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

**Институт искусственного интеллекта (ИИИ)**

**Кафедра промышленной информатики (ПИ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Разработка баз данных»

**Практические задания № 1-4**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИКБО-20-22, Шумахер Марк Евгеньевич* | (подпись) | |
| Преподаватель | *Баев Игорь Борисович* | (подпись) | |
| Отчет представлен | « » сентября 2024 г. | |  | |

Москва 2024 г.

Практическое задание 1

**Цель задания**: создать базу данных и таблицы в ней по выбранной теме, на основе разработанных моделей.

**Выполнение работы**

На основе физической модели базы данных создадим базу данных в mysql.

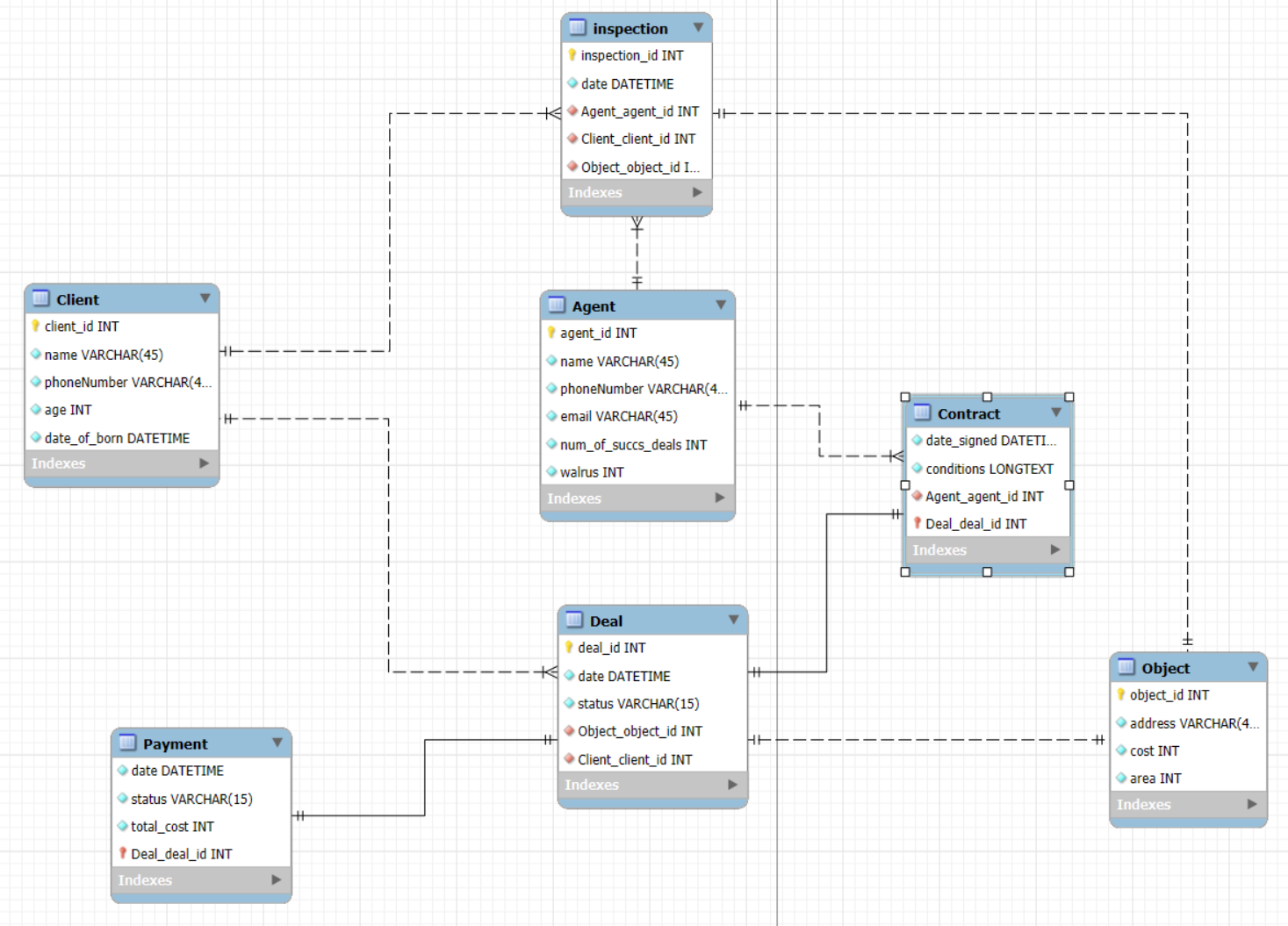


Рисунок 1 –

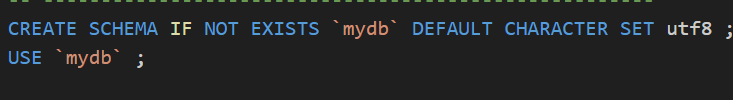


Рисунок 2 –

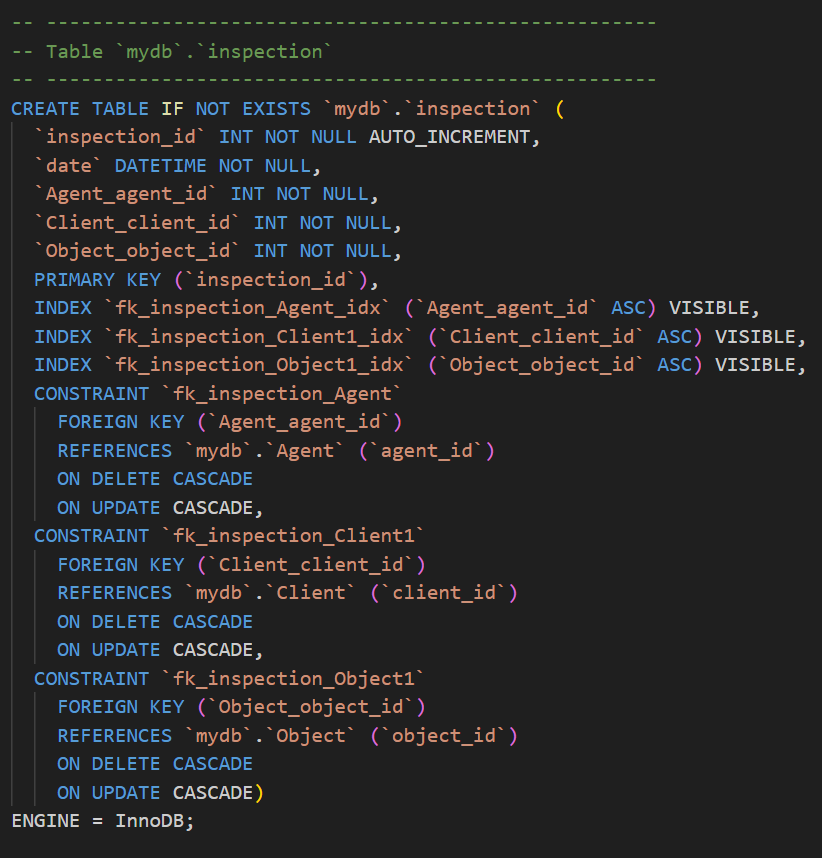


Рисунок 3 –

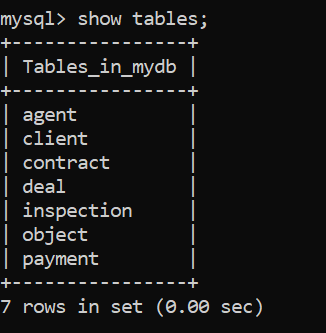


Рисунок 4 –

Ниже представлен полный скрипт для формирования схемы базы данных.

|  |
| --- |
| *-- -----------------------------------------------------*  *-- Schema mydb*  *-- -----------------------------------------------------*  *CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;*  *USE `mydb` ;*  *-- -----------------------------------------------------*  *-- Table `mydb`.`Agent`*  *-- -----------------------------------------------------*  *CREATE TABLE IF NOT EXISTS `mydb`.`Agent` (*  *`agent\_id` INT NOT NULL AUTO\_INCREMENT,*  *`name` VARCHAR(45) NOT NULL,*  *`phoneNumber` VARCHAR(45) NOT NULL,*  *`email` VARCHAR(45) NOT NULL,*  *`num\_of\_succs\_deals` INT NOT NULL,*  *`walrus` INT NOT NULL,*  *PRIMARY KEY (`agent\_id`))*  *ENGINE = InnoDB;*  *-- -----------------------------------------------------*  *-- Table `mydb`.`Client`*  *-- -----------------------------------------------------*  *CREATE TABLE IF NOT EXISTS `mydb`.`Client` (*  *`client\_id` INT NOT NULL AUTO\_INCREMENT,*  *`name` VARCHAR(45) NOT NULL,*  *`phoneNumber` VARCHAR(45) NOT NULL,*  *`age` INT NOT NULL,*  *`date\_of\_born` DATETIME NOT NULL,*  *PRIMARY KEY (`client\_id`))*  *ENGINE = InnoDB;*  *-- -----------------------------------------------------*  *-- Table `mydb`.`Object`*  *-- -----------------------------------------------------*  *CREATE TABLE IF NOT EXISTS `mydb`.`Object` (*  *`object\_id` INT NOT NULL AUTO\_INCREMENT,*  *`address` VARCHAR(45) NOT NULL,*  *`cost` INT NOT NULL,*  *`area` INT NOT NULL,*  *PRIMARY KEY (`object\_id`))*  *ENGINE = InnoDB;*  *-- -----------------------------------------------------*  *-- Table `mydb`.`inspection`*  *-- -----------------------------------------------------*  *CREATE TABLE IF NOT EXISTS `mydb`.`inspection` (*  *`inspection\_id` INT NOT NULL AUTO\_INCREMENT,*  *`date` DATETIME NOT NULL,*  *`Agent\_agent\_id` INT NOT NULL,*  *`Client\_client\_id` INT NOT NULL,*  *`Object\_object\_id` INT NOT NULL,*  *PRIMARY KEY (`inspection\_id`),*  *INDEX `fk\_inspection\_Agent\_idx` (`Agent\_agent\_id` ASC) VISIBLE,*  *INDEX `fk\_inspection\_Client1\_idx` (`Client\_client\_id` ASC) VISIBLE,*  *INDEX `fk\_inspection\_Object1\_idx` (`Object\_object\_id` ASC) VISIBLE,*  *CONSTRAINT `fk\_inspection\_Agent`*  *FOREIGN KEY (`Agent\_agent\_id`)*  *REFERENCES `mydb`.`Agent` (`agent\_id`)*  *ON DELETE CASCADE*  *ON UPDATE CASCADE,*  *CONSTRAINT `fk\_inspection\_Client1`*  *FOREIGN KEY (`Client\_client\_id`)*  *REFERENCES `mydb`.`Client` (`client\_id`)*  *ON DELETE CASCADE*  *ON UPDATE CASCADE,*  *CONSTRAINT `fk\_inspection\_Object1`*  *FOREIGN KEY (`Object\_object\_id`)*  *REFERENCES `mydb`.`Object` (`object\_id`)*  *ON DELETE CASCADE*  *ON UPDATE CASCADE)*  *ENGINE = InnoDB;*  *-- -----------------------------------------------------*  *-- Table `mydb`.`Deal`*  *-- -----------------------------------------------------*  *CREATE TABLE IF NOT EXISTS `mydb`.`Deal` (*  *`deal\_id` INT NOT NULL AUTO\_INCREMENT,*  *`date` DATETIME NOT NULL,*  *`status` VARCHAR(15) NOT NULL,*  *`Object\_object\_id` INT NOT NULL,*  *`Client\_client\_id` INT NOT NULL,*  *PRIMARY KEY (`deal\_id`),*  *INDEX `fk\_Deal\_Object1\_idx` (`Object\_object\_id` ASC) VISIBLE,*  *INDEX `fk\_Deal\_Client1\_idx` (`Client\_client\_id` ASC) VISIBLE,*  *CONSTRAINT `fk\_Deal\_Object1`*  *FOREIGN KEY (`Object\_object\_id`)*  *REFERENCES `mydb`.`Object` (`object\_id`)*  *ON DELETE CASCADE*  *ON UPDATE CASCADE,*  *CONSTRAINT `fk\_Deal\_Client1`*  *FOREIGN KEY (`Client\_client\_id`)*  *REFERENCES `mydb`.`Client` (`client\_id`)*  *ON DELETE CASCADE*  *ON UPDATE CASCADE)*  *ENGINE = InnoDB;*  *-- -----------------------------------------------------*  *-- Table `mydb`.`Payment`*  *-- -----------------------------------------------------*  *CREATE TABLE IF NOT EXISTS `mydb`.`Payment` (*  *`date` DATETIME NOT NULL,*  *`status` VARCHAR(15) NOT NULL,*  *`total\_cost` INT NOT NULL,*  *`Deal\_deal\_id` INT NOT NULL,*  *PRIMARY KEY (`Deal\_deal\_id`),*  *CONSTRAINT `fk\_Payment\_Deal1`*  *FOREIGN KEY (`Deal\_deal\_id`)*  *REFERENCES `mydb`.`Deal` (`deal\_id`)*  *ON DELETE CASCADE*  *ON UPDATE CASCADE)*  *ENGINE = InnoDB;*  *-- -----------------------------------------------------*  *-- Table `mydb`.`Contract`*  *-- -----------------------------------------------------*  *CREATE TABLE IF NOT EXISTS `mydb`.`Contract` (*  *`date\_signed` DATETIME NOT NULL,*  *`conditions` LONGTEXT NOT NULL,*  *`Agent\_agent\_id` INT NOT NULL,*  *`Deal\_deal\_id` INT NOT NULL,*  *PRIMARY KEY (`Deal\_deal\_id`),*  *INDEX `fk\_Contract\_Agent1\_idx` (`Agent\_agent\_id` ASC) VISIBLE,*  *INDEX `fk\_Contract\_Deal1\_idx` (`Deal\_deal\_id` ASC) VISIBLE,*  *CONSTRAINT `fk\_Contract\_Agent1`*  *FOREIGN KEY (`Agent\_agent\_id`)*  *REFERENCES `mydb`.`Agent` (`agent\_id`)*  *ON DELETE CASCADE*  *ON UPDATE CASCADE,*  *CONSTRAINT `fk\_Contract\_Deal1`*  *FOREIGN KEY (`Deal\_deal\_id`)*  *REFERENCES `mydb`.`Deal` (`deal\_id`)*  *ON DELETE CASCADE*  *ON UPDATE CASCADE)*  *ENGINE = InnoDB;*  *SET SQL\_MODE=@OLD\_SQL\_MODE;*  *SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;*  *SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;* |

Практическое задание 2

**Цель задания**: изучить и создать выборку и сортировку данных. Изучить и применить операторы для изменения данных в таблицах.

**Выполнение работы**

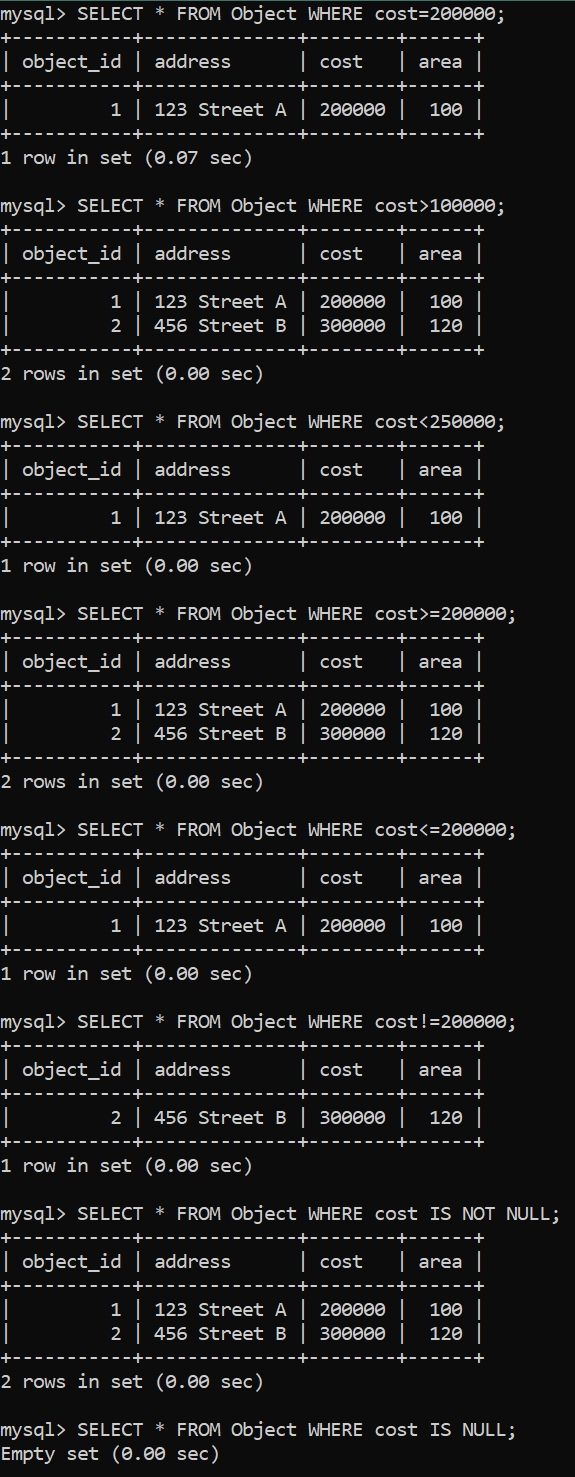


Рисунок 5 –

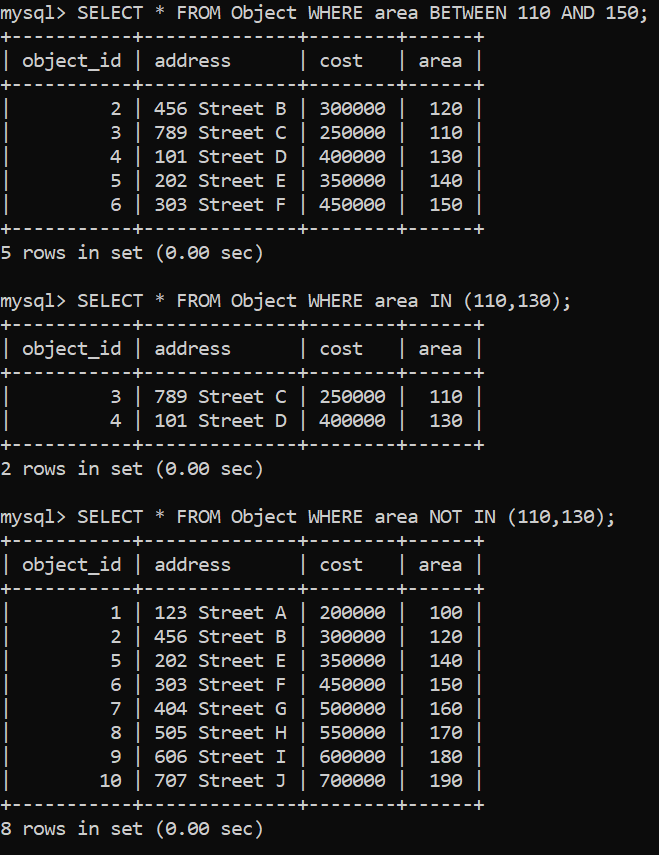


Рисунок 6 –

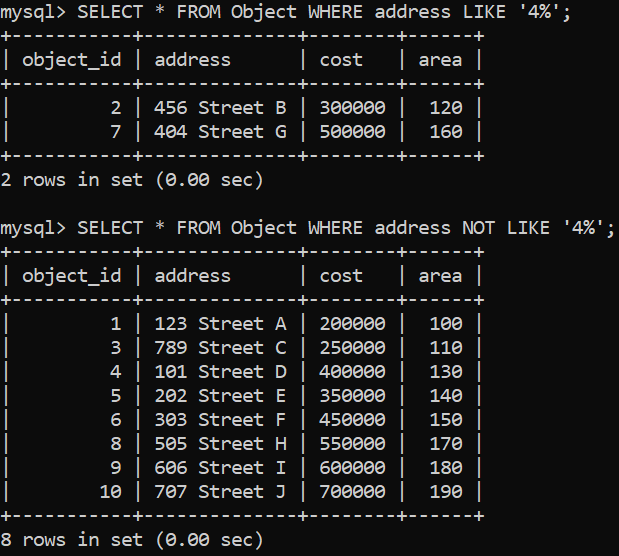


Рисунок 7 –

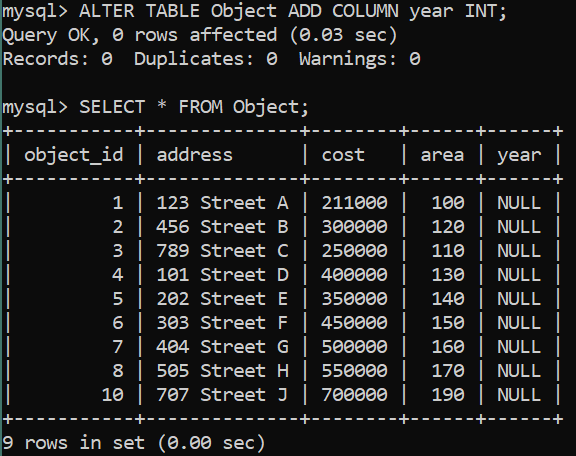


Рисунок 8 –

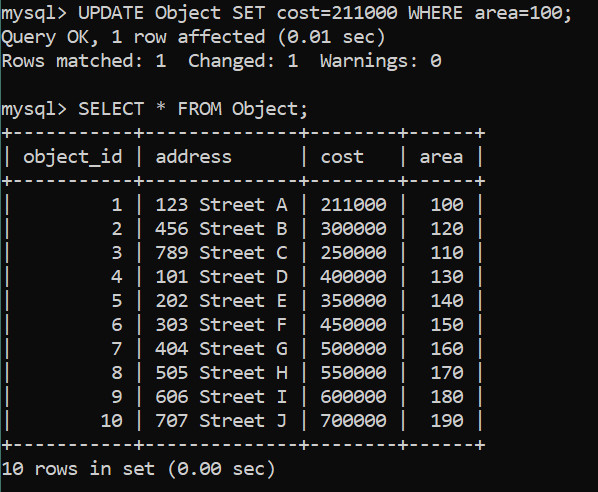


Рисунок 9 –

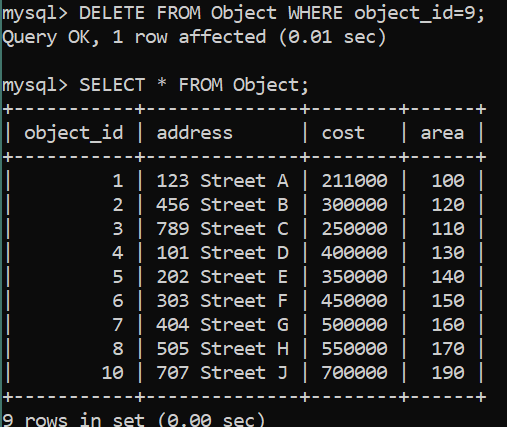


Рисунок 10 –

**Вывод**

В результате выполнения практической работы была изучена и создана выборка и сортировка данных. Изучены и применены операторы для изменения данных в таблицах.

Практическое задание 3

**Цель задания:** осуществить выборку данных по своей теме с помощью различных операторов. Изучить и применить к своей БД хранимые процедуры, функции и триггеры.

**Выполнение работы**

Для начала заполним базу данных.

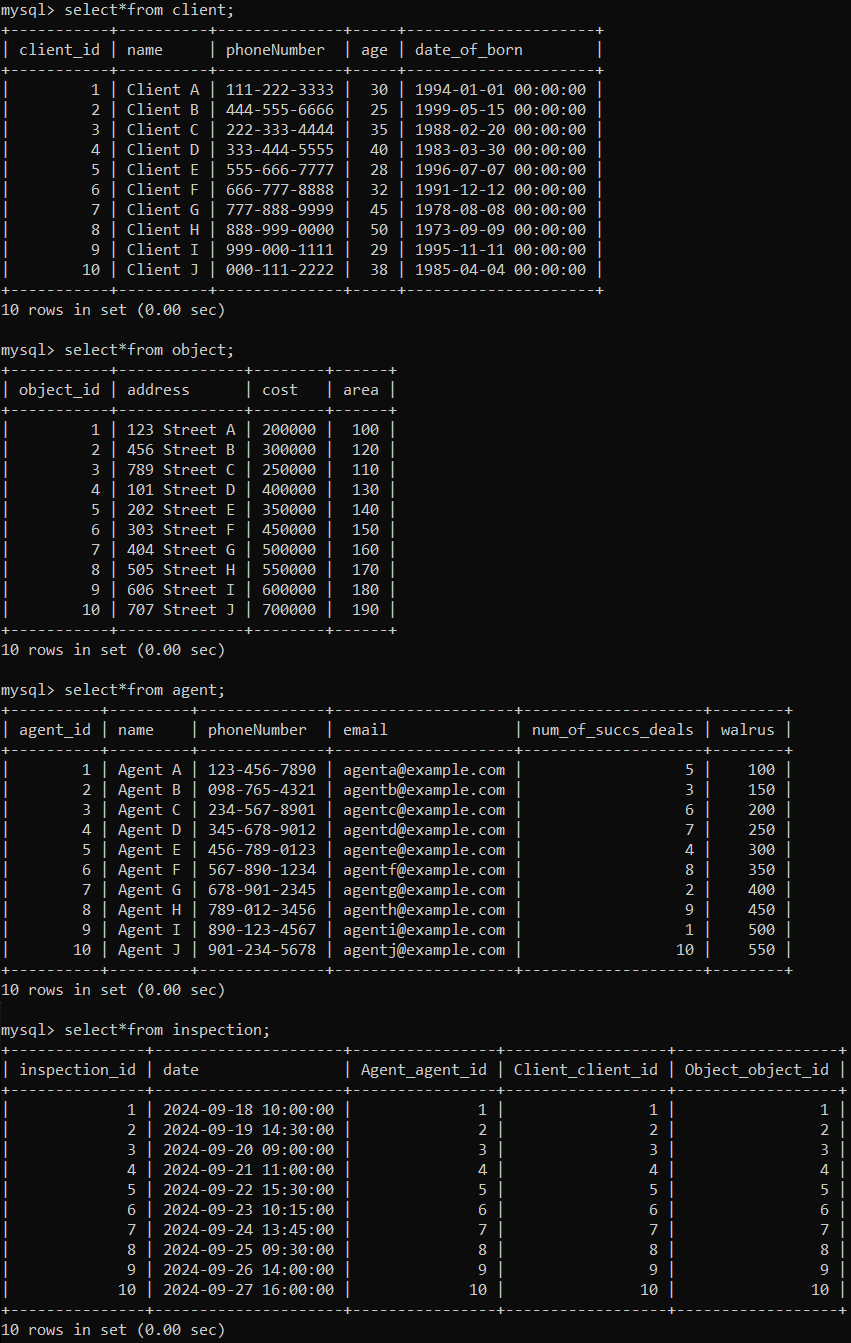


Рисунок 11 – Вывод данных

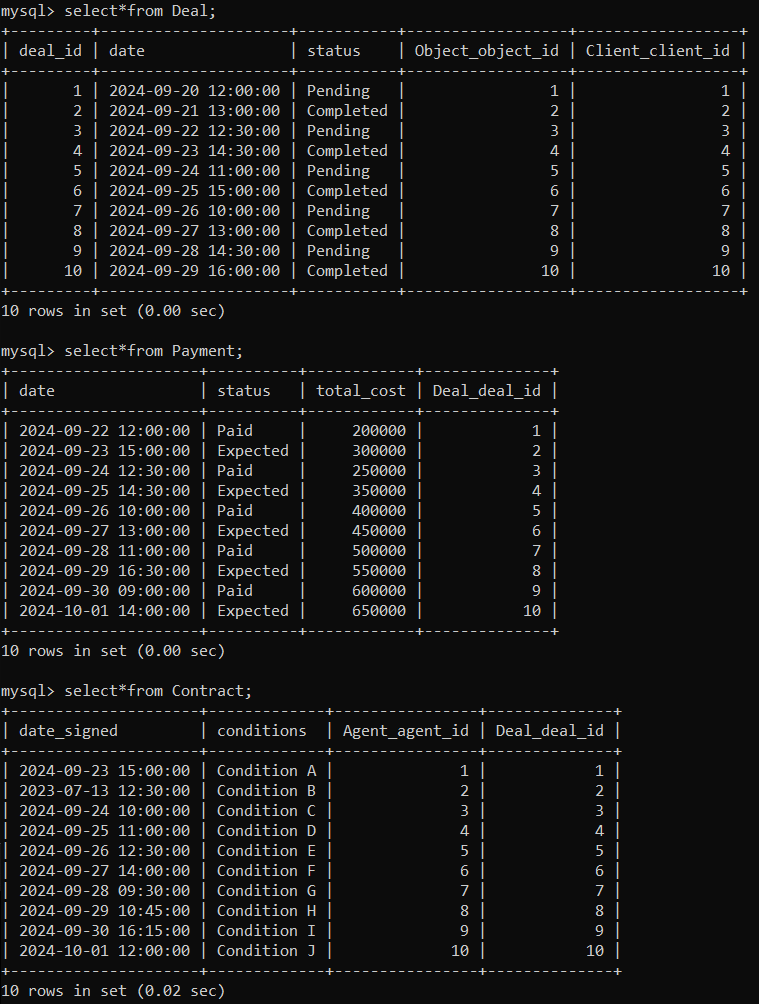


Рисунок 12 – Вывод данных

**Задание 1. Реляционная алгебра.**

Операция проекции:

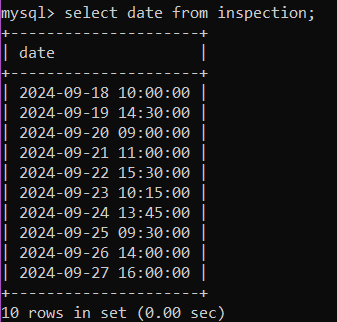


Рисунок 13 – Операция проекции

Операция селекции:

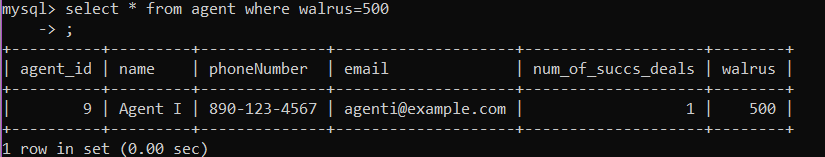


Рисунок 14 – Операция селекции

Операции соединения:

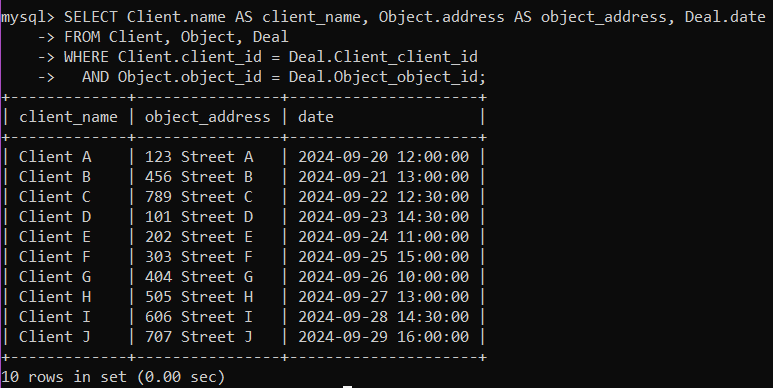


Рисунок 15 – Операции соединения

Операция объединения:

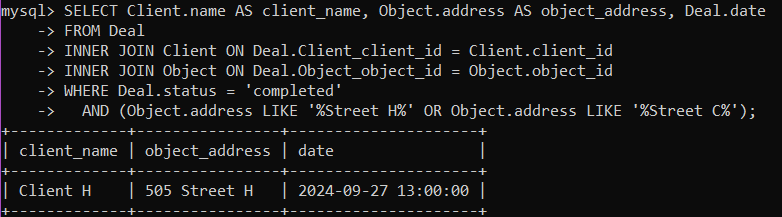


Рисунок 16 – Операция объединения

Операция пересечения (выводим те сделки, которые были с объектами на улицах начинающихся с “street” и “street h” с тем же клиентом):

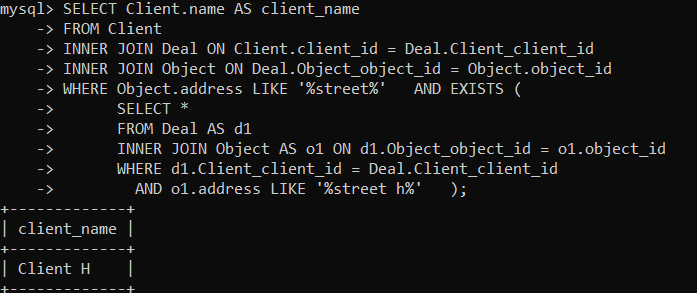


Рисунок 17 – Операция пересечения

Операция разности (выводим те сделки, которые были с объектами на улицах начинающихся с “street”, но не “street h” для одного и того же клиента):

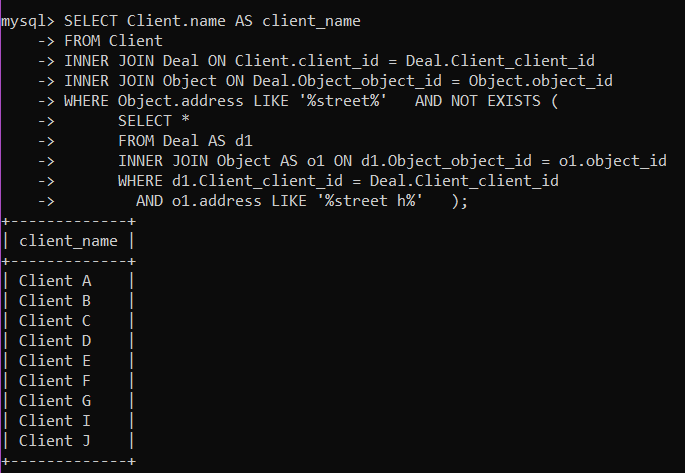


Рисунок 18 – Операция разности

Операция группировки(посчитаем общее количество сделок, совершенных каждым клиентом, с указанием минимальной и максимальной стоимости объектов, участвующих в этих сделках):

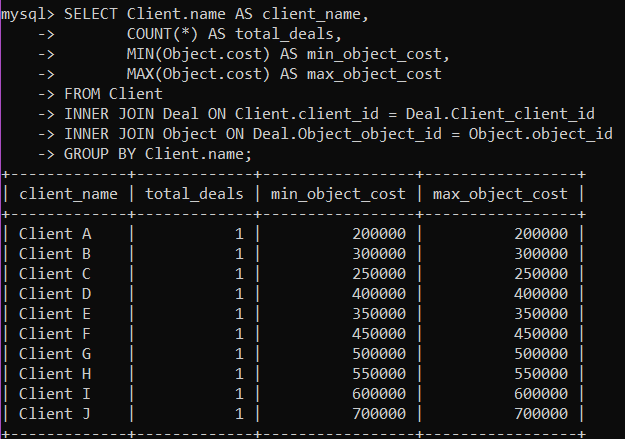


Рисунок 19 – Операция группировки

Операция сортировки (запрос будет выводить всех агентов, которым были назначены сделки, и сортировать их по количеству сделок в порядке убывания):

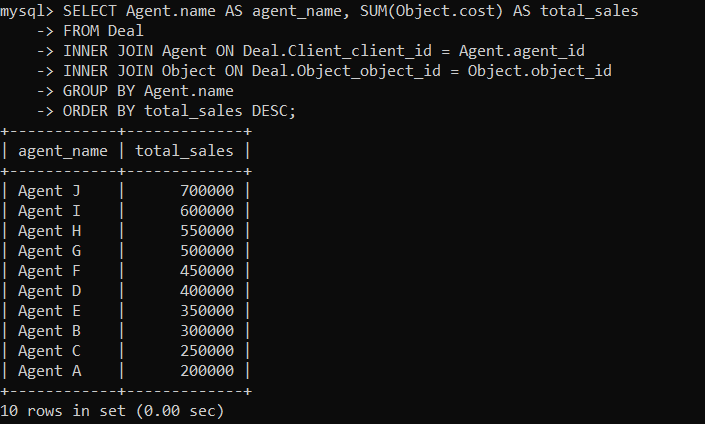


Рисунок 20 – Операция сортировки

Операция деления: этот запрос позволяет найти только тех клиентов, которые заключили сделки по всем объектам, существующим в таблице Object.

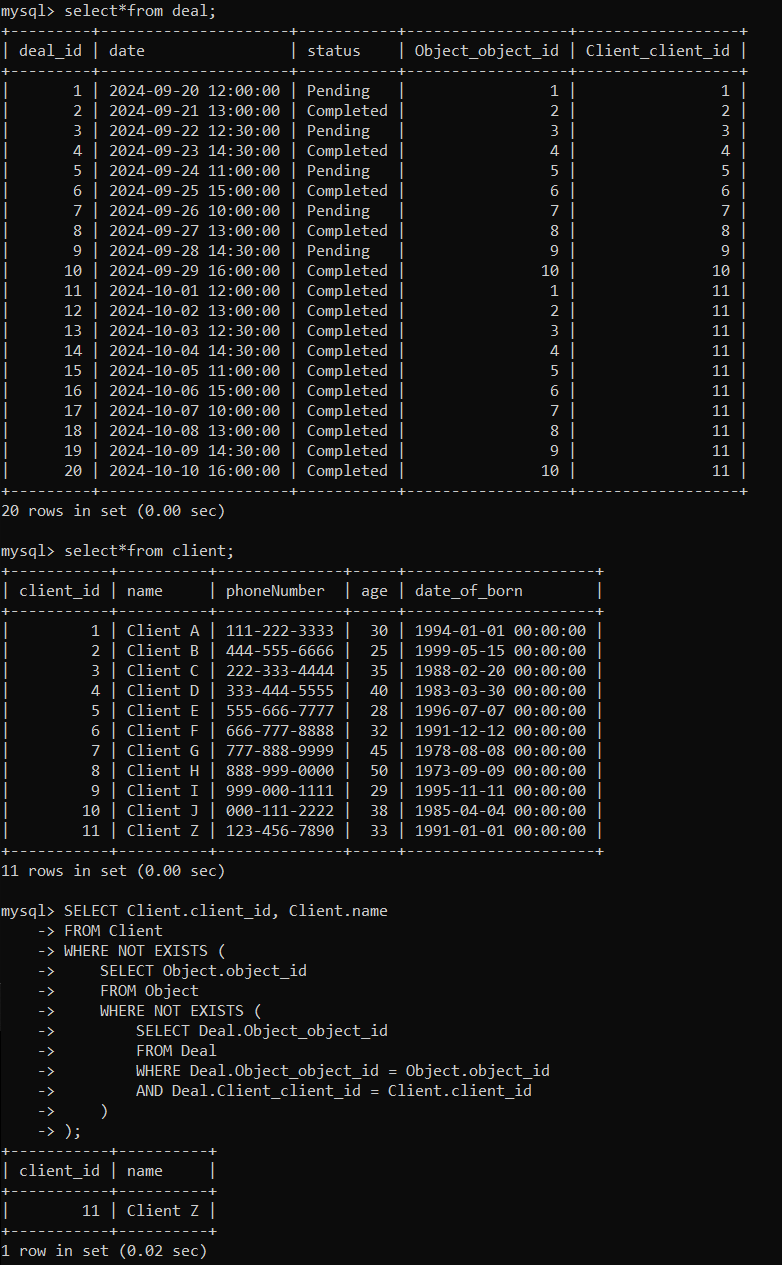


Рисунок 21 – Операция деления

**Задание 2. ХРАНИМЫЕ ПРОЦЕДУРЫ, ФУНКЦИИ И ТРИГГЕРЫ**

Создадим процедуру добавления нового клиента:

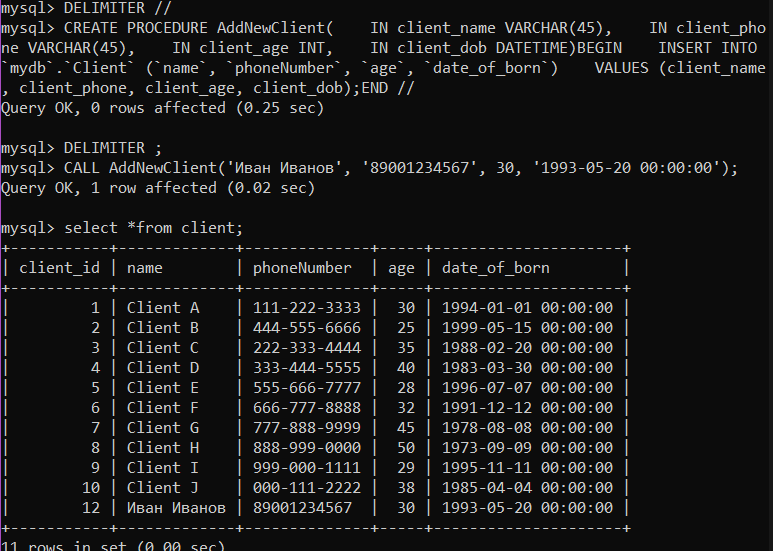


Рисунок 22 – Процедура добавления нового клиента

Создадим процедуру для удаления клиента по имени и фамилии:

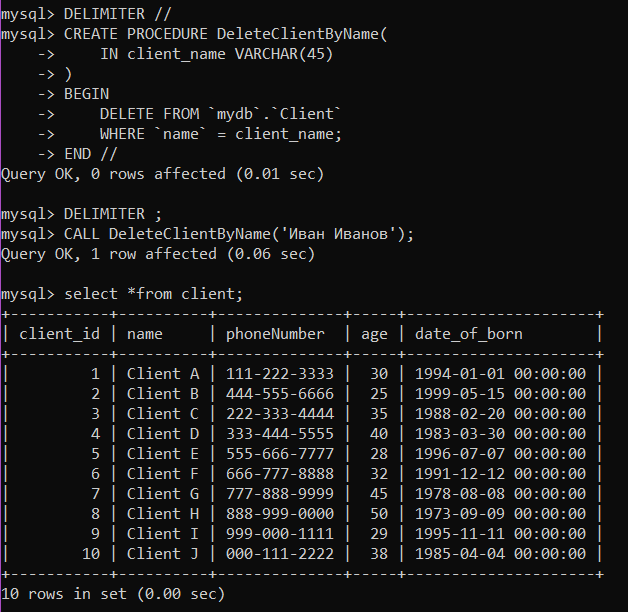


Рисунок 23 – Процедура для удаления клиента по имени и фамилии

Создадим функцию для вычисления общей стоимости всех объектов в базе данных:

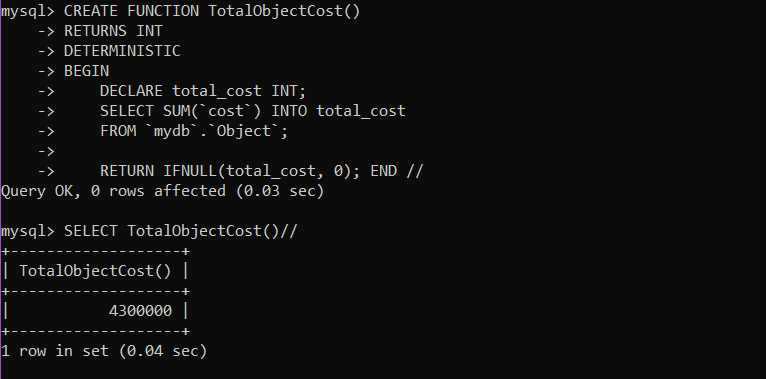


Рисунок 24 – Функция для вычисления общей стоимости

Функция, которая находит средний возраст всех клиентов в базе данных:

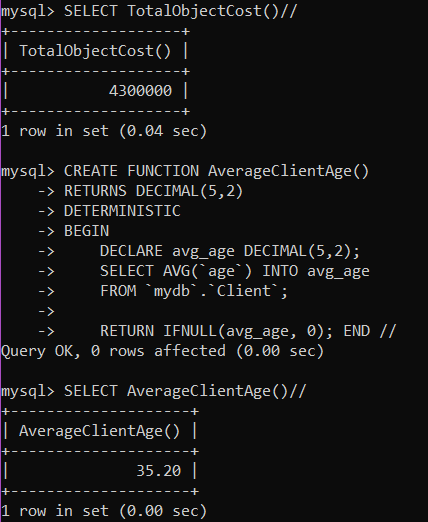


Рисунок 25 – Функция поиска среднего возраста

Напишем 5 триггеров для базы данных:

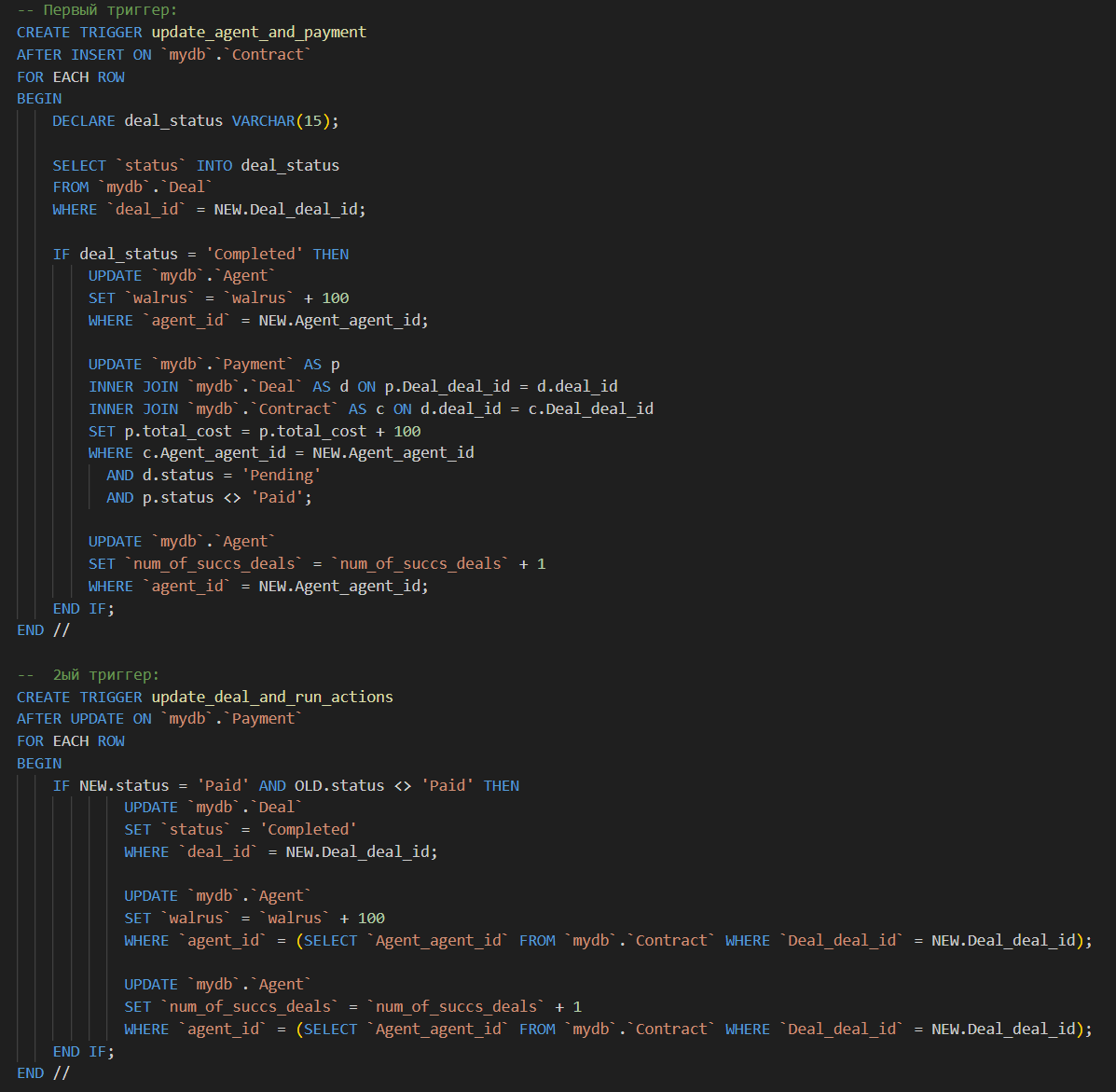


Рисунок 26 – Создание триггеров

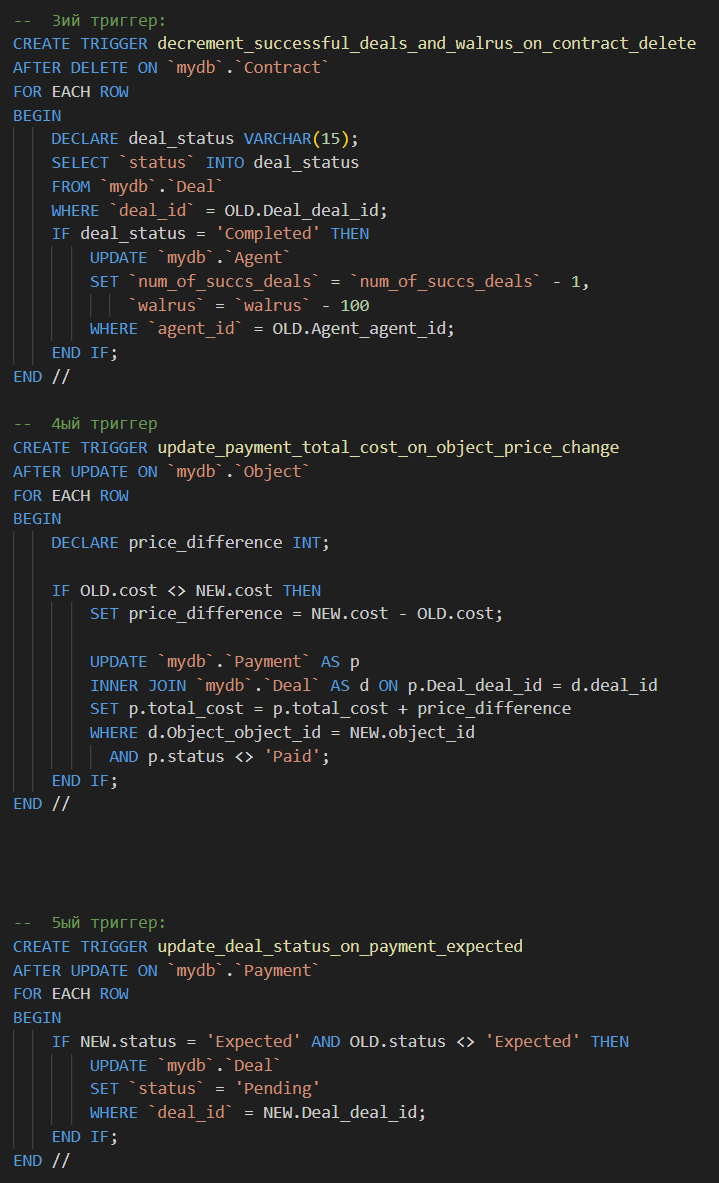


Рисунок 27 – Создание триггеров

Проверим работу каждого триггера, начнем с триггера который при создании контракта с условием, что он связан с завершенной сделкой, увеличивает кол-во успешных сделок агента и повышает стоимость работы агента, а также увеличивает финальную стоимость сделки:

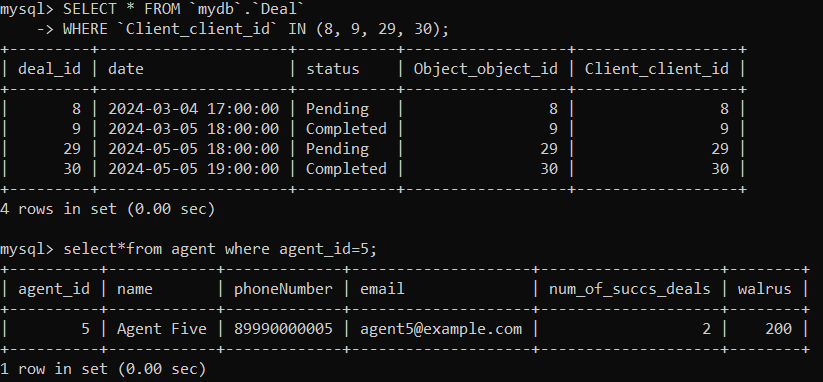


Рисунок 28 – Работа триггера 1

Проверим работу триггера, который при установке статуса платежа на “оплачено”, делает статус связанной с ним сделки “завершенным”, а также увеличивает кол-во успешных сделок и стоимость работы агента:

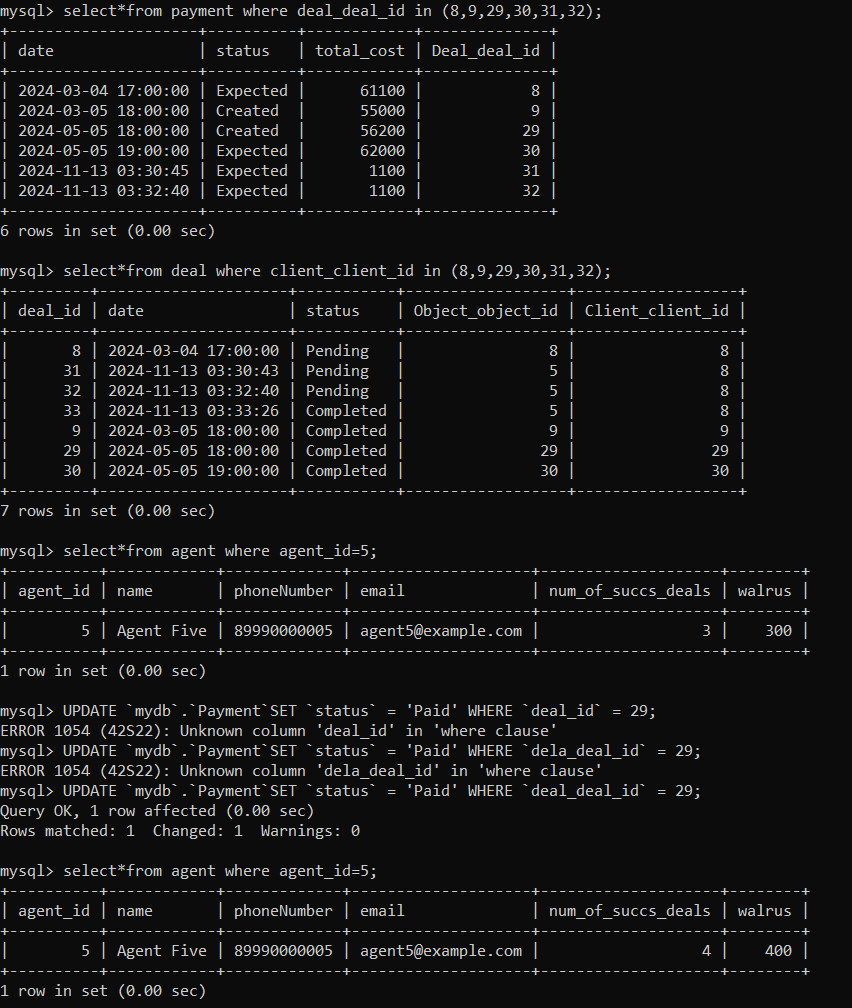


Рисунок 29 – Работа триггера 2

Проверим работу триггера, который делает обратный результат от первого триггера при удалении контракта:

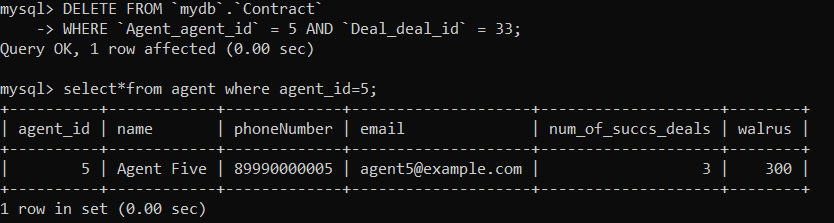


Рисунок 30 – Работа триггера 3

Проверим работу триггера, который при обновлении стоимости объекта, меняет финальную стоимость сделки в платеже:

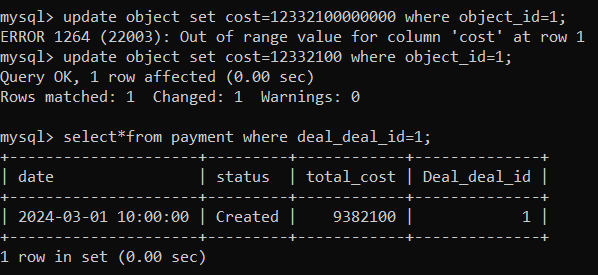


Рисунок 31 – Работа триггера 4

Проверим триггер, который устанавливает статус сделки на “рассмотрение” при изменении статуса платежа на “ожидание”:

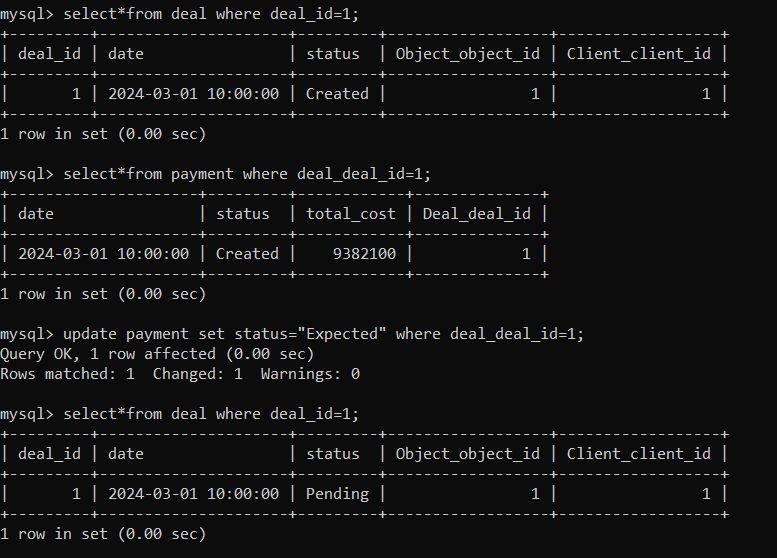


Рисунок 32 – Работа триггера 5

**Вывод**

В результате выполнения практической работы была изучена и создана выборка и сортировка данных. Изучены и применены операторы для изменения данных в таблицах.

Практическое задание 4

**Цель задания**: изучить и применить к своей БД оконные функции.

**Выполнение работы**

**Задание 1. Агрегатные функции.**

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

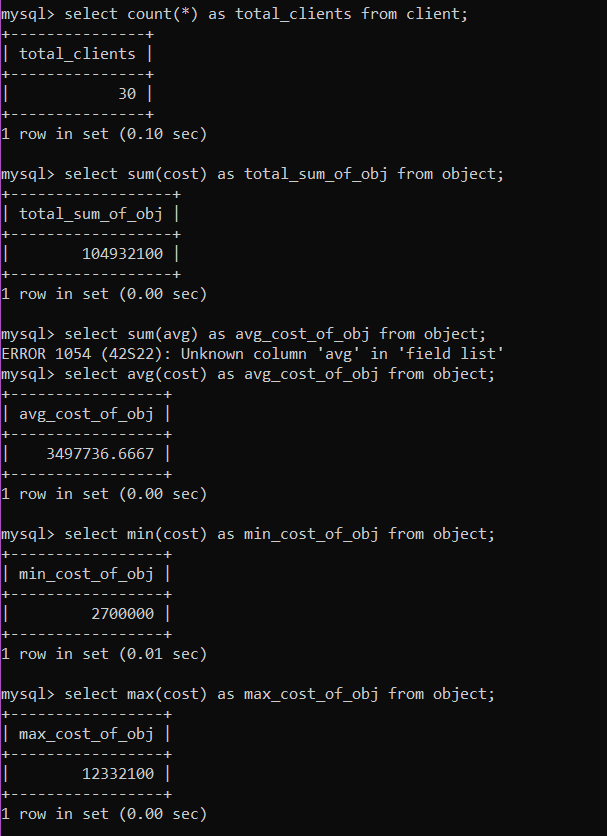


Рисунок 33 – Использование агрегатных функций

**Задание 2. Ранжирующие функции.**

Ранжирующие функции — это функции, которые определяют ранг для каждой строки в окне. Например, их можно использовать для присвоения порядковых номеров или для составления рейтинга.

Получим список сделок, отсортированных по дате, с нумерацией для каждого клиента.

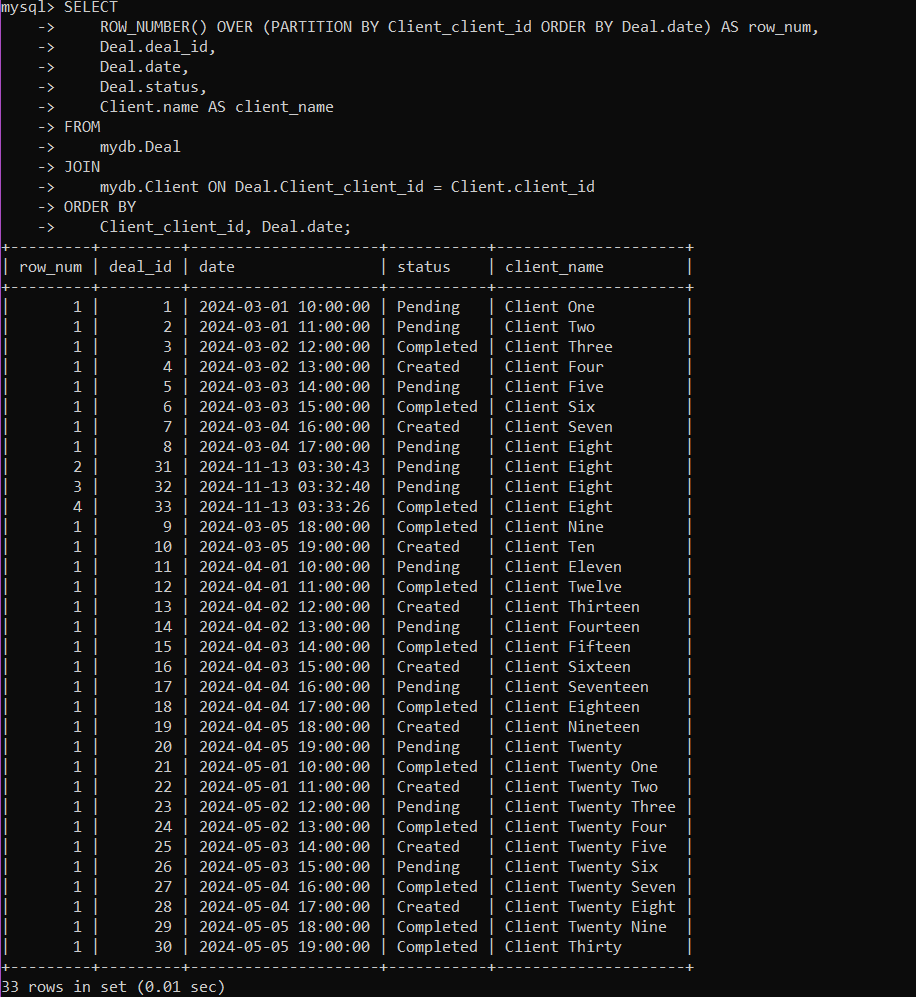


Рисунок 34 – Использование ранжирующей функции

Отсортируем сделки по стоимости (total\_cost) и присвоим каждому ряду ранг в зависимости от стоимости сделки с использованием RANK и DENSE\_RANK.

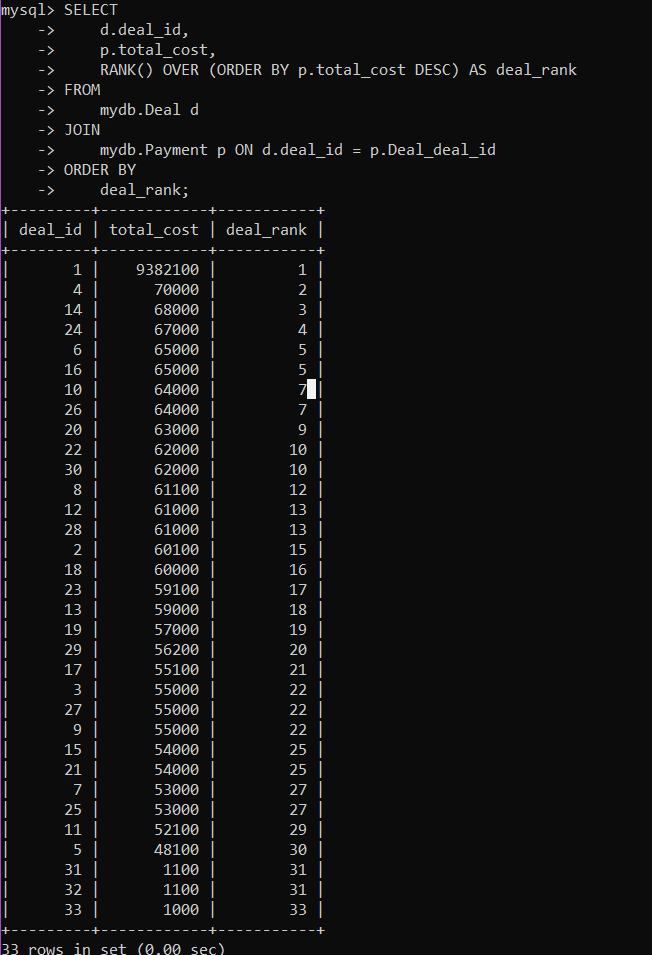


Рисунок 35 – Использование ранжирующей функции

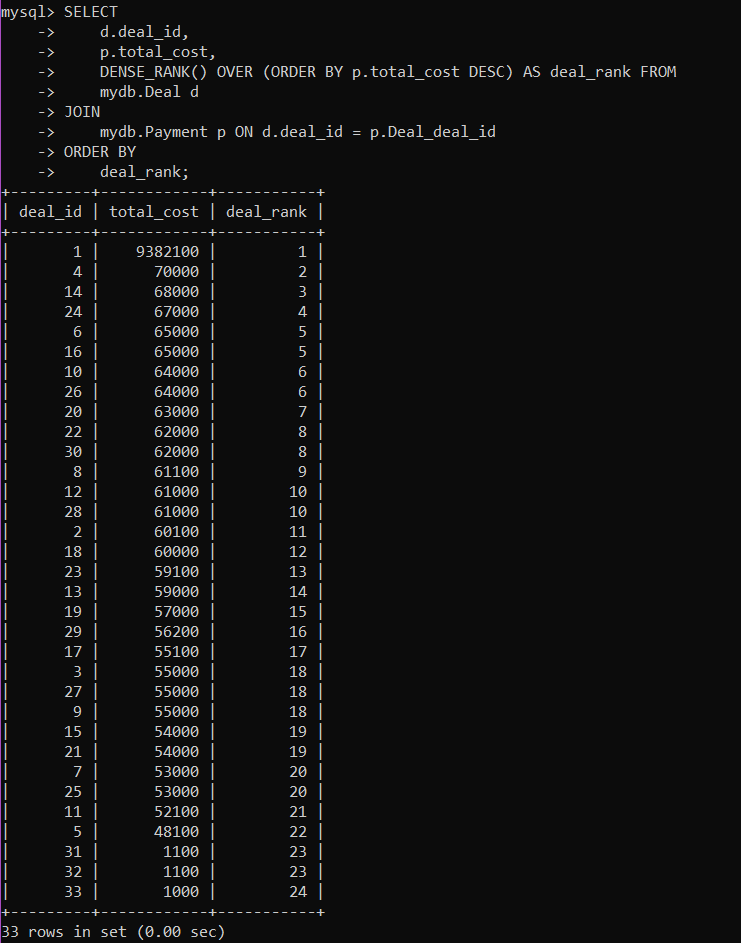


Рисунок 36 – Использование ранжирующей функции

Разделим все сделки на 4 группы по сумме total\_cost в таблице Payment, где сумма total\_cost будет использоваться для определения групп (в порядке убывания).

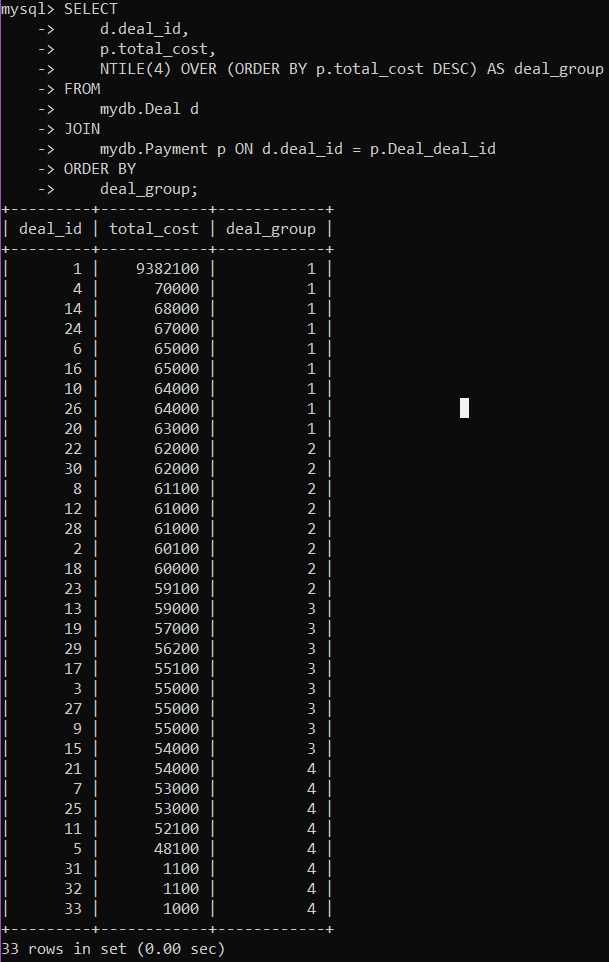


Рисунок 37 – Использование ранжирующей функции

**Задание 3. 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.

Получим данные из предыдущей и следующей строки для каждой сделки в пределах одного клиента.

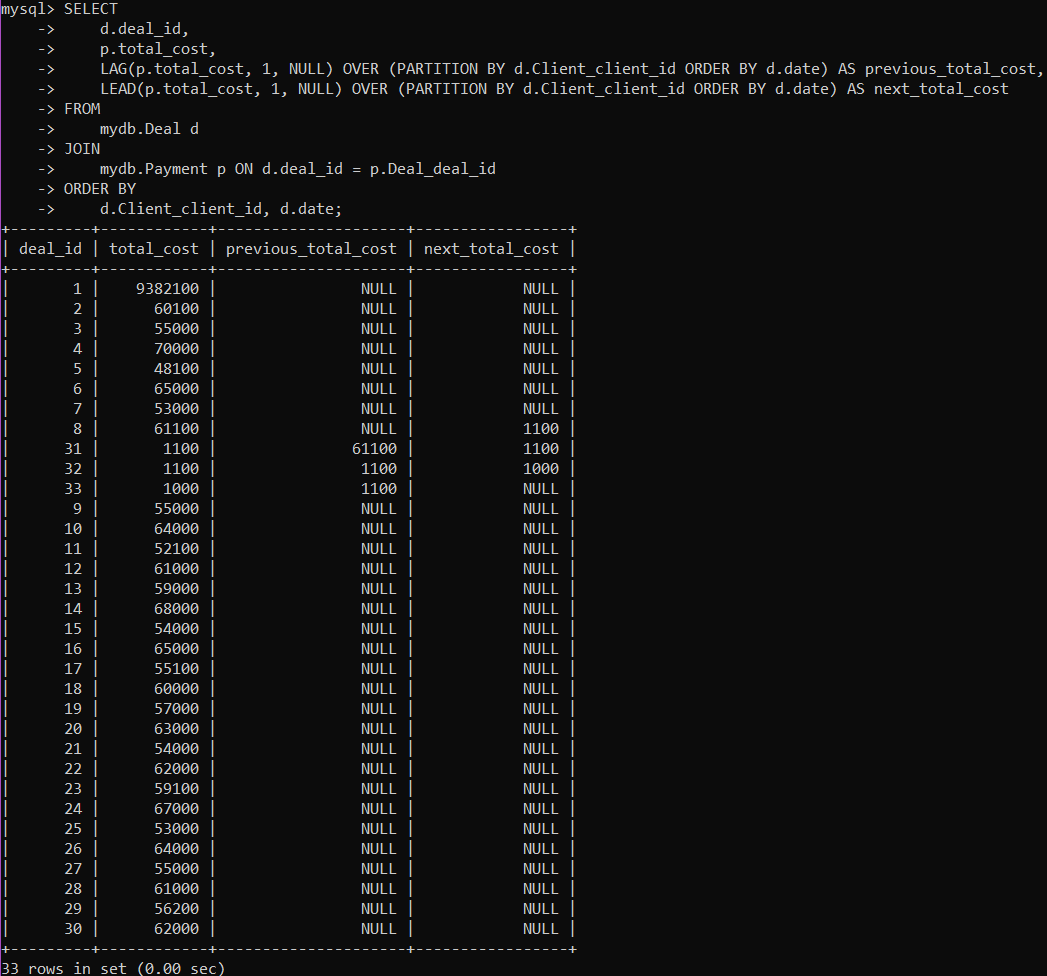


Рисунок 38 – Использование функций смещения

Напишем запрос, который покажет для каждого агента его первый и последний контракт.

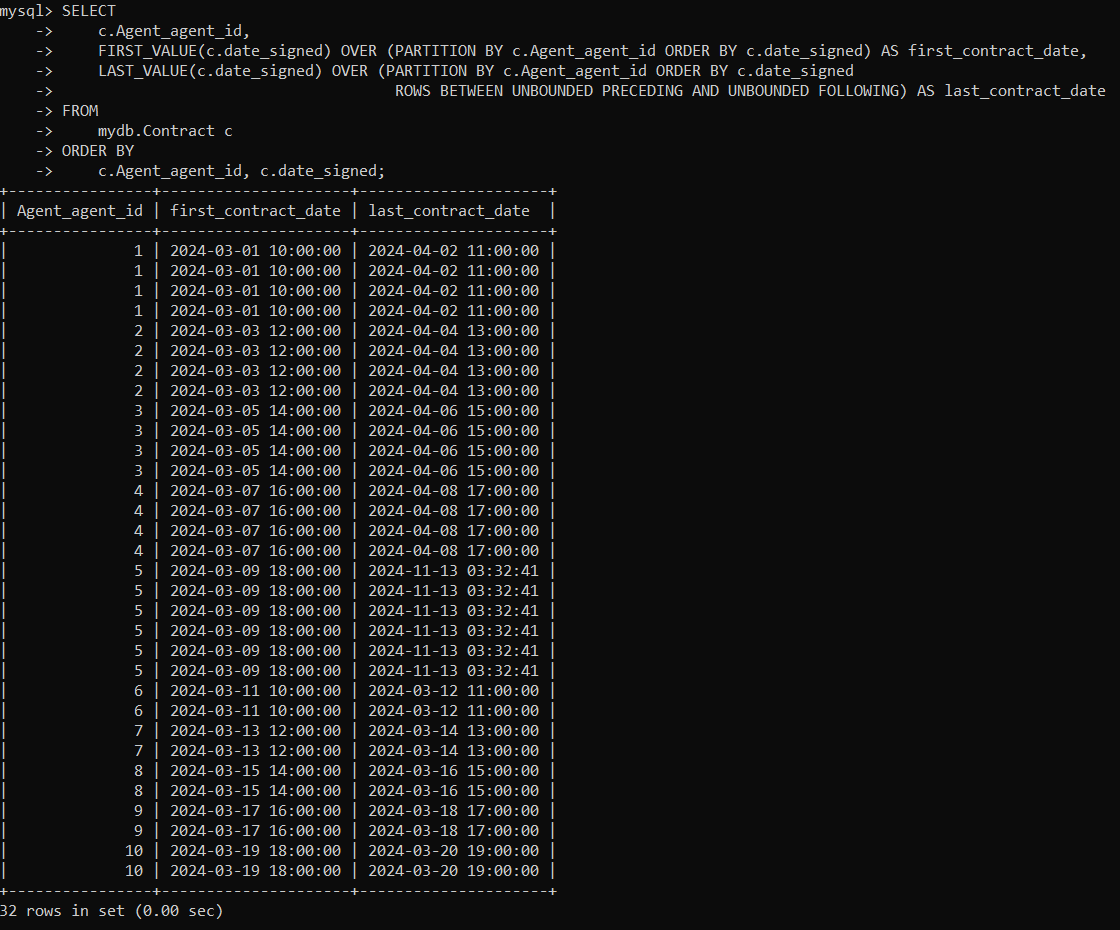


Рисунок 39 – Использование функций смещения

**Вывод**

В ходе практической работы были изучены и применены оконные функции различных типов.