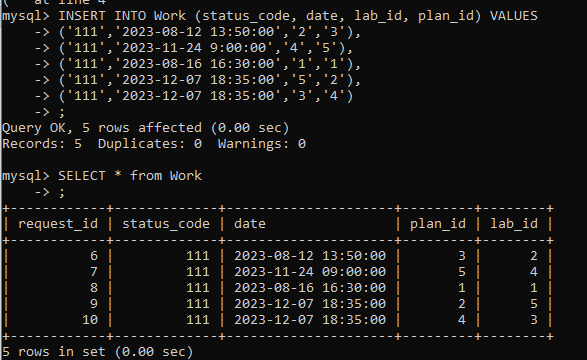


Рисунок 21– Создание и заполнение таблицы

Таблица Worker\_has\_work:

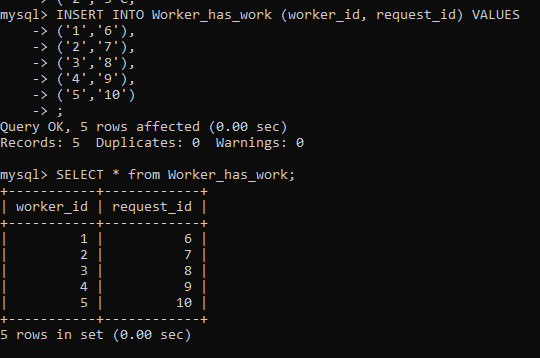


Рисунок 22– Создание и заполнение таблицы

Таблица Work\_has\_equipment:

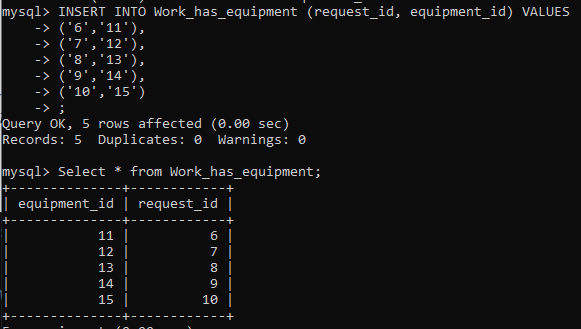


Рисунок 23– Создание и заполнение таблицы

# **Глава 4**

## Часть 1: Основные команды для работы с данными таблицы

*Выборка данных*

1. Просмотр всех столбцов таблицы:

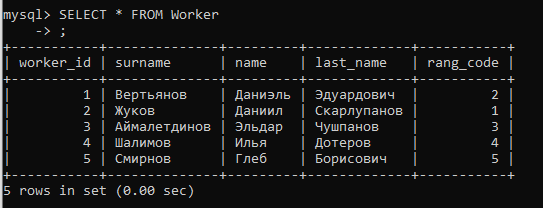


Рисунок 24 – Запрос SELECT

1. Просмотр конкретного столбца таблицы:

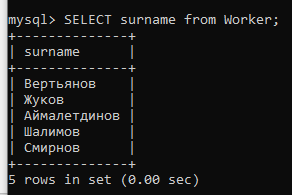


Рисунок 25 – Запрос SELECT

1. Просмотр конкретных столбцов таблицы:

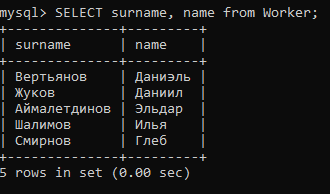


Рисунок 26 – Запрос SELECT

*Сортировка данных*

1. Сортировка данных таблицы по возрастанию:

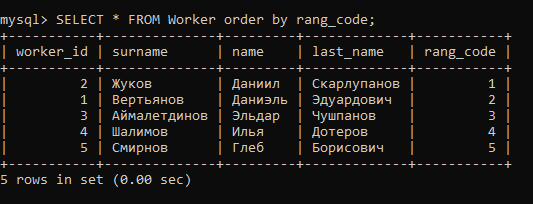


Рисунок 27 – Запрос SELECT с применением ORDER BY

1. Сортировка данных таблицы по убыванию:

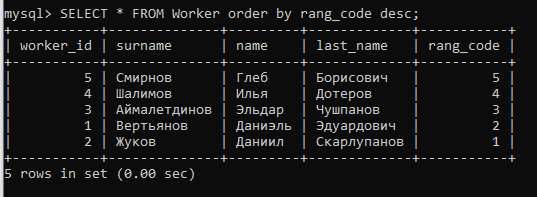


Рисунок 28 – Запрос SELECT с применением ORDER BY и DECS

*Выборка конкретных данных*

1. Просмотр конкретных данных таблицы:

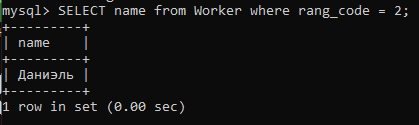


Рисунок 29 – Запрос SELECT с применением WHERE

*Применение операторов*

1. = (равно) – отбираются значения равные указанному:

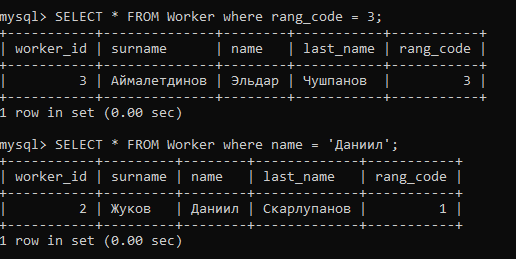


Рисунок 30 – Пример использования оператора =

1. > (больше) – отбираются значения больше указанного:

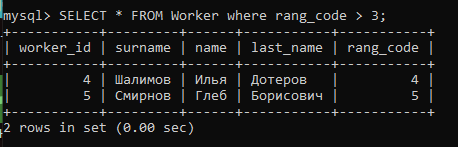


Рисунок 31 – Пример использования оператора >

1. < (меньше) – отбираются значения меньше указанного:

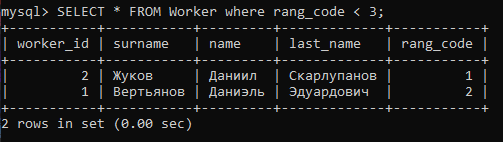


Рисунок 32 – Пример использования оператора <

1. >= (больше или равно) – отбираются значения большие и равные указанному:

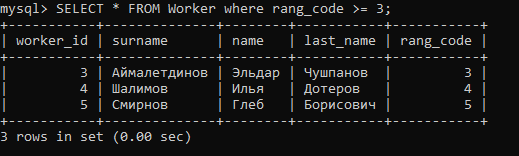


Рисунок 33 – Пример использования оператора >=

1. <= (меньше или равно) – отбираются значения меньшие и равные указанному:

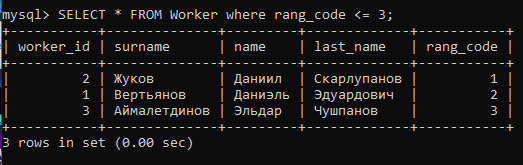


Рисунок 34 – Пример использования оператора <=

1. != (не равно) – отбираются значения не равные указанном:

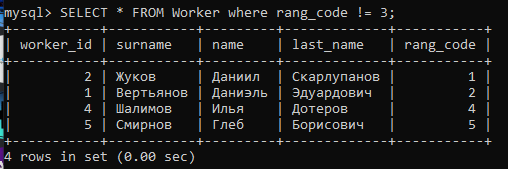


Рисунок 35 – Пример использования оператора !=

1. IS NOT NULL – отбираются строки, имеющие значения в указанном поле:

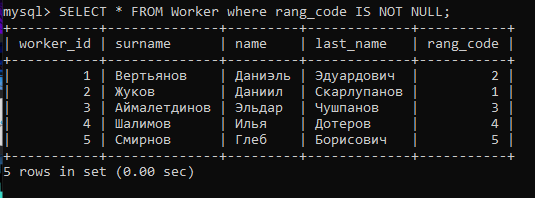


Рисунок 36 – Пример использования оператора IS NOT NULL

1. IS NULL – отбираются строки, не имеющие значения в указанном поле:



Рисунок 37 – Пример использования оператора IS NULL

1. BETWEEN (между) – отбираются значения, находящиеся между указанными:

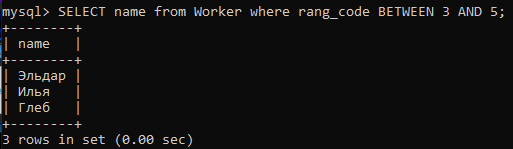


Рисунок 38 – Пример использования оператора BETWEEN

1. IN (значение содержится) – отбираются значения, соответствующие указанным:

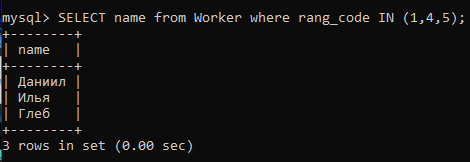


Рисунок 39 – Пример использования оператора IN

1. NOT IN (значение не содержится) – отбираются значения, кроме указанных:

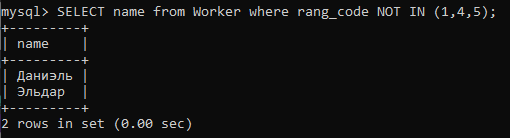


Рисунок 40 – Пример использования оператора NOT IN

1. LIKE (соответствие) – отбираются значения, соответствующие образцу:

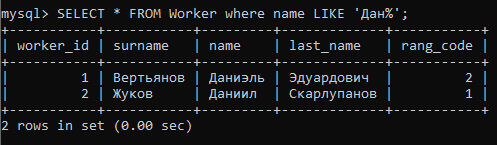


Рисунок 41 – Пример использования оператора LIKE

1. NOT LIKE (не соответствие) – отбираются значения, не соответствующие образцу:

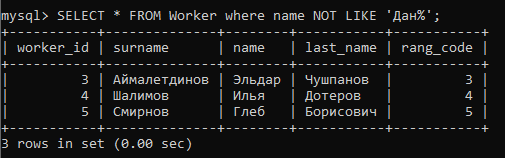


Рисунок 42 – Пример использования оператора NOT LIKE

*Изменение данных в таблице*

1. Добавление столбца в таблицу:

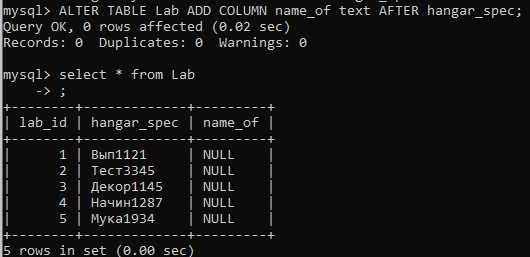


Рисунок 43 – Добавление столбца «названия» в таблицу «Лаборатория»

1. Изменение данных в таблице:

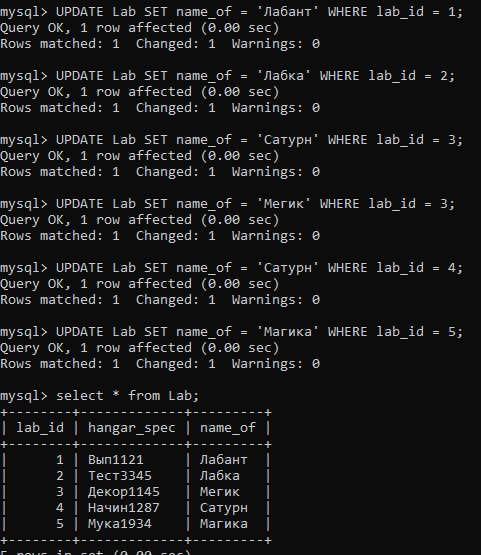


Рисунок 44 – Изменение строк в таблице

1. Переименование названия столбца:

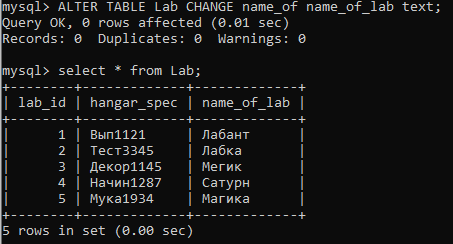


Рисунок 45 – Замена названия

1. Удаление строки из таблицы:

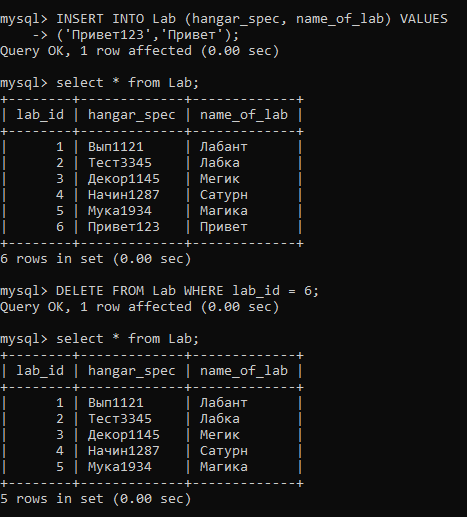


Рисунок 46 – Вставка новой строки в таблицу, удаление этой строки

## Часть 2: Ссылочная целостность

1. Plan:

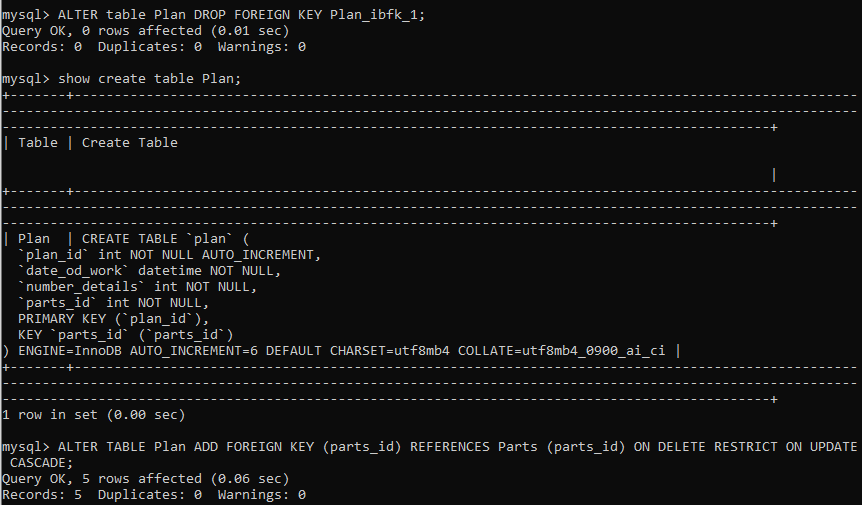


Рисунок 47 – Восстановление ссылочной целостности таблицы

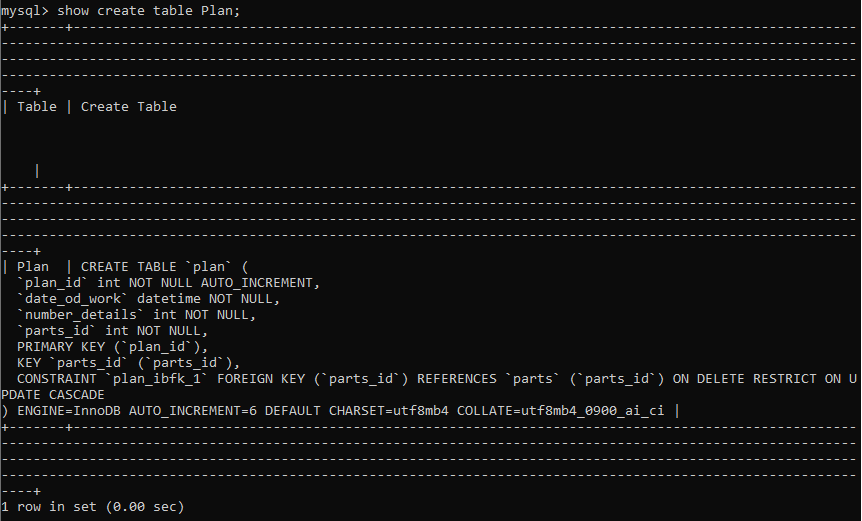


Рисунок 48 – FOREIGN KEY создан

1. Worker:

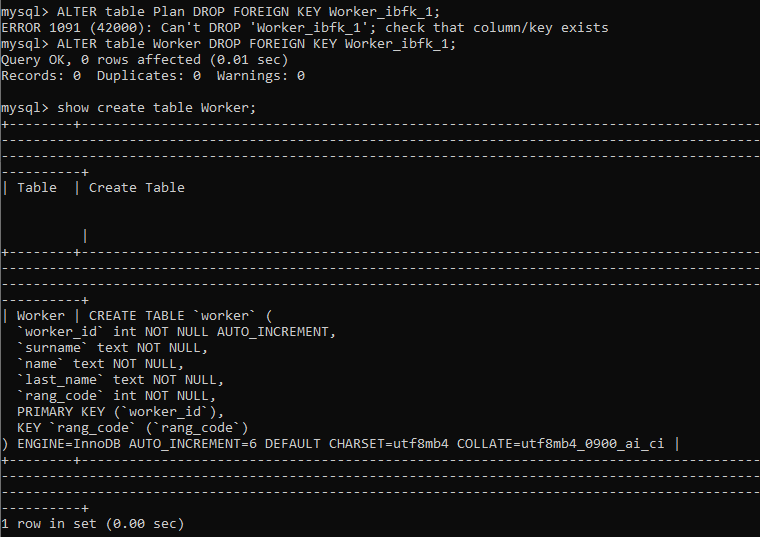


Рисунок 49 – Восстановление ссылочной целостности таблицы

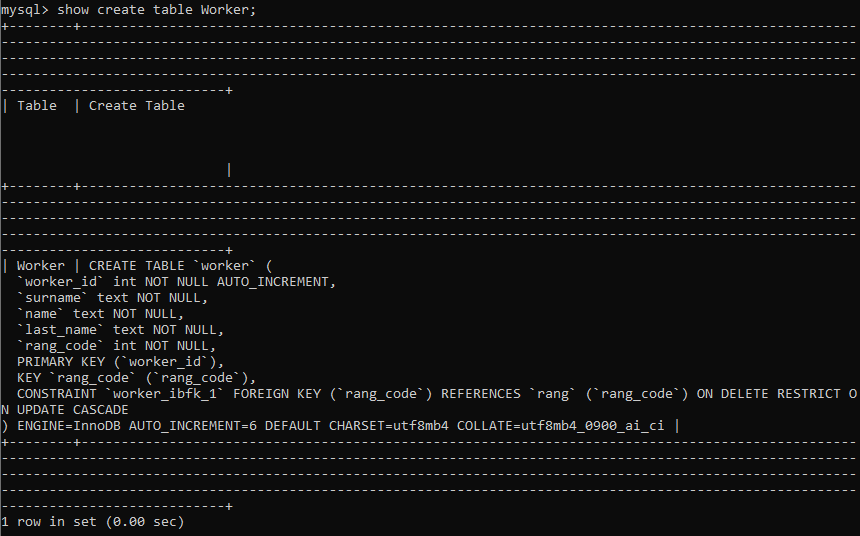


Рисунок 50 – FOREIGN KEY создан

1. Work:

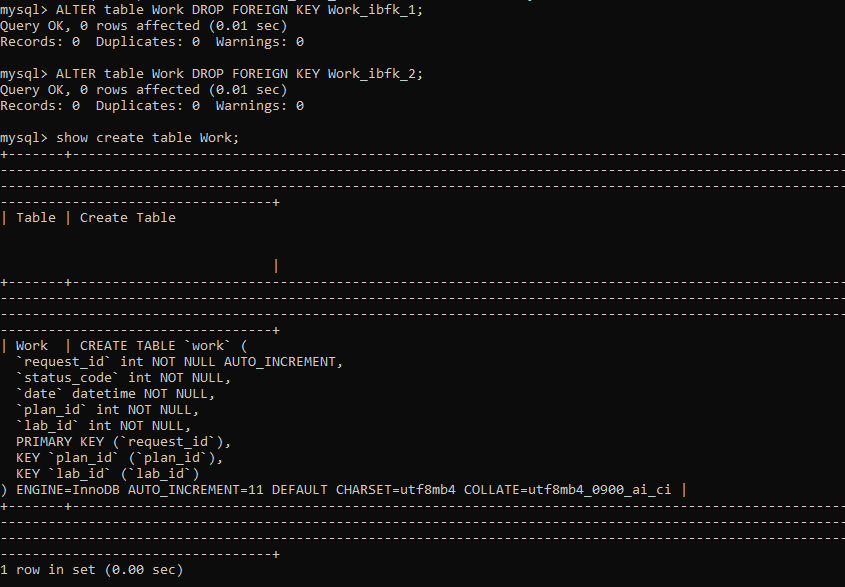


Рисунок 51 - Восстановление ссылочной целостности таблицы

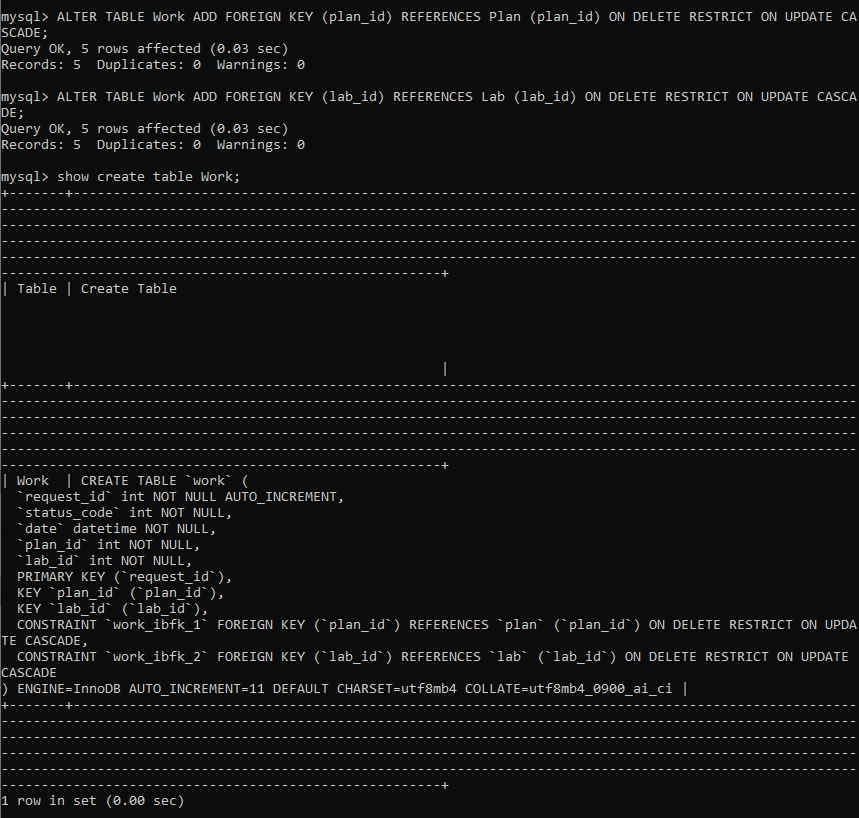


Рисунок 52 – FOREIGN KEY создан

1. Equipment:

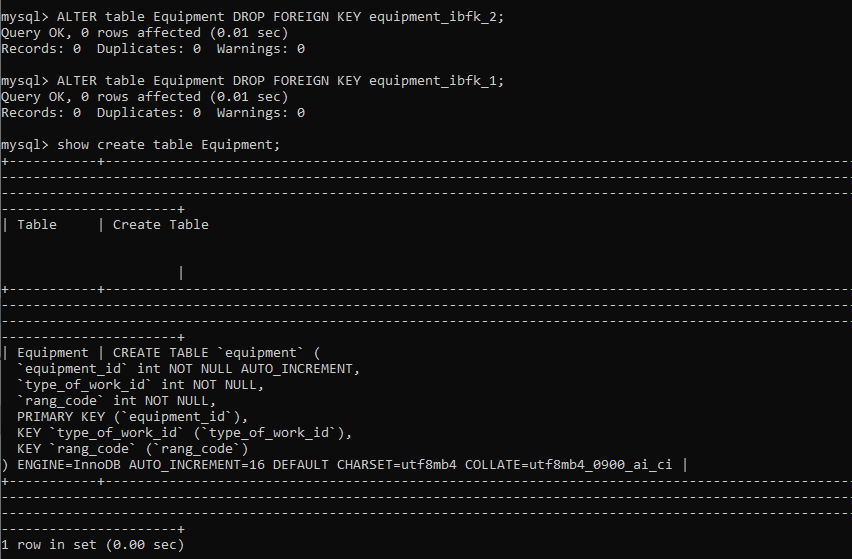


Рисунок 53 - Восстановление ссылочной целостности таблицы

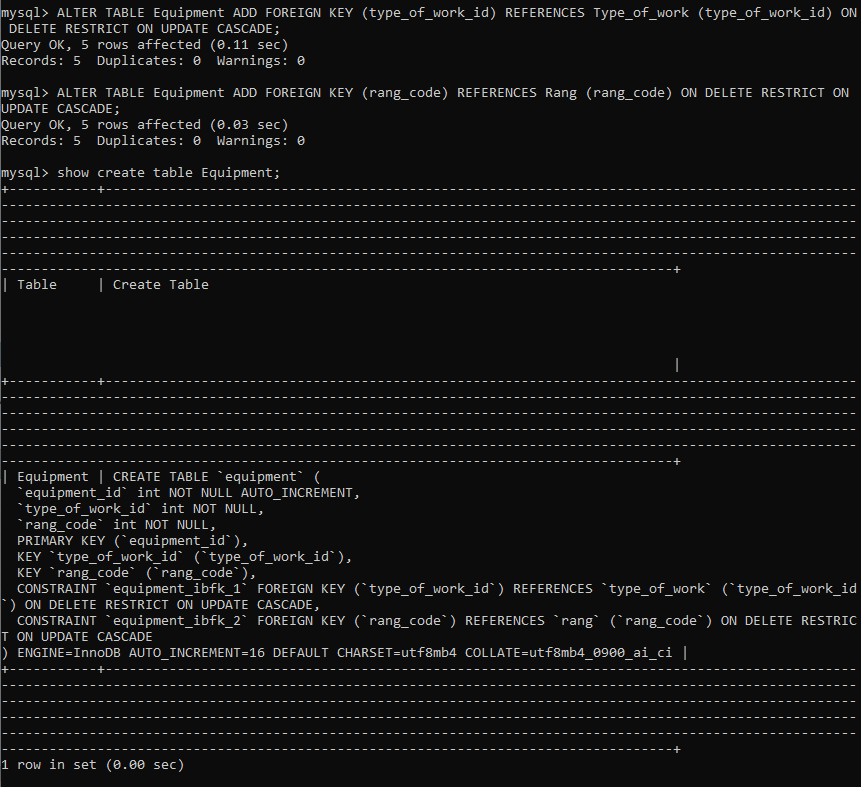


Рисунок 54 – FOREIGN KEY создан

1. Worker\_has\_work:

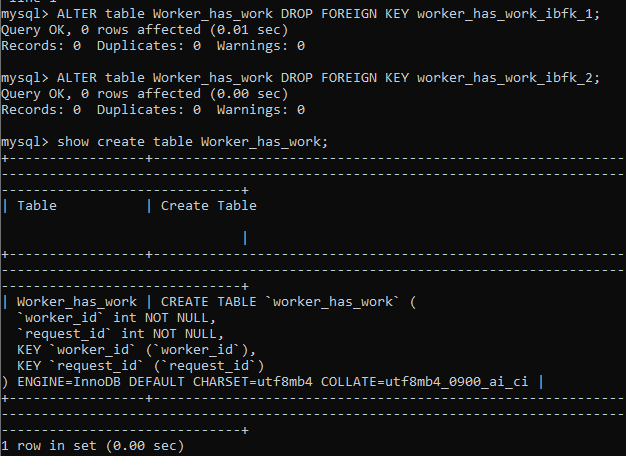


Рисунок 55 - Восстановление ссылочной целостности таблицы

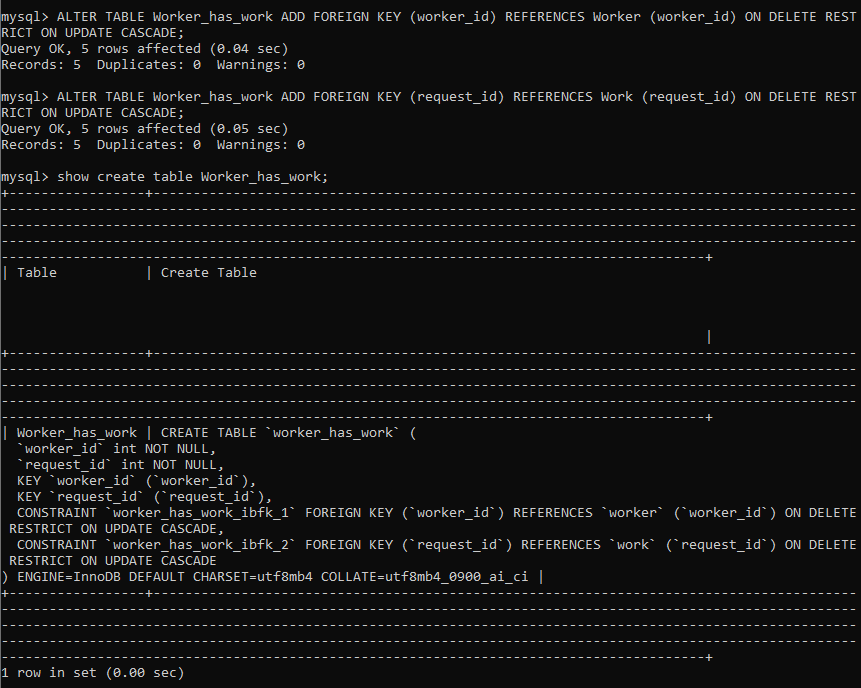


Рисунок 56 – FOREIGN KEY создан

1. Work\_has\_equipment:

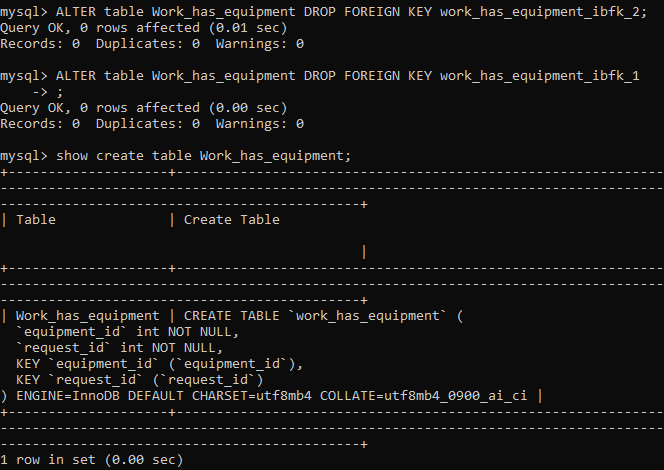


Рисунок 57 - Восстановление ссылочной целостности таблицы

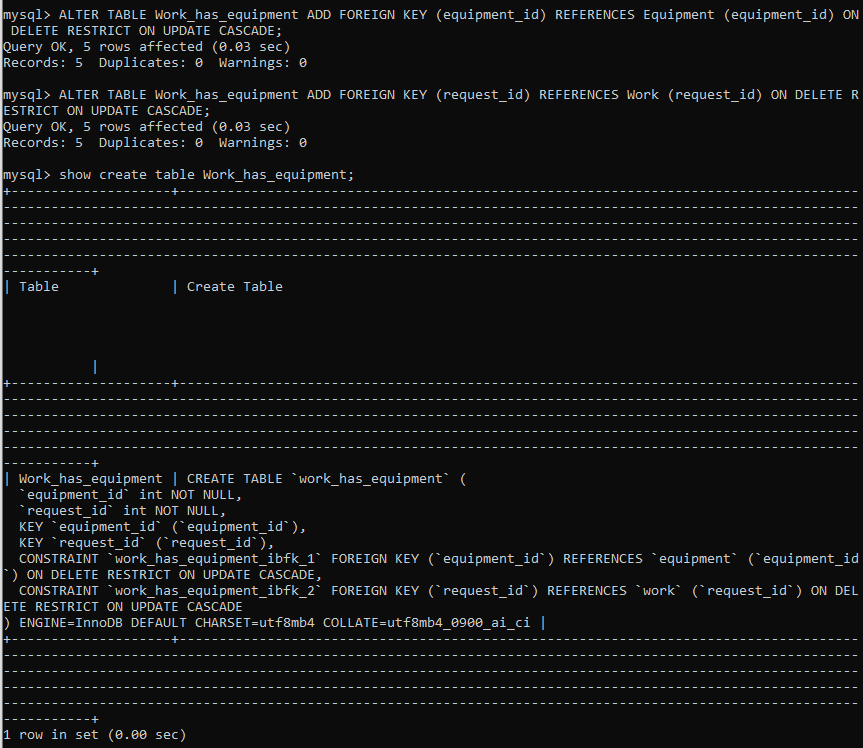


Рисунок 58 – FOREIGN KEY создан

## Часть 3: Импорт и экспорт ER-диаграммы

Вернемся к ER-диаграмме. Экспортируем ее и получим следующий скрипт:

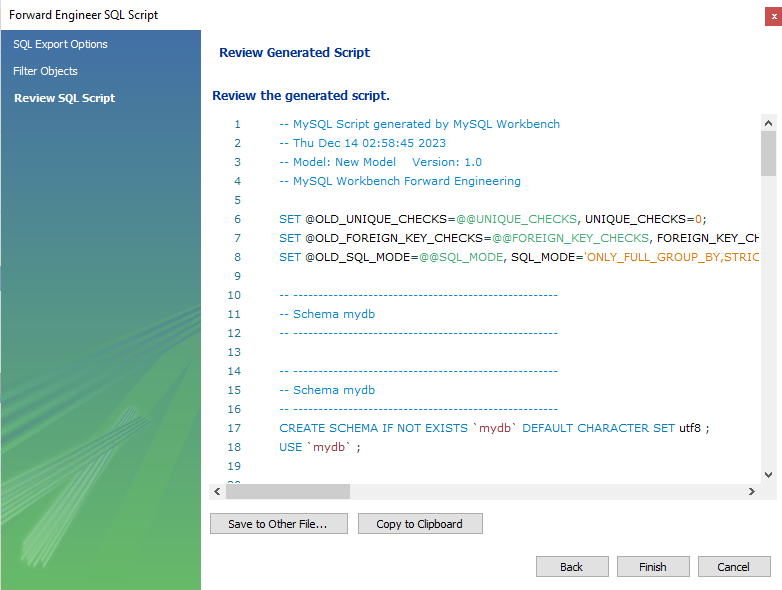


Рисунок 59 – Экспорт модели

## Часть 4: Сложные запросы

*Операция проекции*

Осуществляется выбор только части полей таблицы, т.е. производится вертикальная выборка данных:

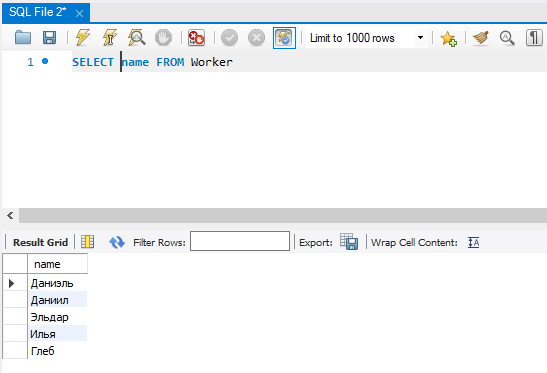


Рисунок 60 – Проекция

*Операция селекции*

Осуществляется горизонтальная выборка – в результат попадают только записи, удовлетворяющие условию.

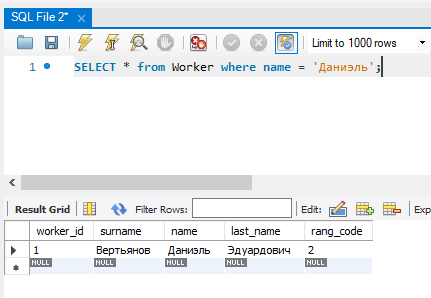


Рисунок 61 – Селекция

*Операция соединения*

Позволяет использовать несколько условий для запроса.

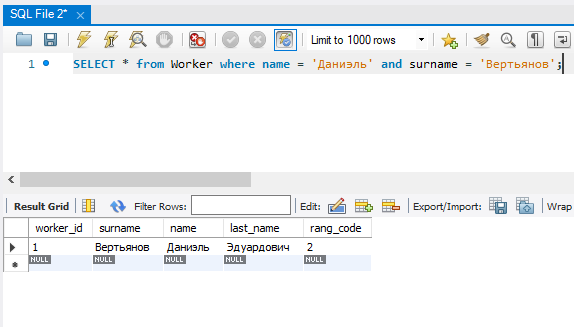


Рисунок 62 – Соединение

С помощью этой операции можно получать данные из разных таблиц:

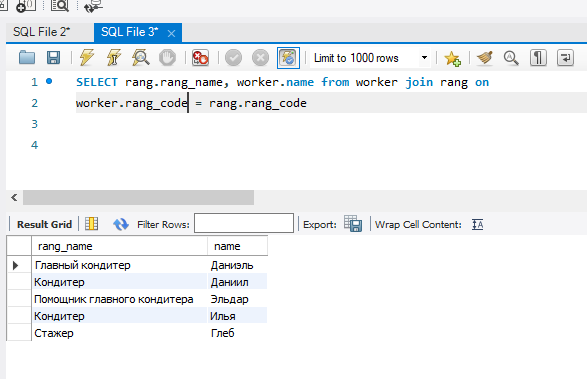


Рисунок 63 – Получение данных из разных таблиц

Это позволяет «раскрывать» id и получать более наглядную информацию:

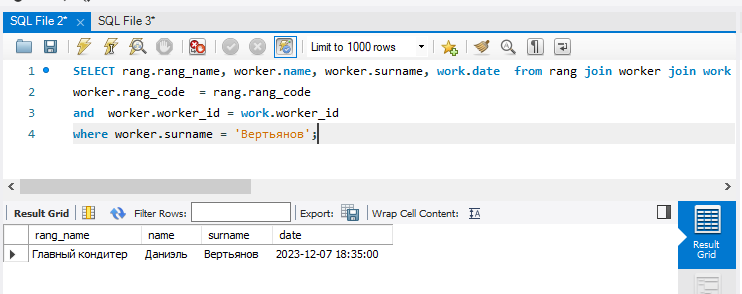


Рисунок 64 – Сложный запрос

*Операция объединения*

Теоретико-множественные операции часто можно записать с помощью логических операций, примененных в конструкции WHERE запроса.

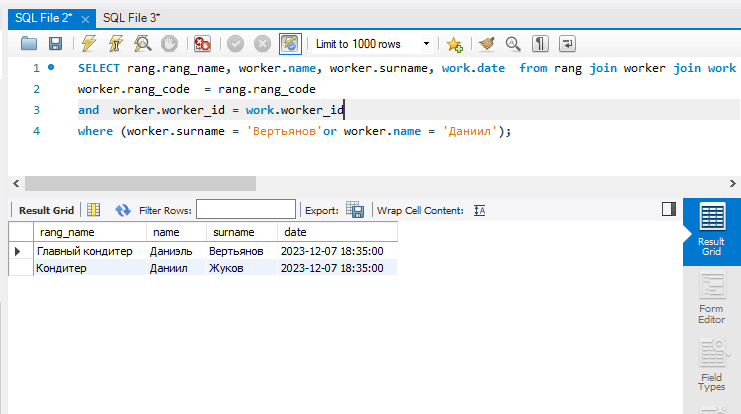


Рисунок 65 – Операция объединения

Аналогичный результат будет получен с помощью объединения результатов двух запросов (подзапросов) с одинаковой структурой результата:

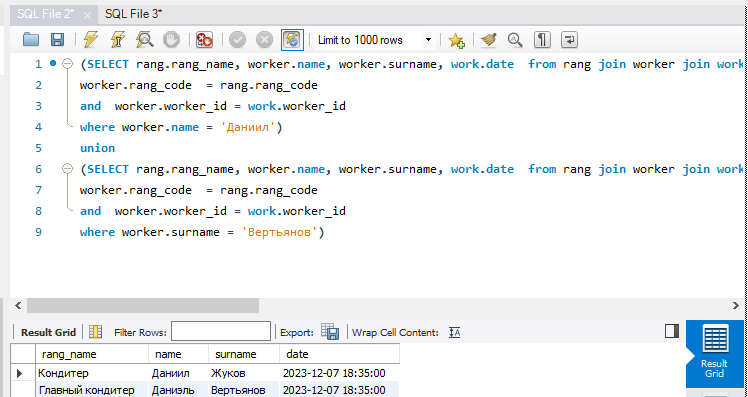


Рисунок 66 – Альтернативная запись

*Операция пересечения*

В простых случаях эту операцию можно описать с помощью логической операции AND. В более сложных случаях эта операция определяется чаще всего с помощью подзапроса и ключевого слова EXISTS, которое показывает наличие похожего элемента во множестве, которое задается подзапросом.

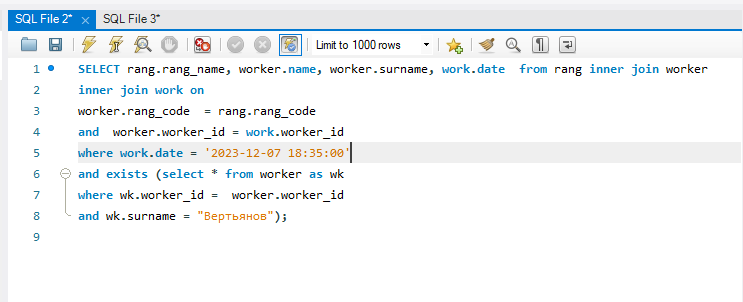


Рисунок 67 – Пересечение

*Операция разности*

Эта операция также определяется часто с помощью подзапроса с ключевым словом NOT EXISTS, которое показывает отсутствие элемента во множестве, задаваемом подзапросом. Приведем аналогичный предыдущему пример.

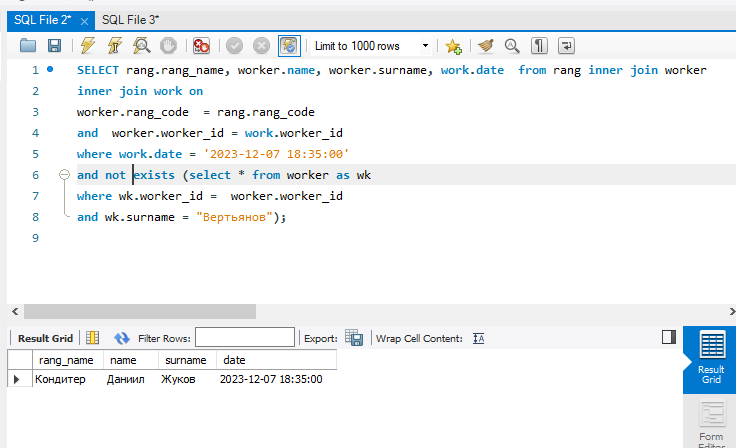


Рисунок 68 – Разность

*Операция группировки*

Эта операция связана со своеобразной сверткой таблицы по полям группировки. Помимо полей группировки результат запроса может содержать итоговые агрегирующие функции по группам (COUNT, SUM, AVG, MAX, MIN).

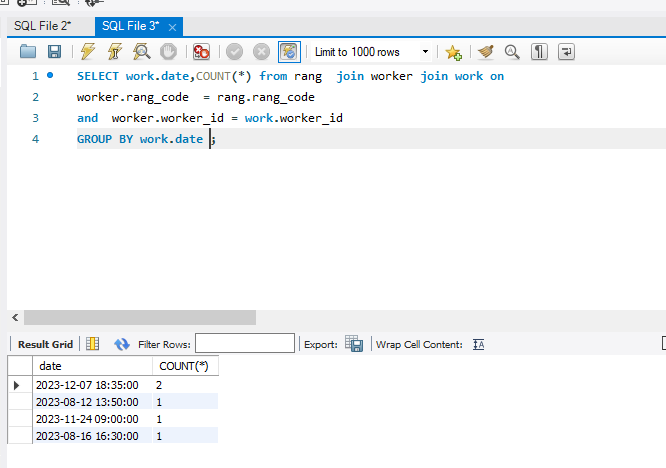


Рисунок 69 - Группировка

В случае, когда при группировке имеется условие отбора на группу, используется конструкция HAVING:

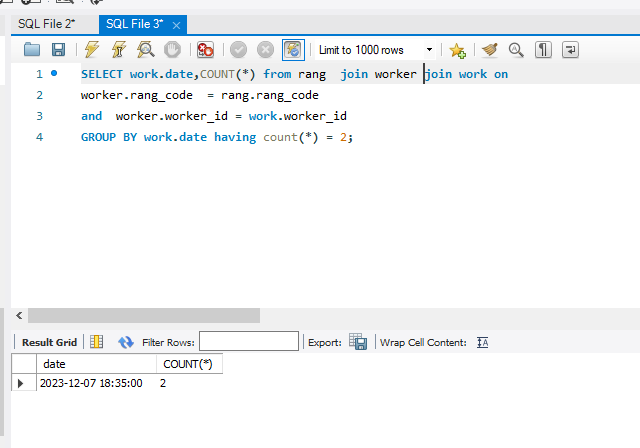


Рисунок 70 – конструкция Having

*Операция сортировки*

Операция сортировки в SQL позволяет упорядочить результаты запроса по заданному столбцу или нескольким столбцам. Для этого используется ключевое слово ORDER BY, после которого указываются столбцы, по которым нужно отсортировать результаты.

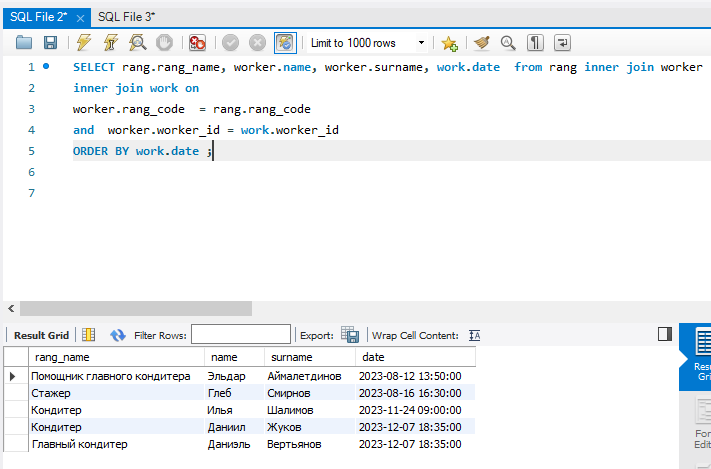


Рисунок 71 – Order by

Можно указать DESC для сортировки в обратном порядке:

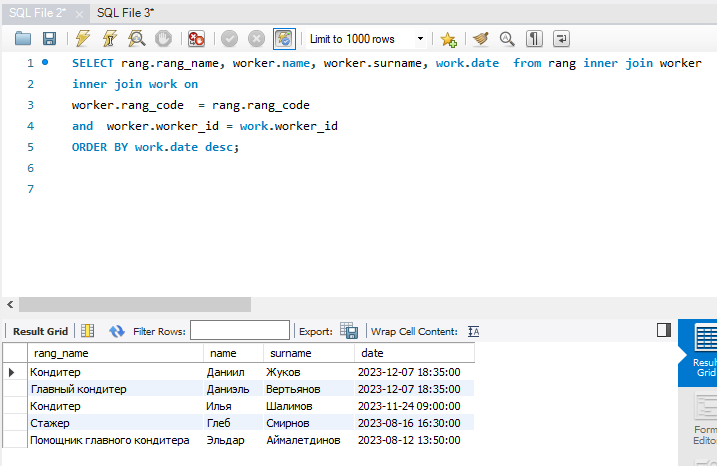


Рисунок 72 – Order by с использованием desc

*Операция деления*

Это самая нетривиальная операция реляционной алгебры, которая обычно применяется тогда, когда требуется найти все записи первой таблицы, которые соединяются естественным образом со всеми записями второй таблицы.

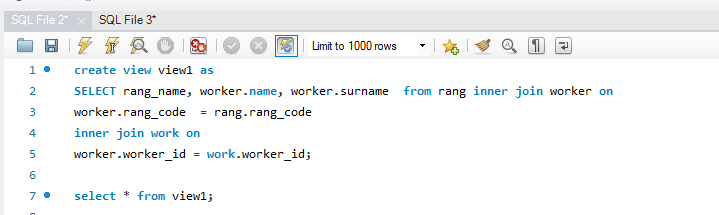


Рисунок 73 – Реализация операции деления с помощью представления

# **Глава 5**

## Процедуры

Создадим первую процедуру, для этого кликаем по папке «Stored Procedures» правой кнопкой мыши, выбираем опцию «Create Stored Procedure». Реализуем получение всех данных из таблицы Worker, запустим и проверим работоспособность процедуры:

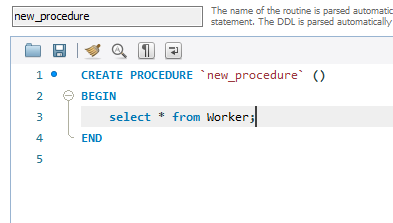


Рисунок 74 – Реализация процедуры

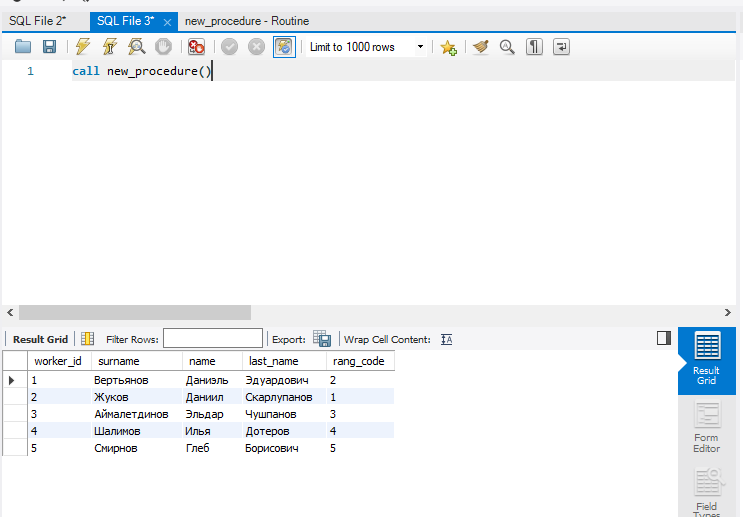


Рисунок 75 – Вызов процедуры

Создадим процедуру, которая будет добавлять новую строку в таблицу Rang:

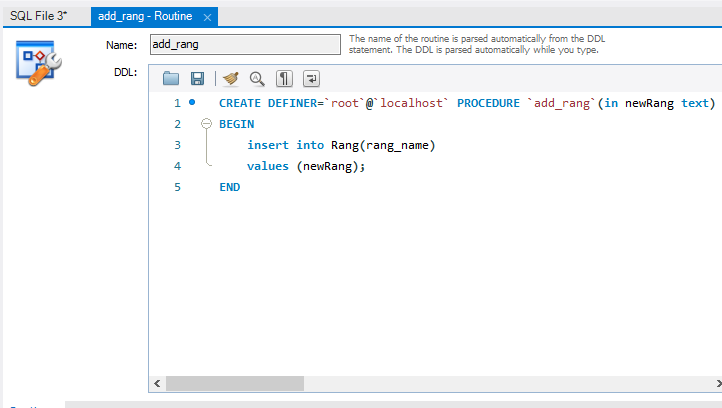


Рисунок 76 – Код процедуры

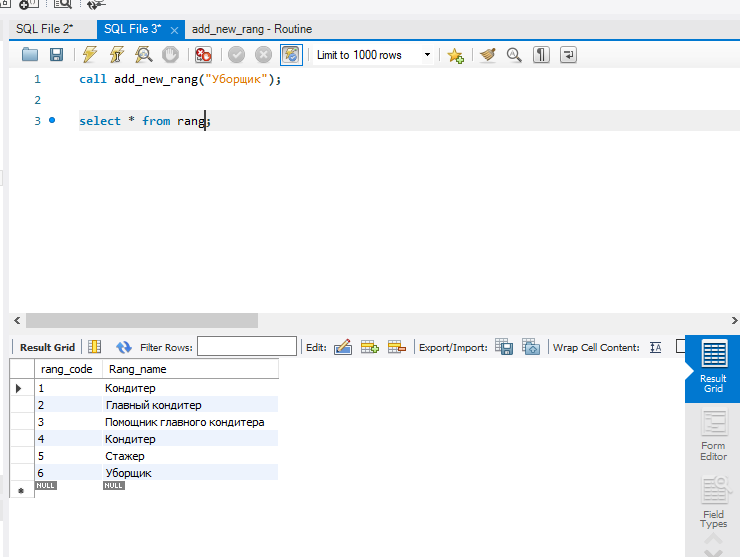


Рисунок 77 – Вызов процедуры

Создадим процедуру, которая будет удалять строчку по указанному id:

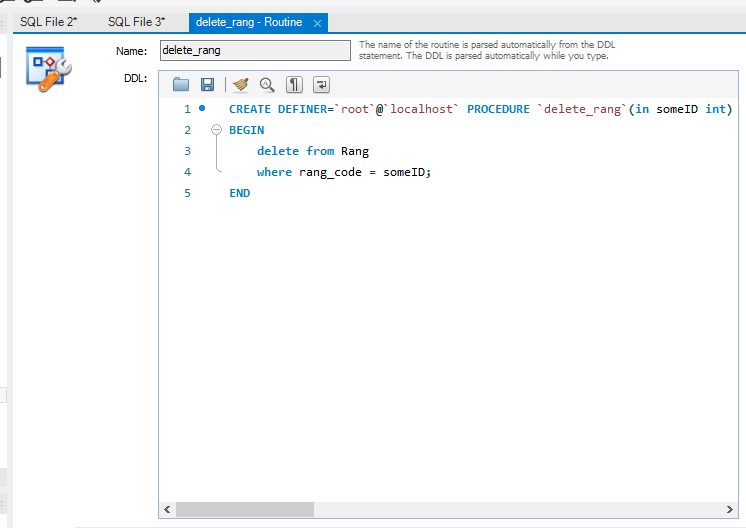


Рисунок 78 – Код процедуры

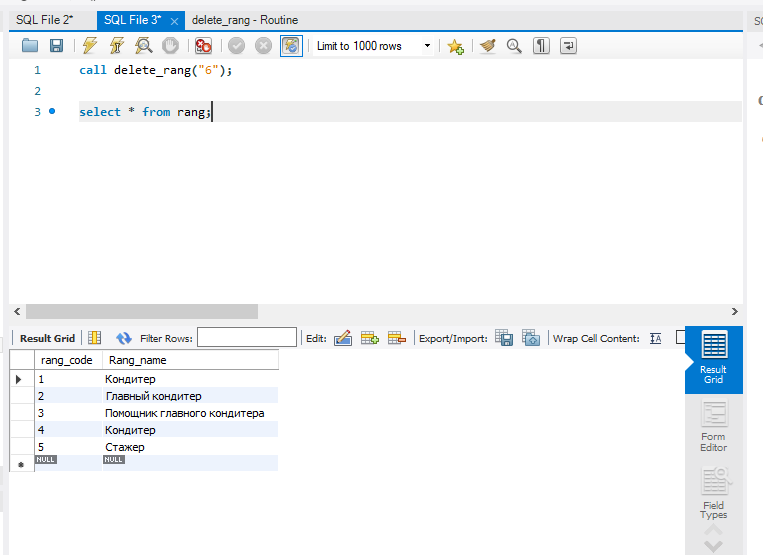


Рисунок 79 – Вызов процедуры

Создадим процедуру, которая будет обновлять название ранга:

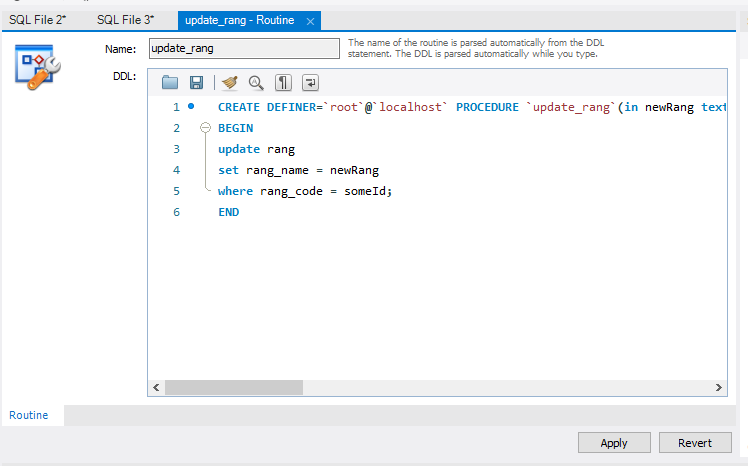


Рисунок 80 – Код процедуры

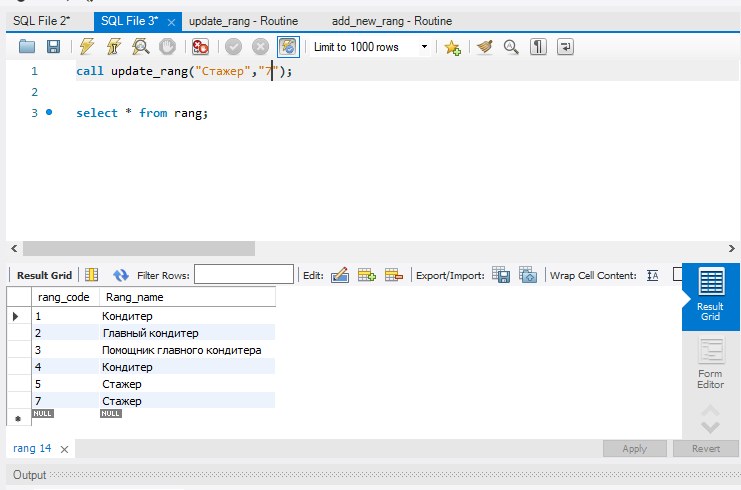


Рисунок 81 – Вызов процедуры

## Функции

Для создания функции используем тот же алгоритм, что и для процедур, но вместо папки «Stored Procedures» используем «Functions».

Создадим функцию, которая вставляет новую строку в таблицу и выводит ее id:

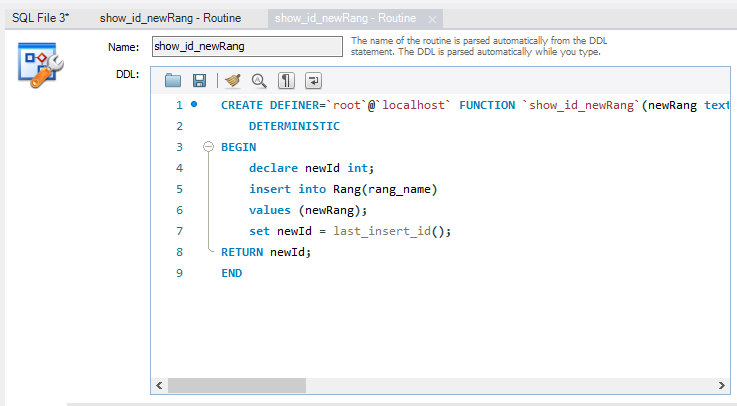


Рисунок 82 – Код функции

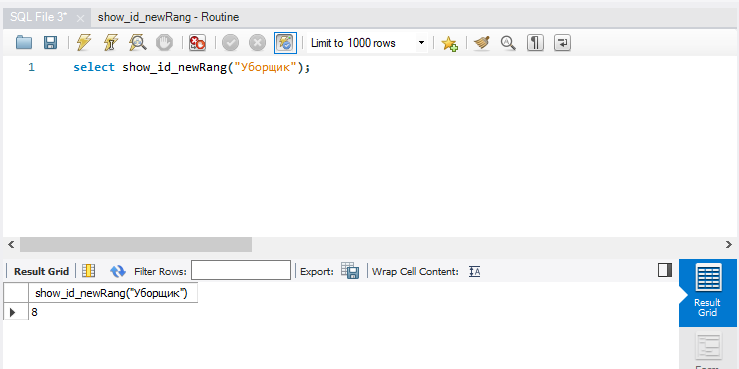


Рисунок 83 – Вызов функции

Создадим функцию, которая удаляет строку из таблицы и возвращает 1, если удаление проведено успешно:

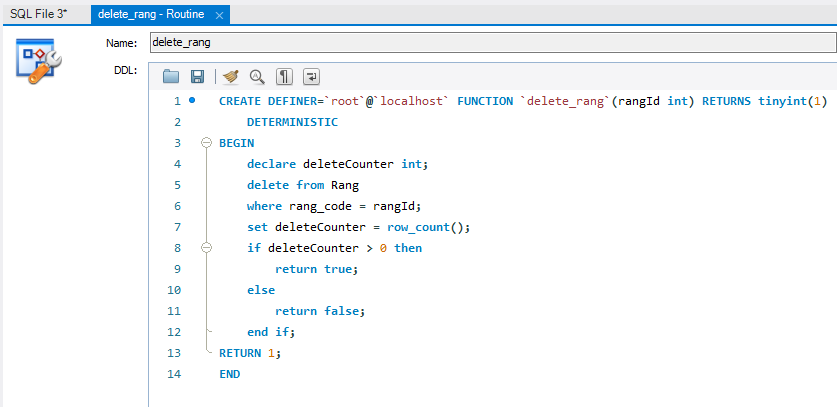


Рисунок 84 – Код функции

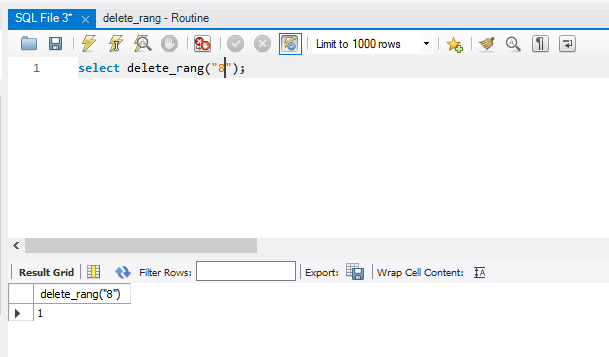


Рисунок 85 - Вызов функции

Создадим функцию, которая будет обновлять указанную строчку и возвращать 1, если обновление произошло:

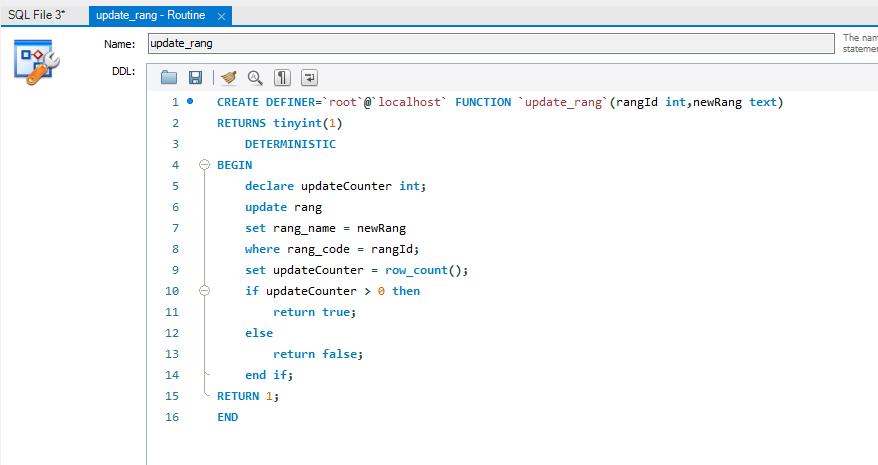


Рисунок 86 – Код функции

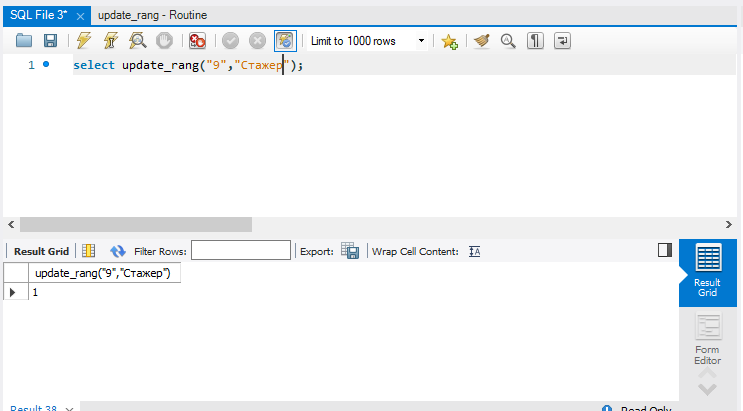


Рисунок 87 – Вызов функции

## Триггеры

Для того чтобы создать триггер в MySQL Workbench нужно нажать ПКМ по таблице, выбрать опцию «Alter Table» и перейти в раздел «Triggers».

Триггер — это набор операций, который срабатывает автоматически при указанном условии.

Напишем триггер, который в случае, когда в таблицу «Rang» вставляется новая строка и если названия ранга нет, заменяет null на «отсутствует»:

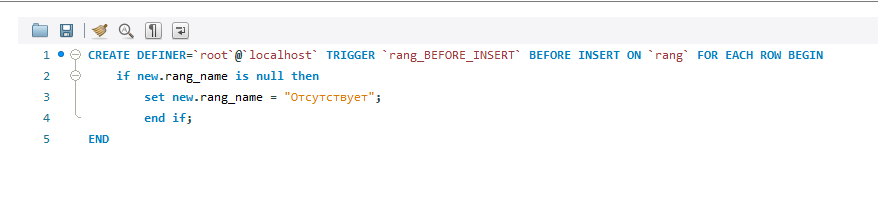


Рисунок 88 – Код триггера

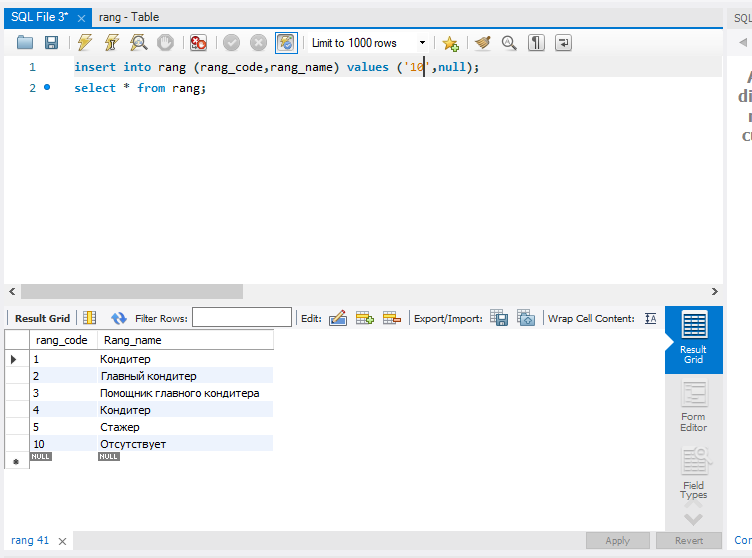


Рисунок 89 – Срабатывание триггера

Напишем триггер, который срабатывает перед удалением строки из таблицы. Если код удаляемой строки является «10», то отклонить удаление и вывести предупреждающее сообщение:

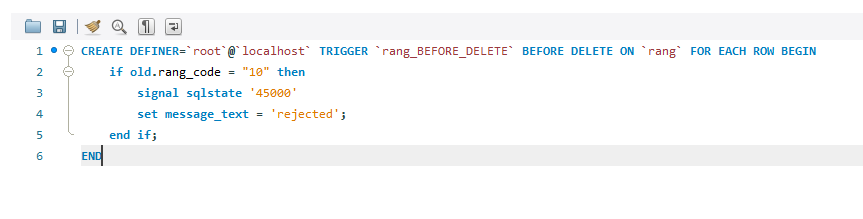


Рисунок 90 – Код триггера



Рисунок 91 – Срабатывание триггера

Напишем триггер, который срабатывает перед обновлением таблицы. Если значение указанное в rang\_code равно 10, то заменить это значение на null:

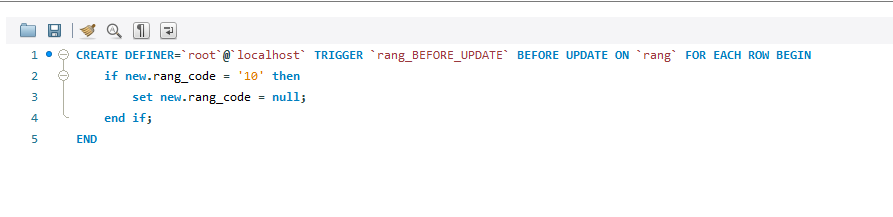


Рисунок 92 – Код триггера

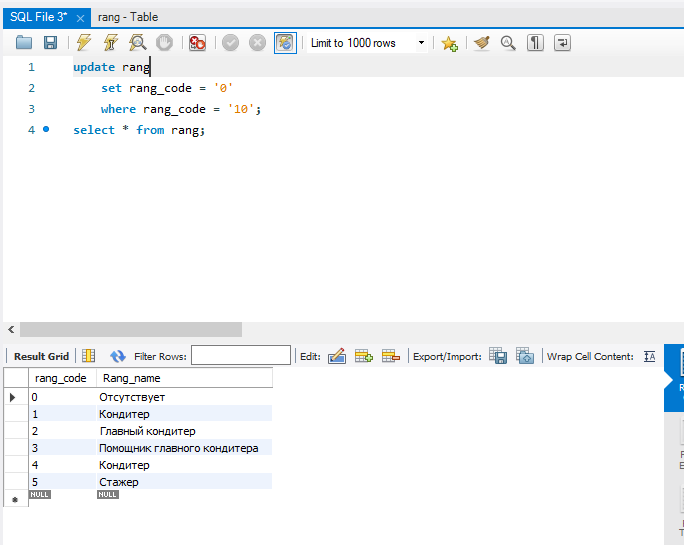


Рисунок 93 – Срабатывание триггера

# **Глава 6**

## Синтаксис оконных функций

Рассчитаем среднее значение id всех рангов, которые использованы в таблице rang с помощью оператора OVER()

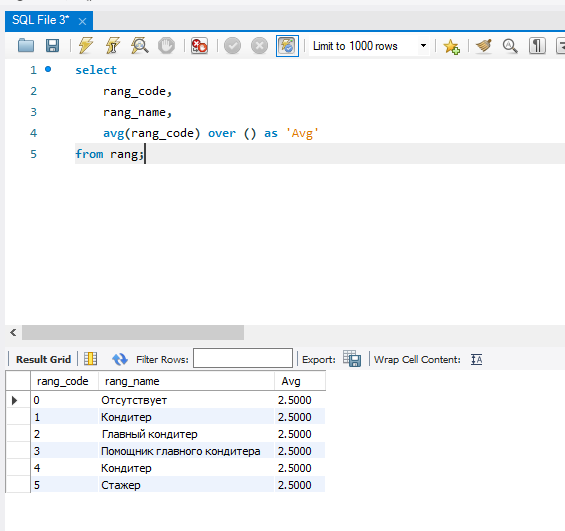


Рисунок 94 – OVER()

Рассчитаем среднее значения по столбцу rang\_code для каждой группы, сгруппированной по столбцу rang\_name с помощью оператора Partition by():

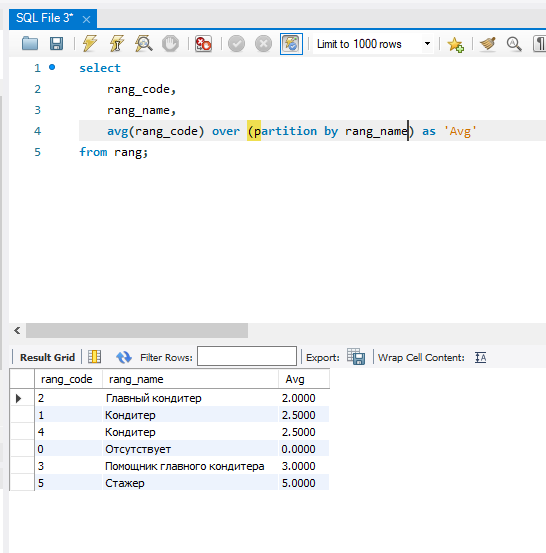


Рисунок 95 – Partition by()

Отсортируем данные по убыванию по столбцу rang\_code с помощью команды order by() desc:

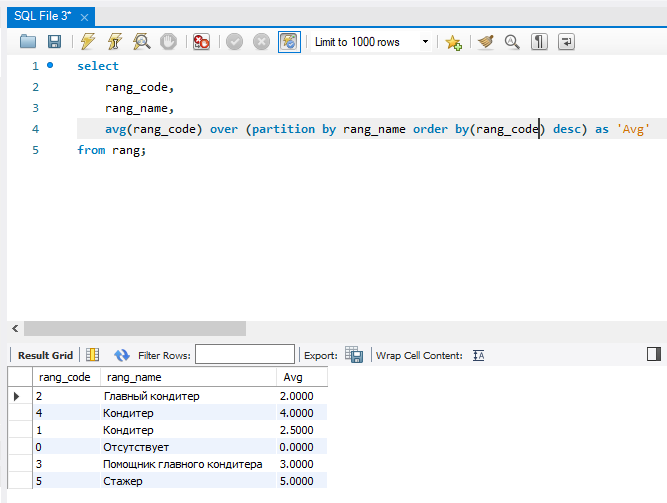


Рисунок 96 – Order by() desc

Укажем, чтобы среднее значение высчитывалось по двум строкам: по первой и последующей за ней.

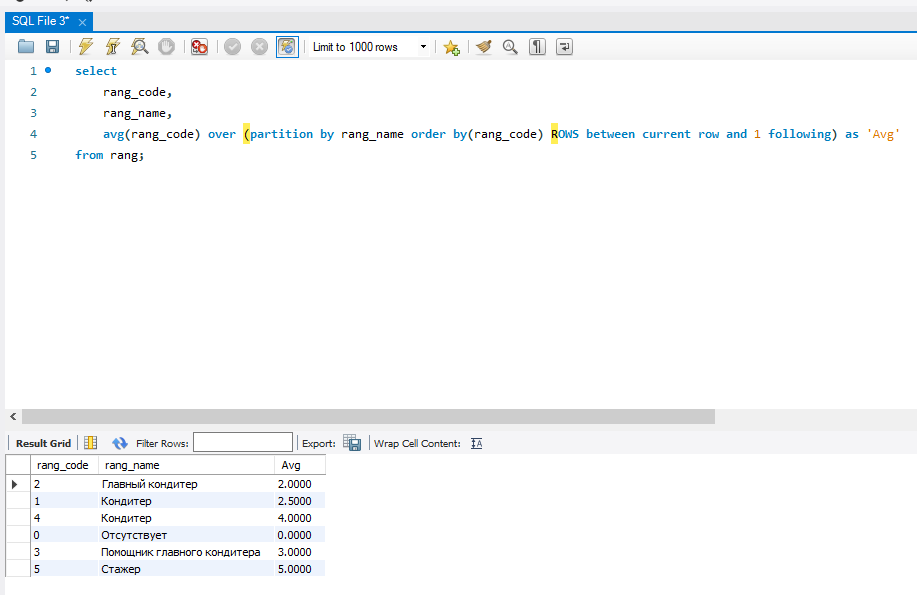


Рисунок 97 – Rows between current row and 1 following

## Агрегатные функции

Агрегатными функциями называются функции, которые выполняют арифметические вычисления на наборе данных и возвращают итоговое значение.

С помощью функции Count(\*) посчитаем количество рангов.

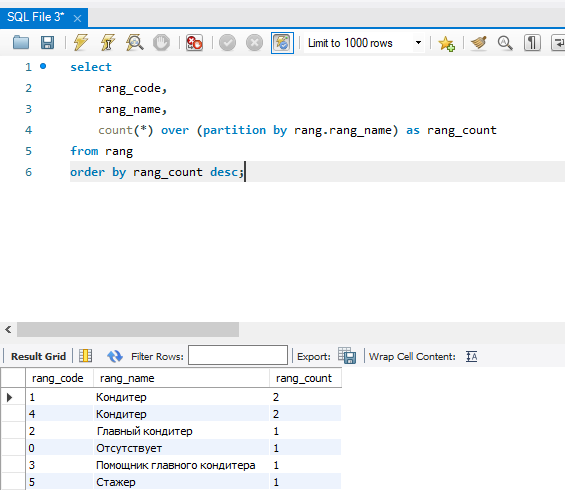


Рисунок 98 – Count(\*)

Найдем самые низкие rang\_code:

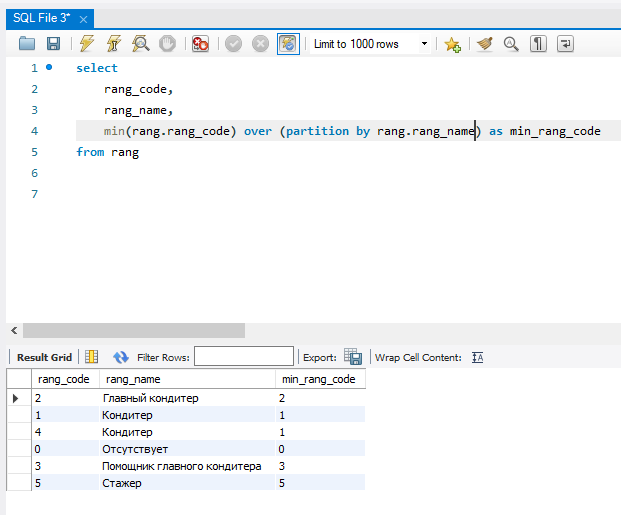


Рисунок 99 – Min()

Найдем самые большой rang\_code.

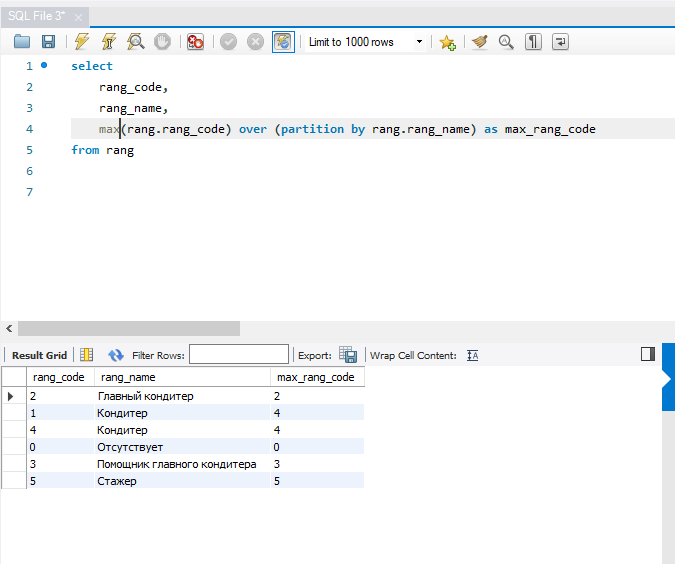


Рисунок 100 – Max()

## Ранжирующие функции

Ранжирующие функции — это функции, которые определяют ранг для каждой строки в окне.

Пронумеруем все строки по группам в таблице:

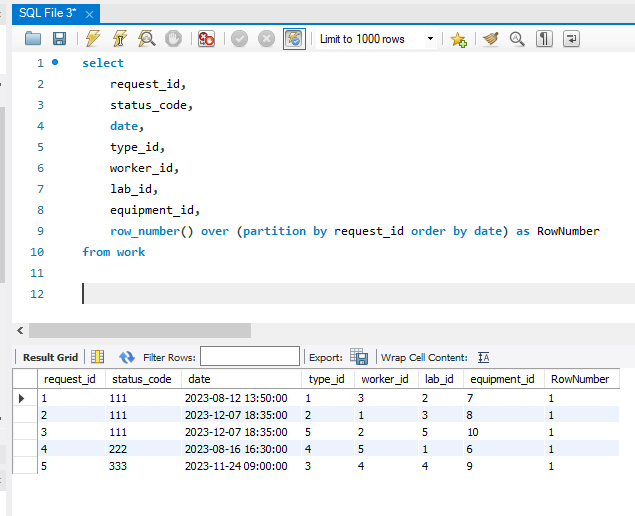


Рисунок 101 – Row\_number()

Определим ранг с помощью Rank():

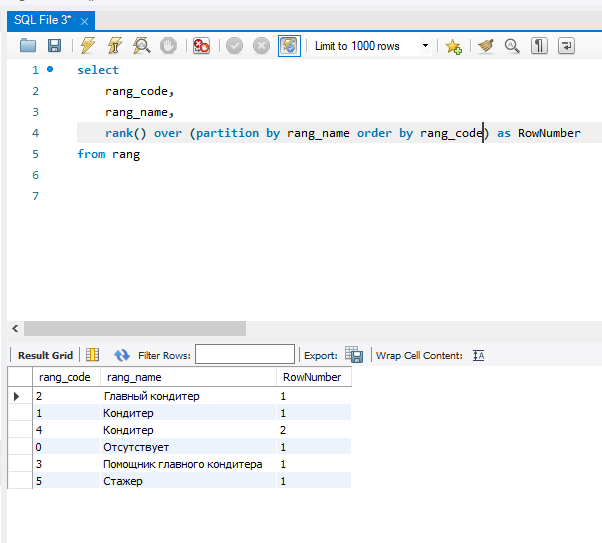


Рисунок 102 – Rank()

Определим ранги по кол-ву рангов:

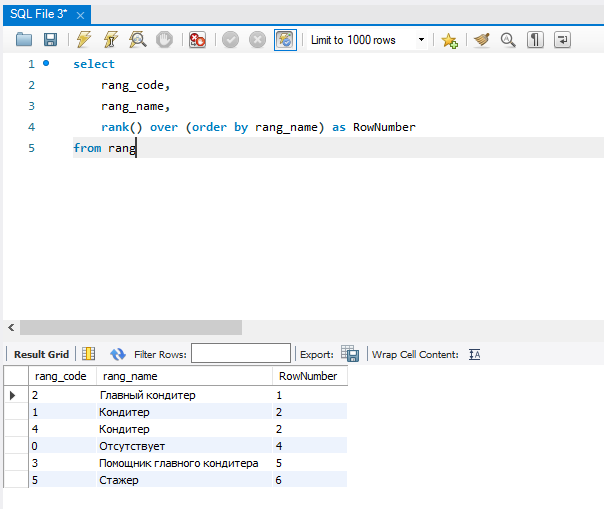


Рисунок 103 – dense\_rank()

Разделим на три группы по request\_id с помощью ntile():

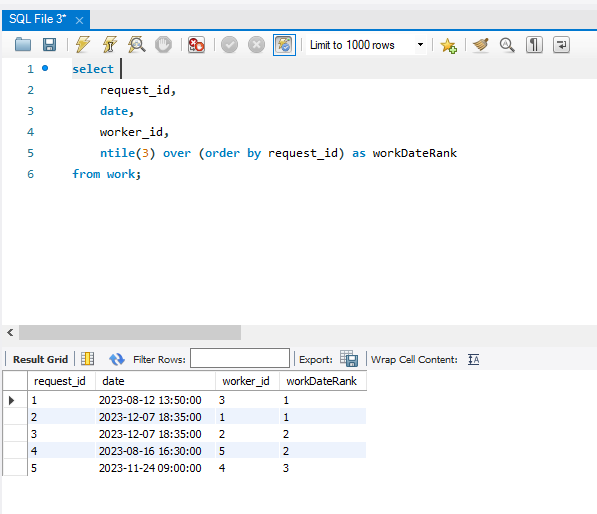


Рисунок 104 – ntile()

## Функции смещения

Функции смещения — это функции, которые позволяют перемещаться и обращаться к разным строкам в окне относительно текущей строки, а также обращаться к значениям в начале или в конце окна.

Используем команду Lag():

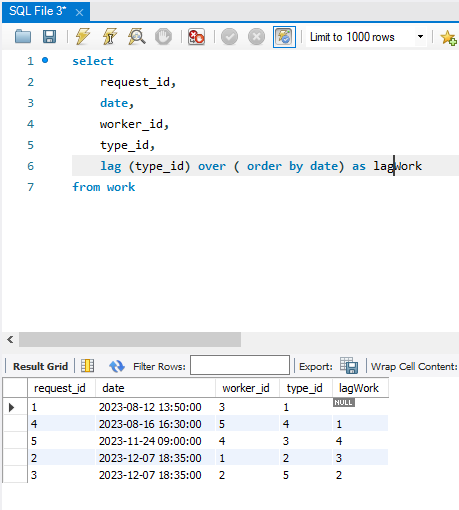


Рисунок 105 – lag()

Используем команду Lead():

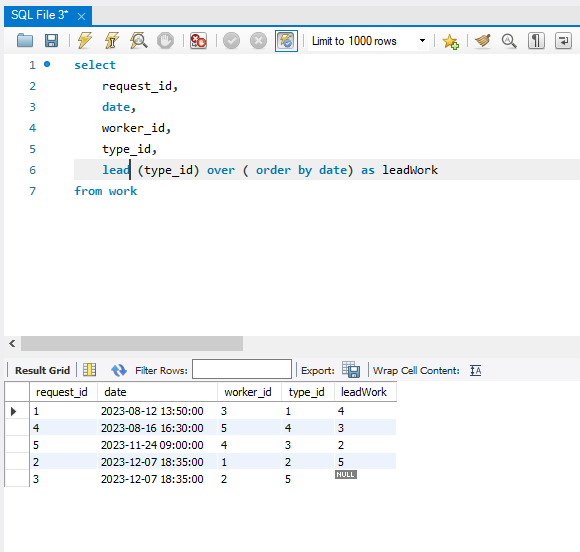


Рисунок 106 – Lead()

Используем команду first\_value():

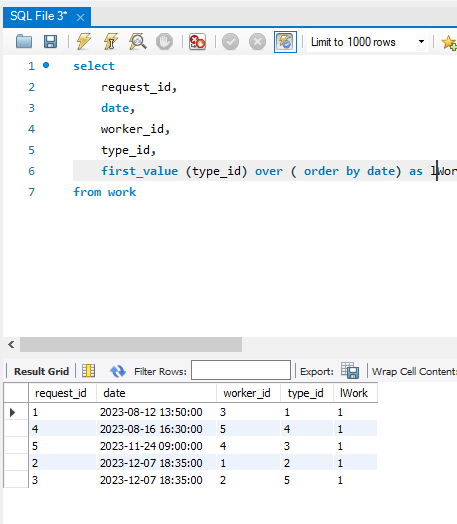


Рисунок 107 – first\_value()

Используем команду last\_value():

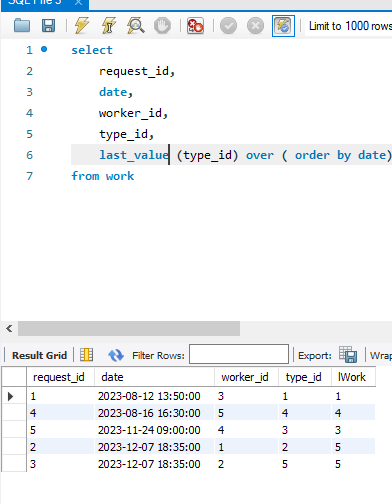


Рисунок 108 – last\_value()