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

**Институт искусственного интеллекта (ИИИ)**

**Кафедра промышленной информатики (ПИ)**

**ИТОГОВЫЙ ОТЧЕТ**

по дисциплине «Разработка баз данных»

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИКБО-11-22, Гришин Андрей Валерьевич* | (подпись) | |
| Старший преподаватель | *Зайцев И.Ю.* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_2024\_г. | |  | |

Москва 2024 г.

**Оглавление**

[**Практическая работа №1** 3](#_Toc185172124)

[**Практическая работа №2** 13](#_Toc185172125)

[**Практическая работа №3** 15](#_Toc185172126)

[**Практическая работа №4** 19](#_Toc185172127)

[**Практическая работа №5** 25](#_Toc185172128)

# **Практическая работа №1**

**Задание:** Создать базу данных и таблицы в ней по выбранной теме, на основе

разработанных моделей

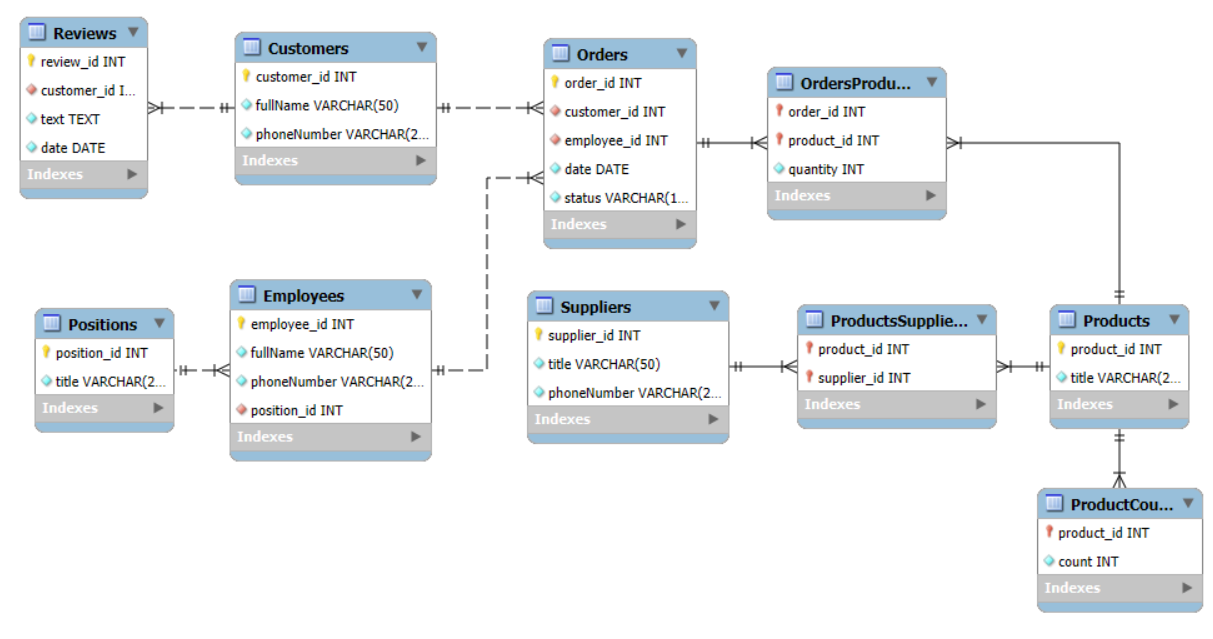


Рисунок 1 – Изначальный план

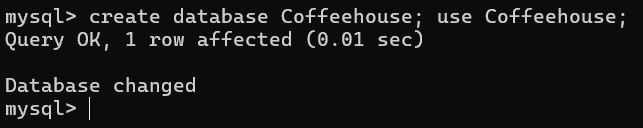


Рисунок 2 – Создание базы данных и переключение в неё

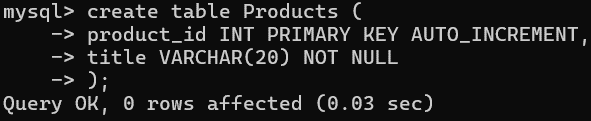


Рисунок 3 – Создание таблицы “Products”

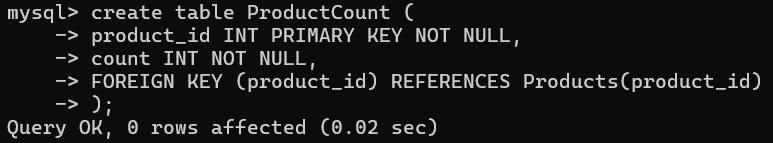


Рисунок 4 – Создание таблицы “ProductCount”

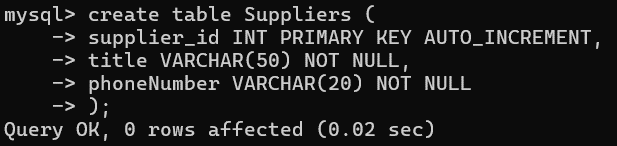


Рисунок 5 – Создание таблицы “Suppliers”

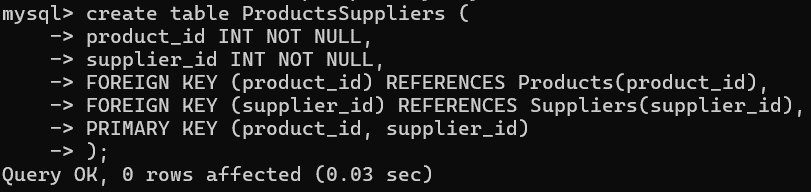


Рисунок 6 – Создание таблицы “ProductsSuppliers”

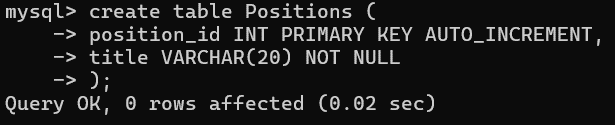


Рисунок 7 – Создание таблицы “Positions”

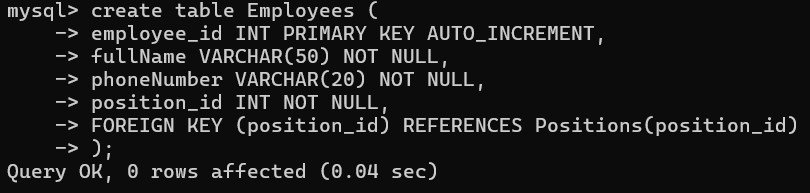


Рисунок 8 – Создание таблицы “Employees”

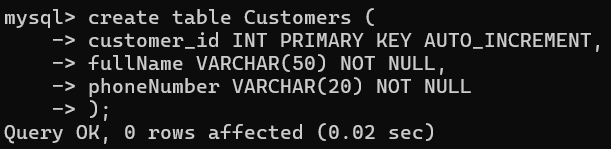
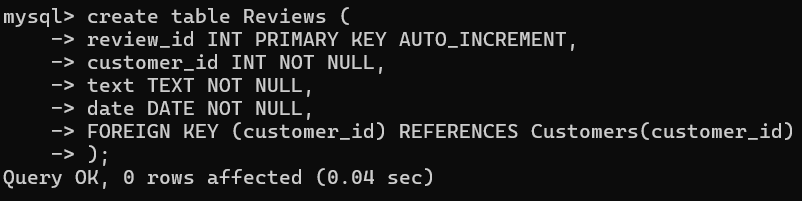


Рисунок 9 – Создание таблицы “Customers”

Рисунок 10 – Создание таблицы “Reviews”

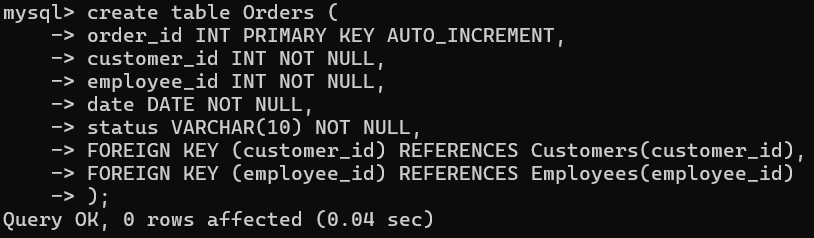


Рисунок 11 – Создание таблицы “Orders”

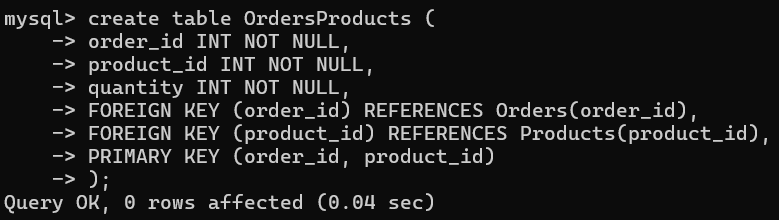


Рисунок 12 – Создание таблицы “OrdersProducts”

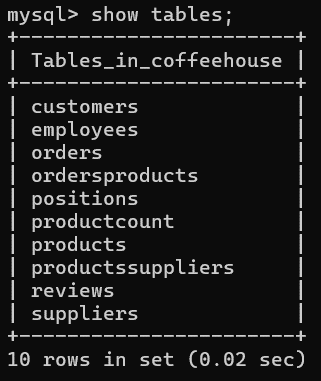


Рисунок 13 – Отображение всех таблиц

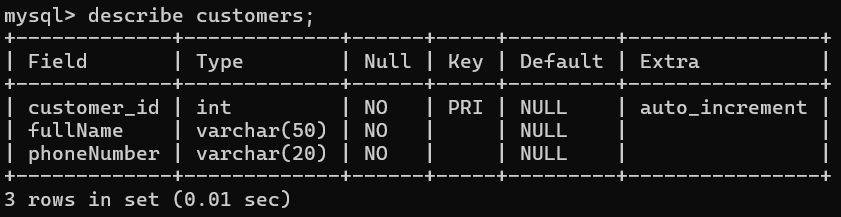


Рисунок 14 – Описание таблицы “Customers”

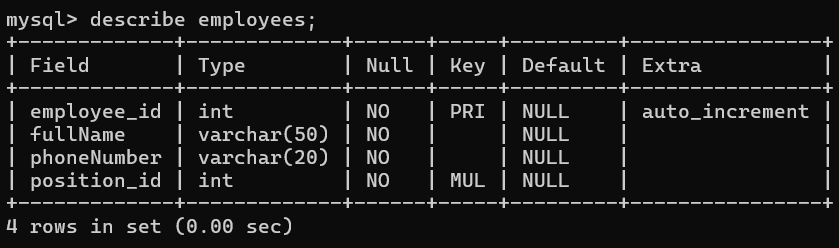


Рисунок 15 – Описание таблицы “Employees”

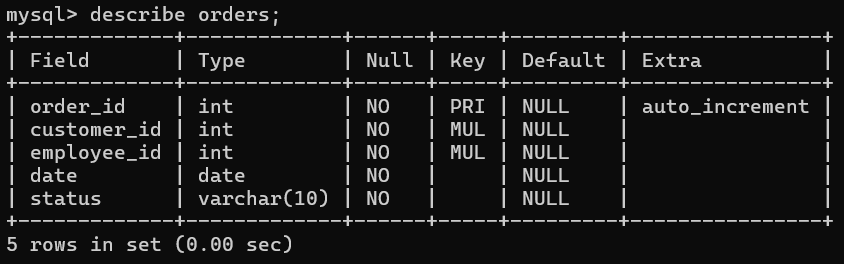


Рисунок 16 – Описание таблицы “Orders”

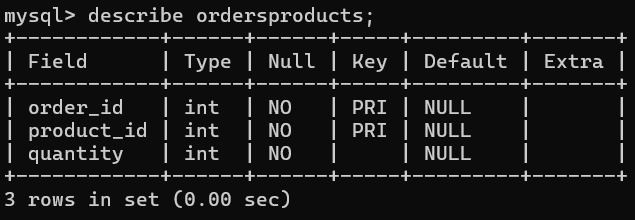


Рисунок 17 – Описание таблицы “OrdersProducts”

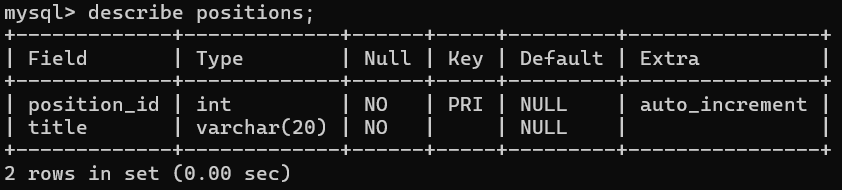


Рисунок 18 – Описание таблицы “Positions”

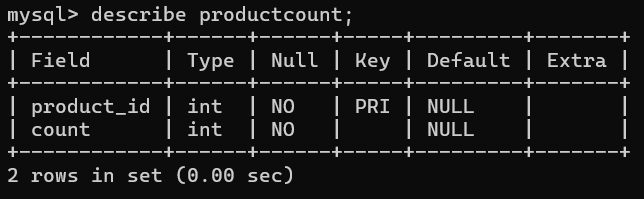


Рисунок 19 – Описание таблицы “ProductCount”

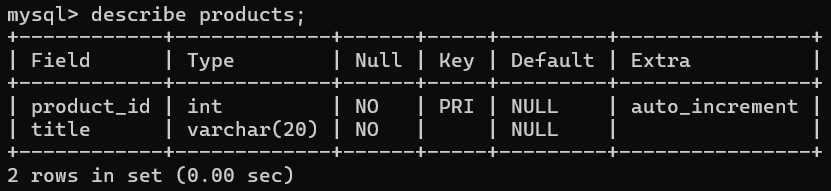


Рисунок 20 – Описание таблицы “Products”

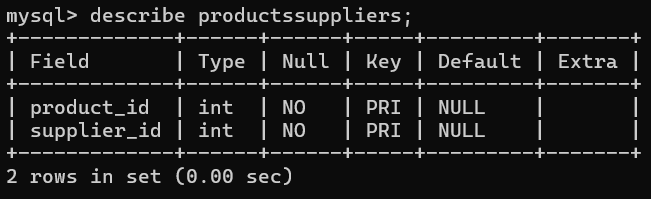


Рисунок 21 – Описание таблицы “ProductsSuppliers”

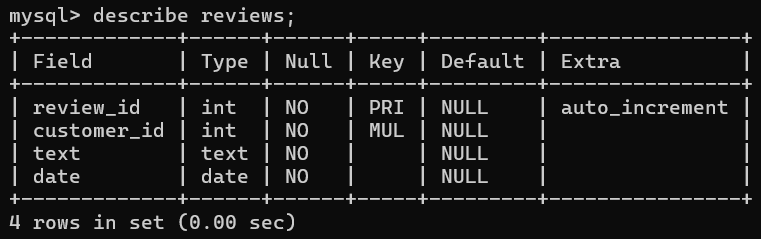


Рисунок 22 – Описание таблицы “Reviews”

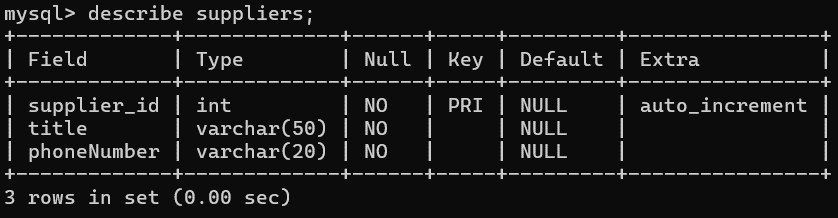
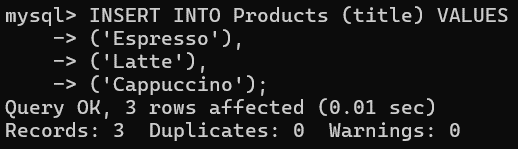


Рисунок 23 – Описание таблицы “Suppliers”



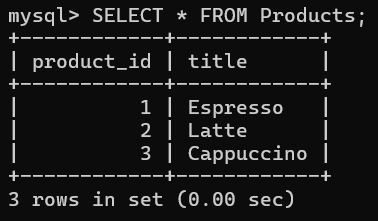
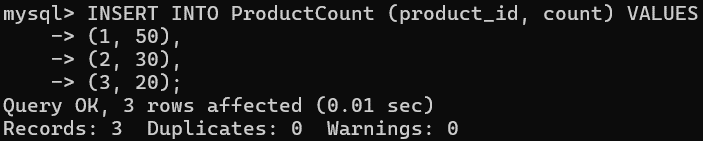


Рисунок 24 – Добавление записей в “Products”



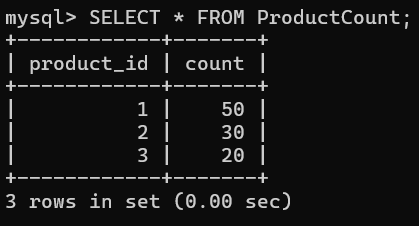
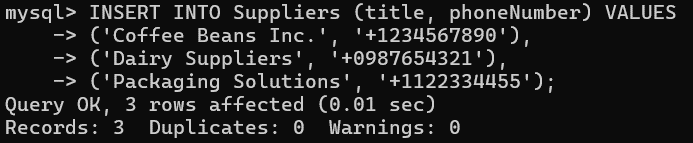


Рисунок 25 – Добавление записей в “ProductCount”



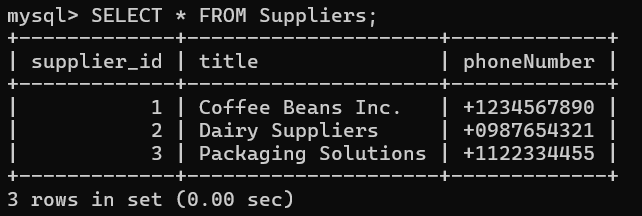
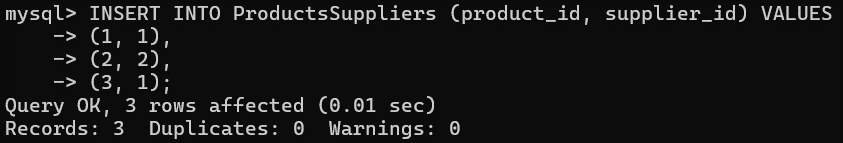


Рисунок 26 – Добавление записей в “Suppliers”



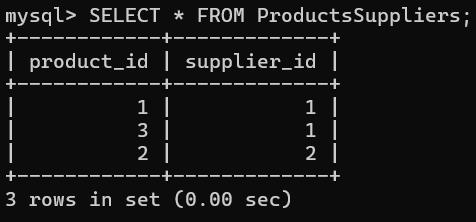
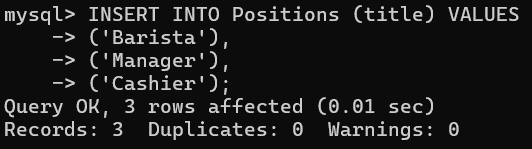


Рисунок 27 – Добавление записей в “ProductsSuppliers”



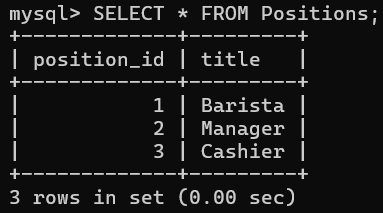
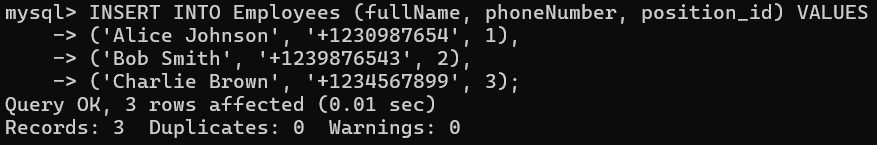


Рисунок 28 – Добавление записей в “Positions”



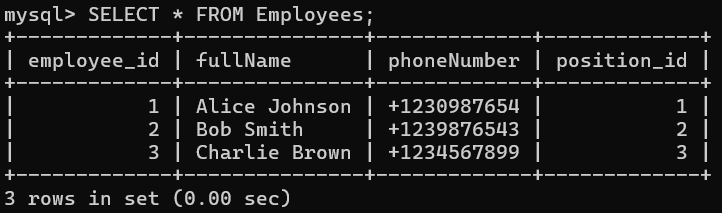
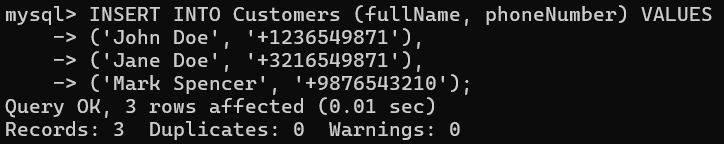


Рисунок 29 – Добавление записей в “Employees”



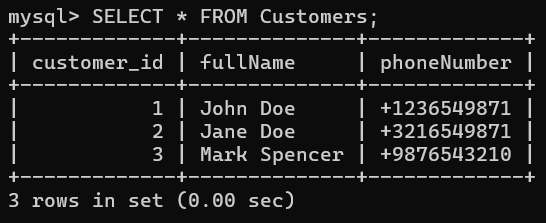
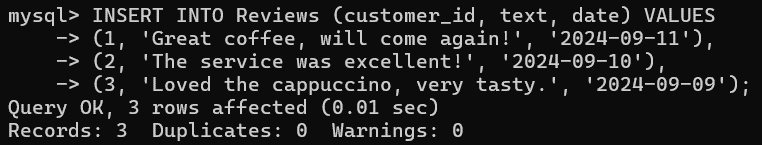


Рисунок 30 – Добавление записей в “Customers”



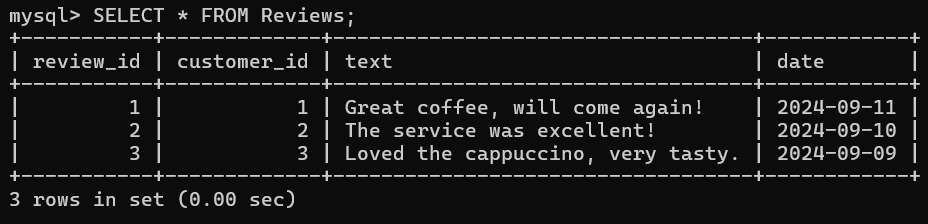
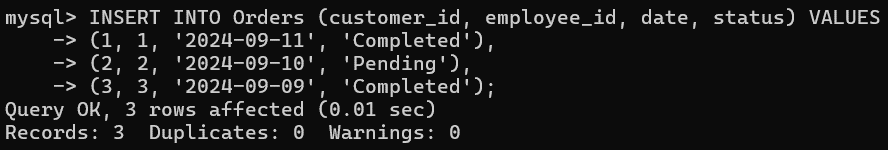


Рисунок 31 – Добавление записей в “Reviews”



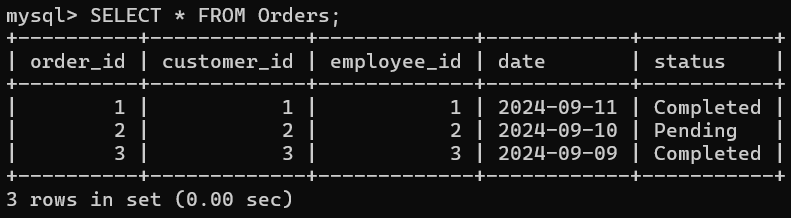


Рисунок 32 – Добавление записей в “Orders”

# **Практическая работа №2**

**Задание:**

Изучить и создать выборку и сортировку данных. Изучить и применить операторы для изменения данных в таблицах.

Результат работы в виде отчета должен содержать:

снимки экрана (скриншоты) выборки данных по различным параметрам (по каждому оператору);

снимок экрана (скриншоты) сортировки данных;

снимки экрана (скриншоты) применения операторов изменения данных в таблицах Вашей базы данных;

добавить все практические работы по SQL в итоговых отчет.

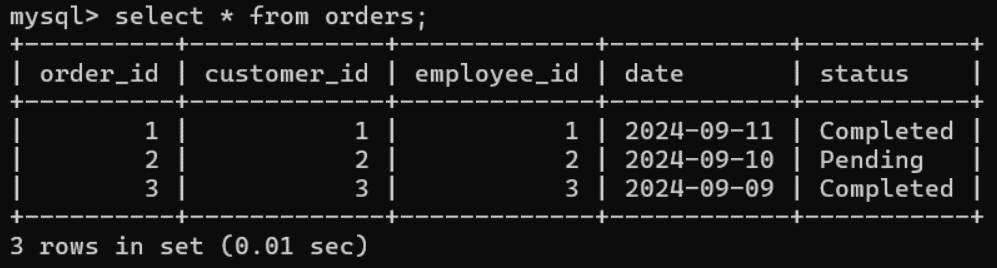


Рисунок 33 – Записи в таблице “Orders”

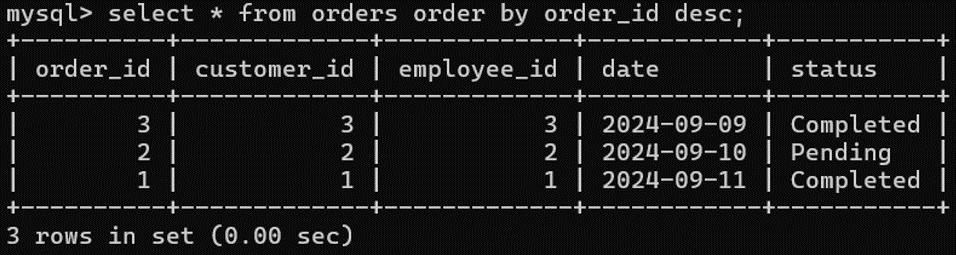


Рисунок 34 – Записи в таблице с сортировкой по столбцу order\_id по убыванию

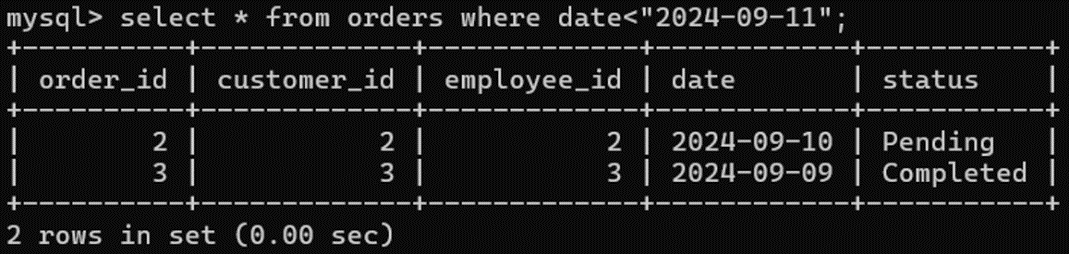


Рисунок 35 – Записи в таблице с датой заказа до 11.09.2024 (не включительно)

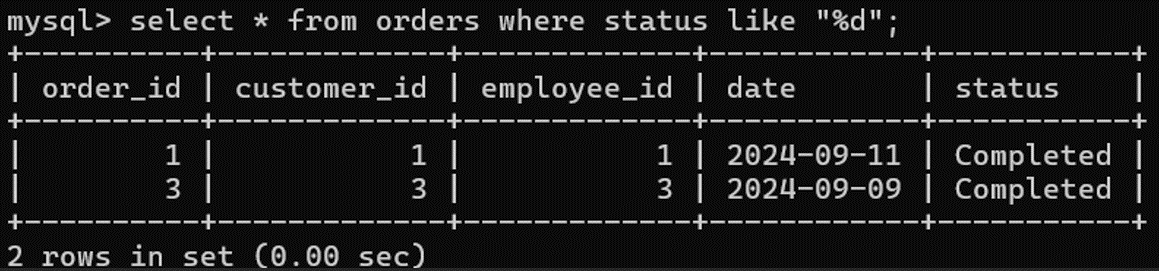


Рисунок 36 – Записи в таблице, в который столбец status оканчивается на “d”

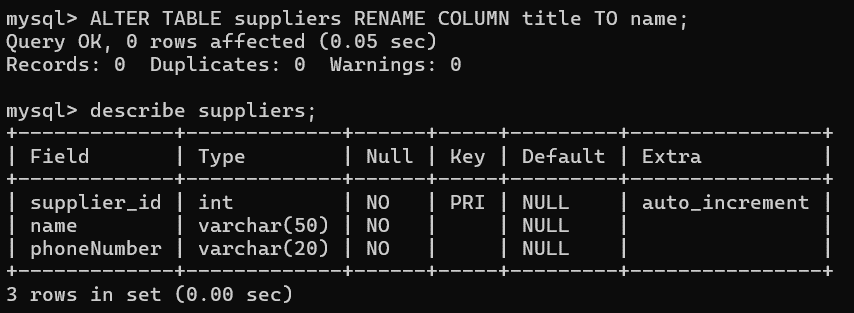


Рисунок 37 – Переименование столбца имени в таблице suppliers

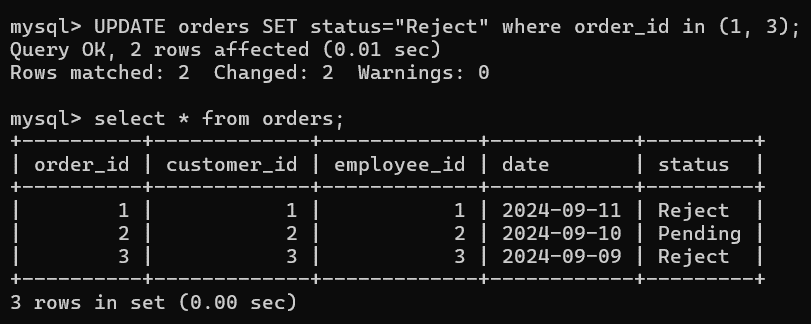


Рисунок 38 – Изменение статуса заказа

# **Практическая работа №3**

**Задание:**

1. Научится формировать модель БД с помощью инструментов СУБД (MySQL Workbench, dbForge Studio, PostgreSQL – по выбору студента) по своей теме.
2. Научится осуществлять перенос своей БД на другой сервер.
3. Изучить команды модификации данных (DML)
4. Осуществить выборку данных по своей теме с помощью различных операторов.
5. Изучить и применить к своей БД хранимые процедуры, функции и триггеры.

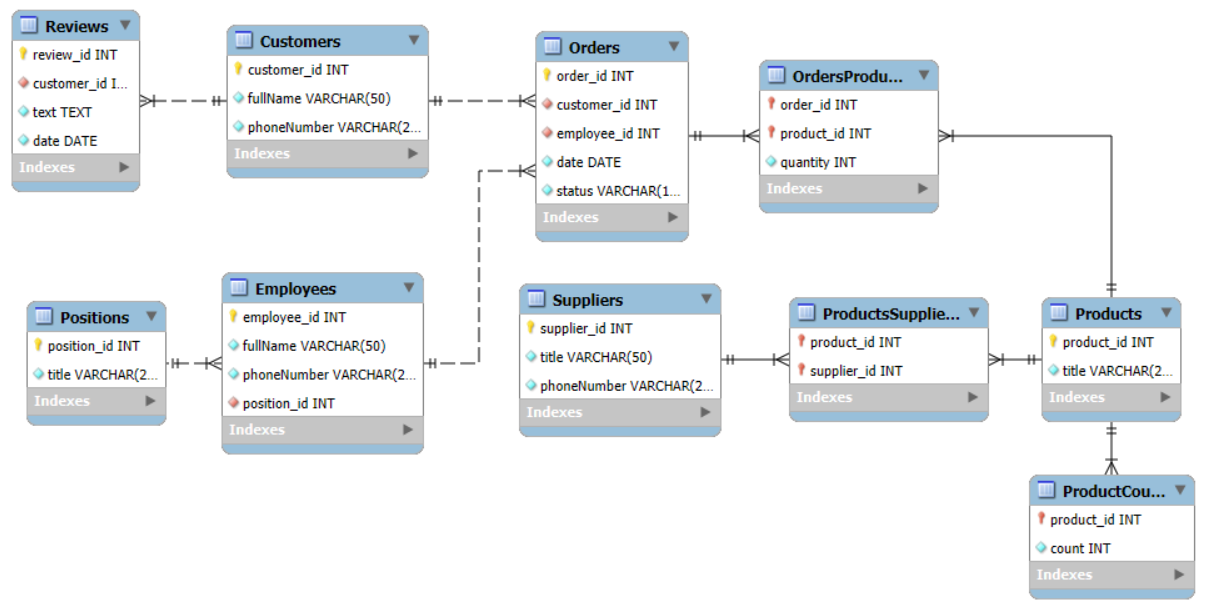


Рисунок 39 – ER-модель

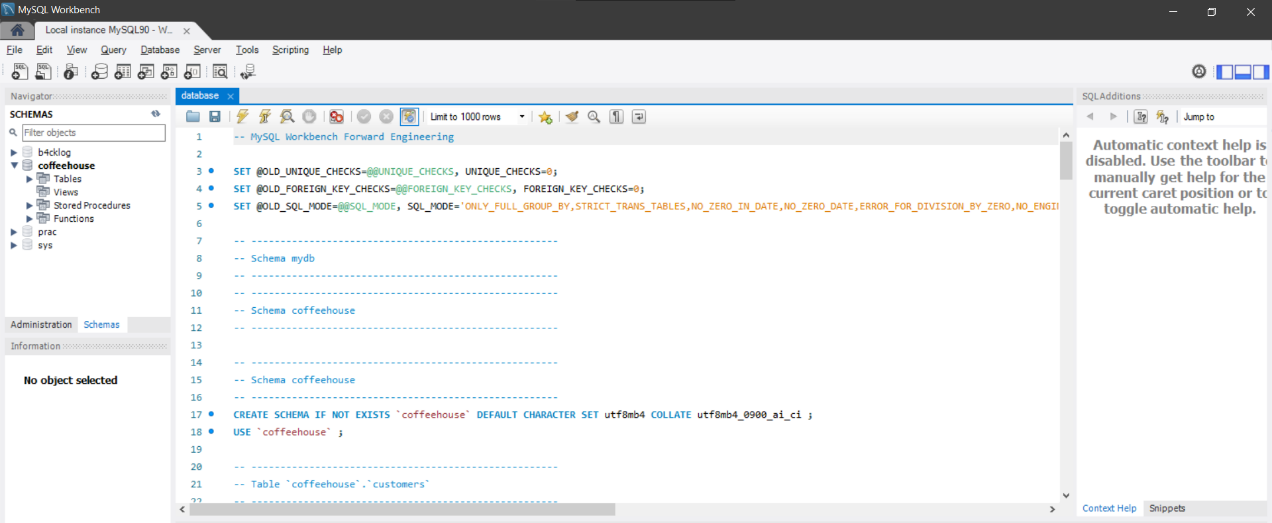


Рисунок 40 – Извлечение SQL-скрипта БД через Forward Engineering для переноса на другой сервер

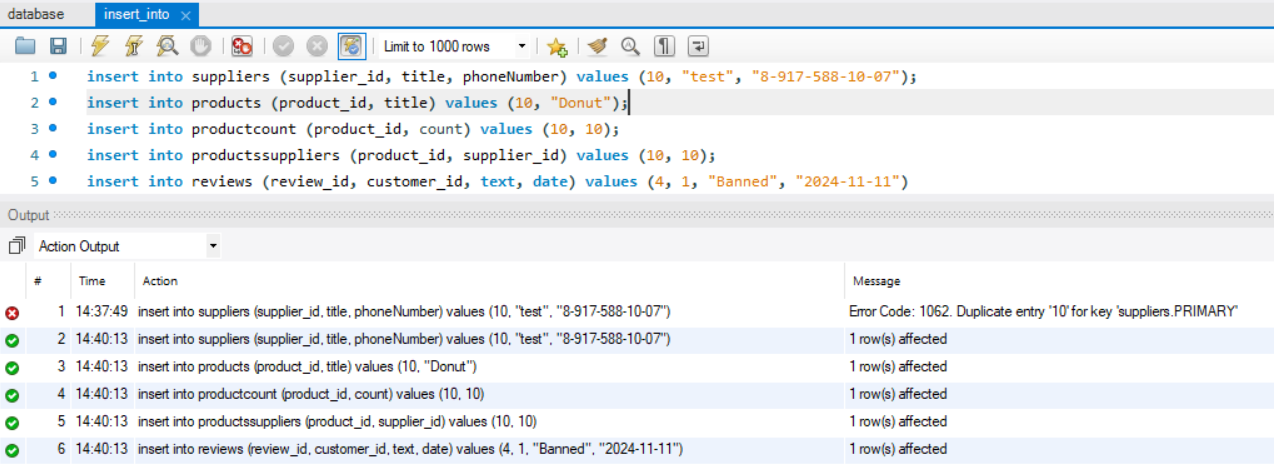


Рисунок 41 – Заполнение БД

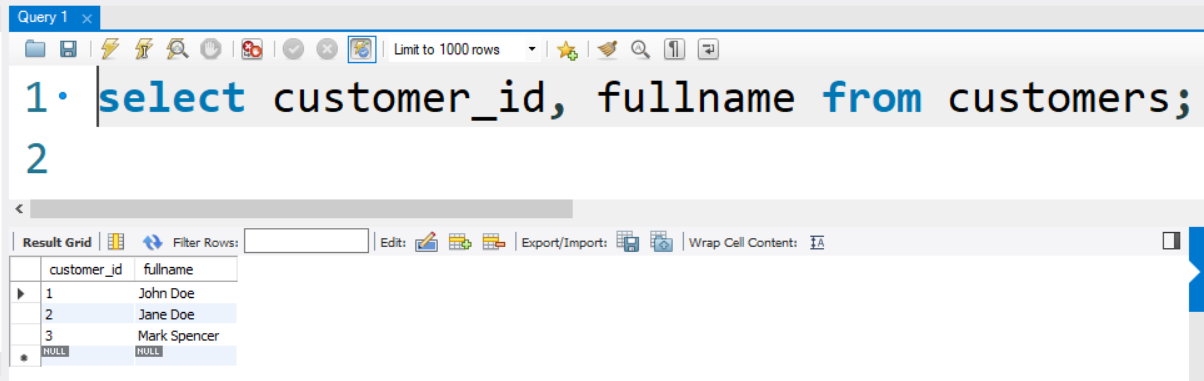


Рисунок 42 – Операция проекции

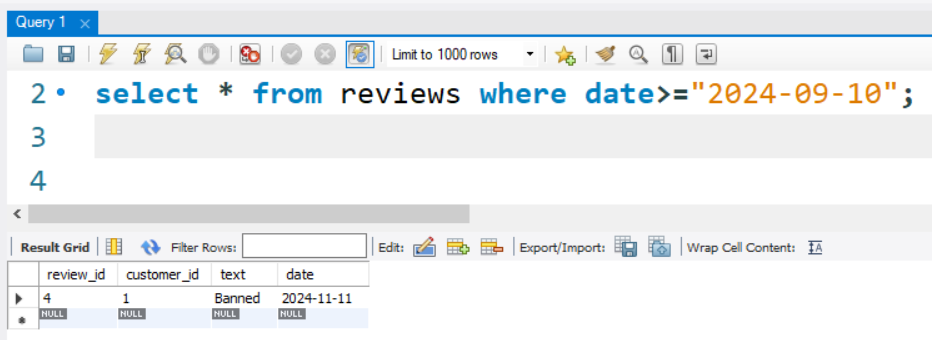


Рисунок 43 – Операция селекции

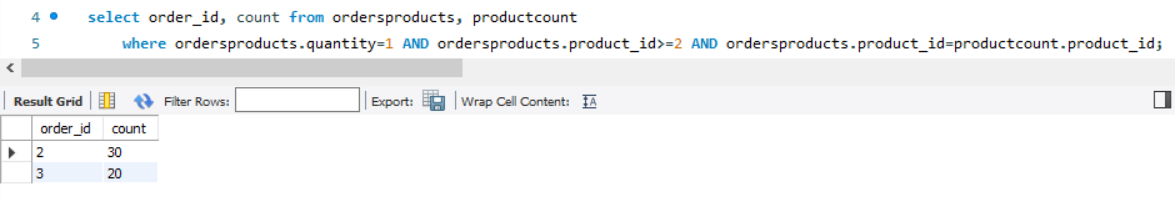


Рисунок 44 – Операция соединения

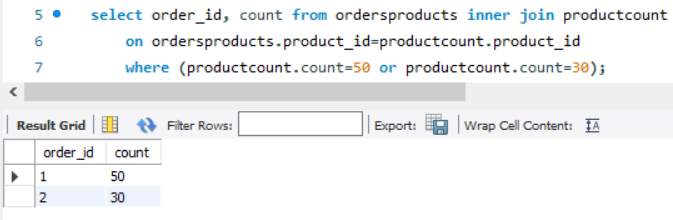


Рисунок 45 – Операция объединения

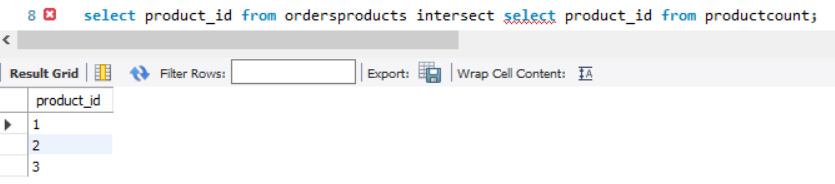


Рисунок 46 – Операция пересечения

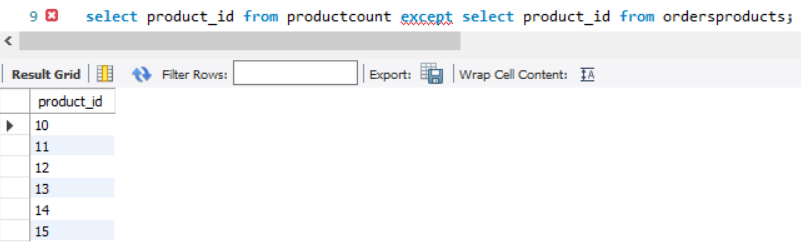


Рисунок 47 – Операция разности

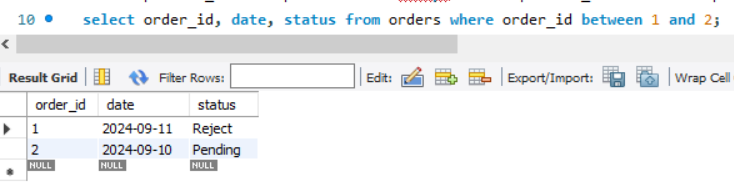


Рисунок 48 – Операция группировки

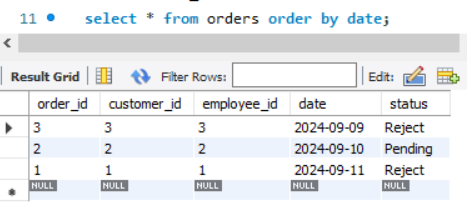


Рисунок 49 – Операция сортировки

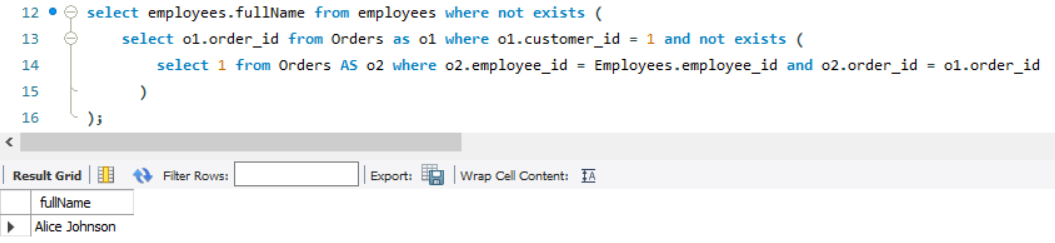


Рисунок 50 – Операция деления

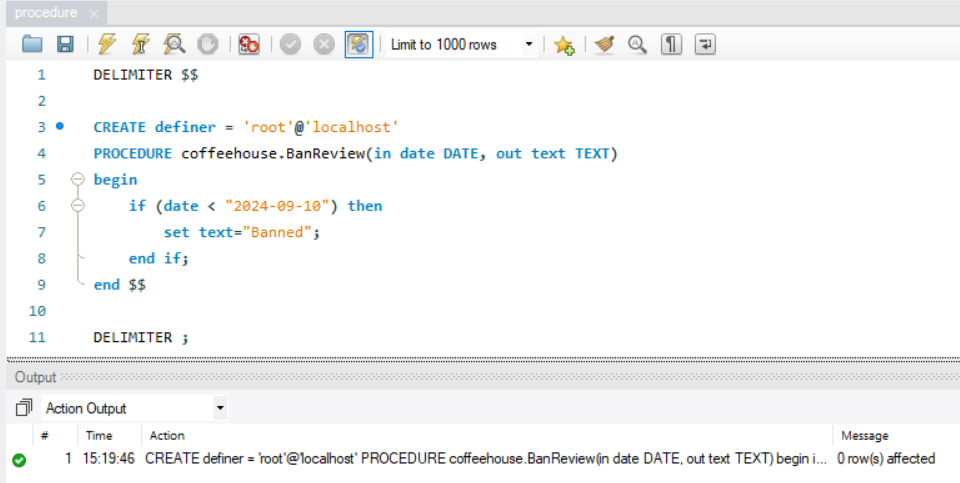


Рисунок 51 – Процедура для обновления текста отзыва

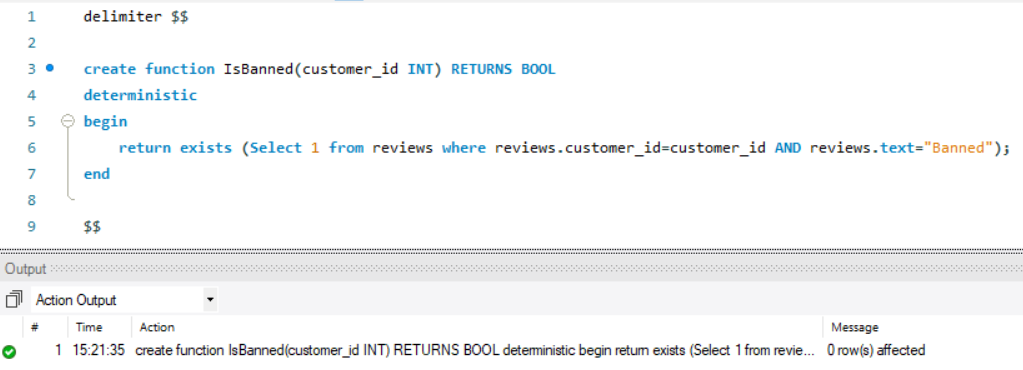


Рисунок 52 – Функция возвращает, в бане ли отзыв клиента

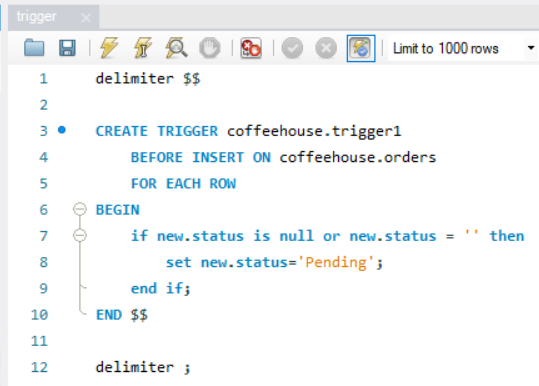


Рисунок 53 – Триггер для обновления статуса заказа при добавлении

# **Практическая работа №4**

**Оконные функции в MySQL** — это специальный тип функции, который позволяет выполнять агрегатные и аналитические операции над группами строк, которые определены внутри отдельного окна (или оконного фрейма). Оконные функции представляют собой удобный инструмент по работы с данными и предоставляют более гибкий способ обращения с ними, чем традиционные агрегатные функции за счёт того, что они могут учитывать порядок сортировки данных и разбивать их на группы без фактической группировки.

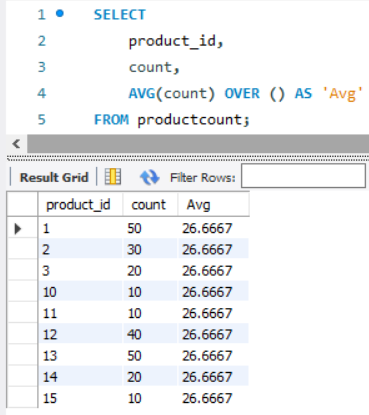


Рисунок 54 – Открытие окна при помощи OVER()

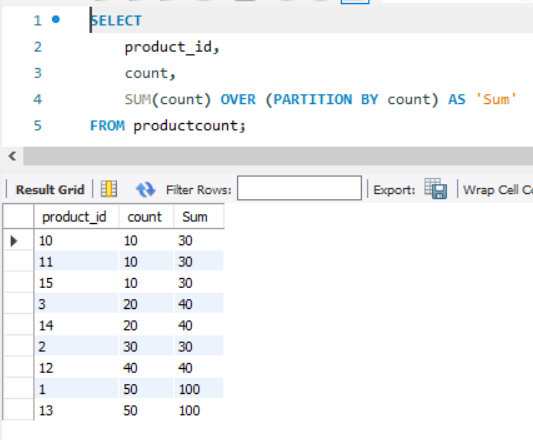


Рисунок 55 – Применение инструкции PARTITION BY()

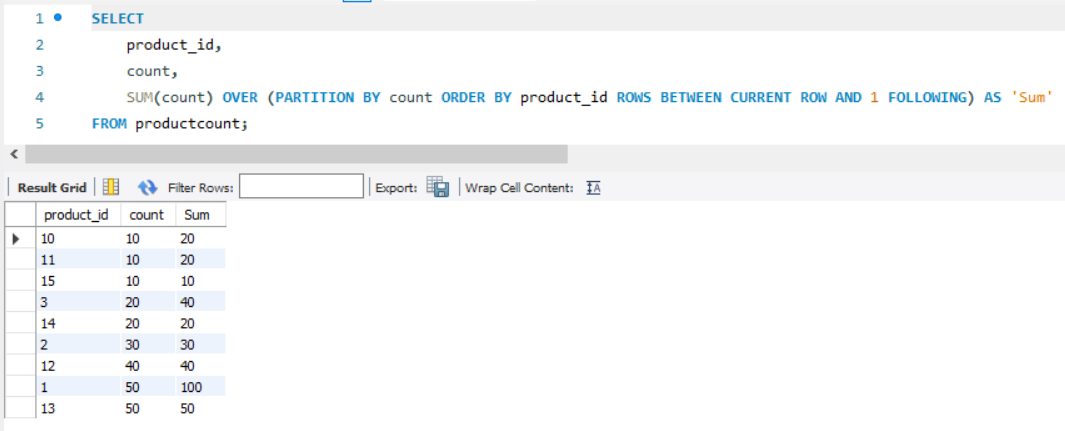


Рисунок 56 – Применение ORDER BY

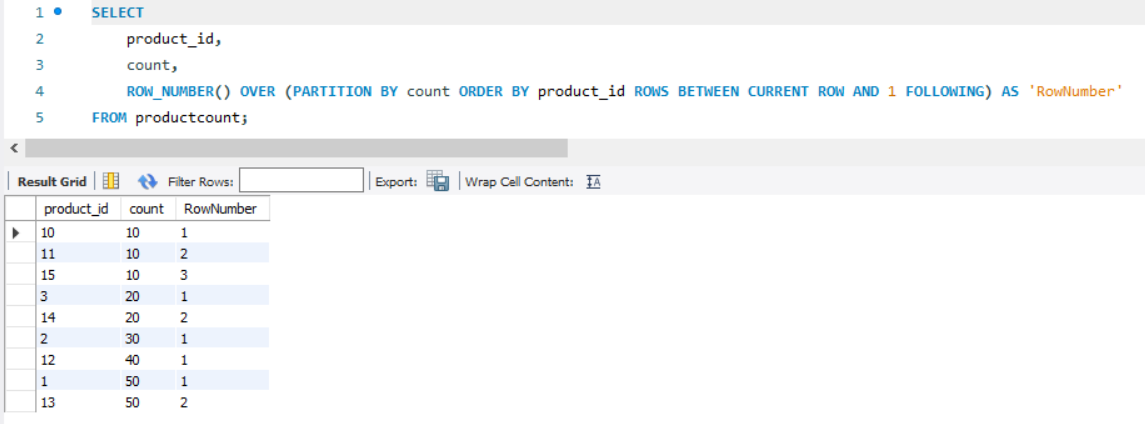


Рисунок 57 – Применение ROWS с инструкциями CURRENT ROW и FOLLOWING

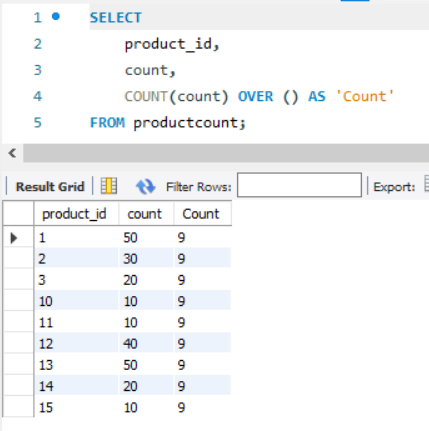


Рисунок 58 – Применение COUNT()

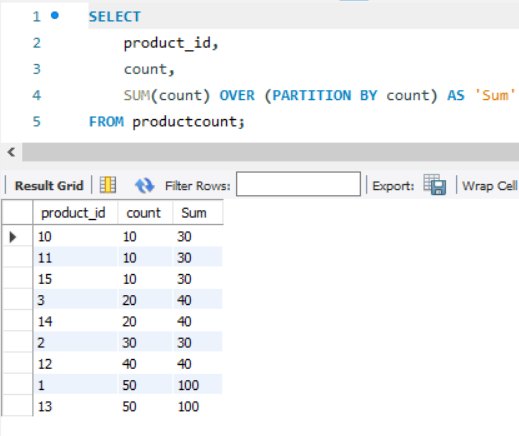


Рисунок 59 – Применение Sum()

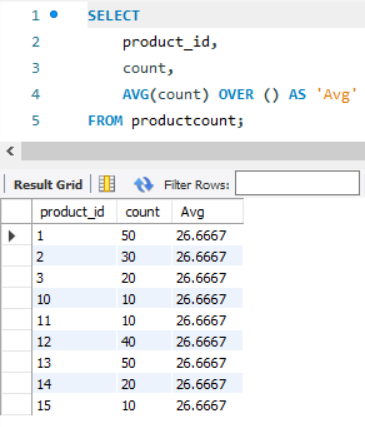


Рисунок 60 – Применение AVG()

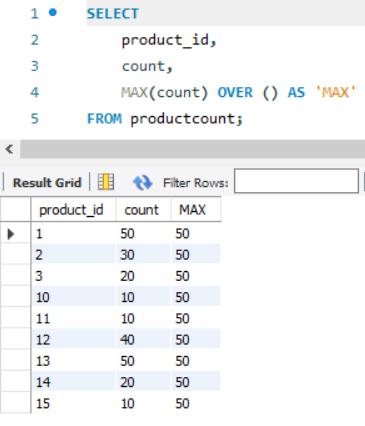
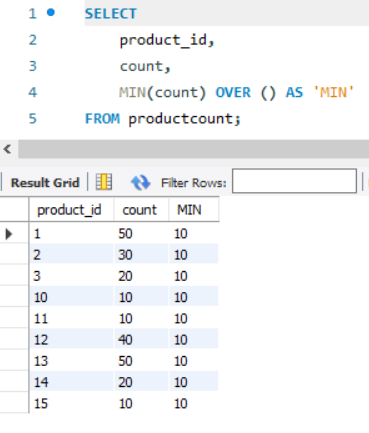


Рисунок 61 – Применение Min и Max

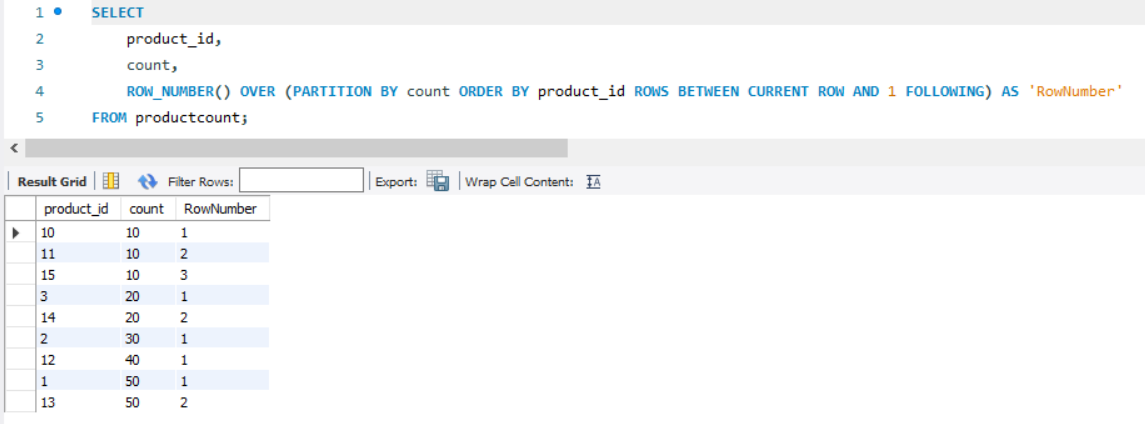


Рисунок 62 –Применение ROW\_NUMBER()

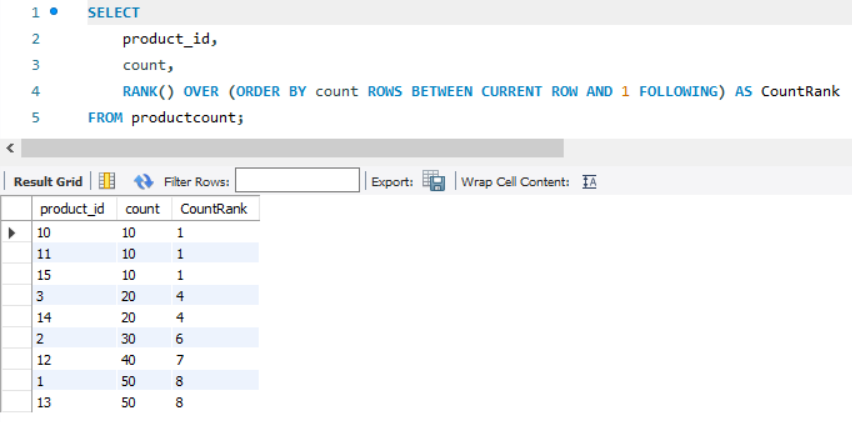


Рисунок 63 – Применение RANK()

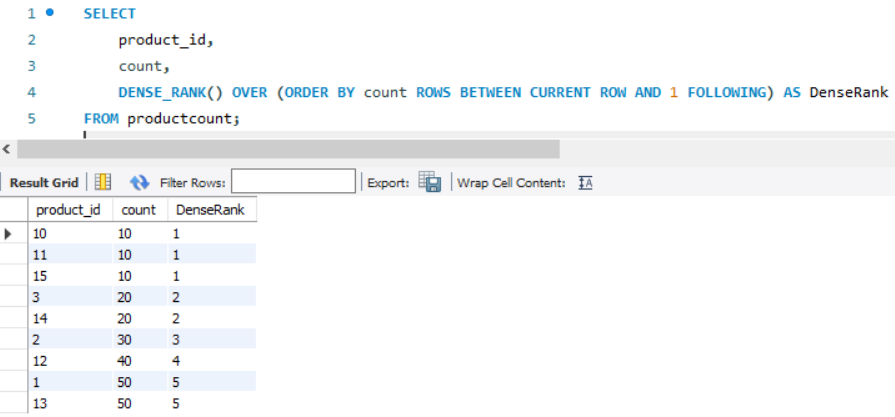


Рисунок 64 – Применение DENSE\_RANK()

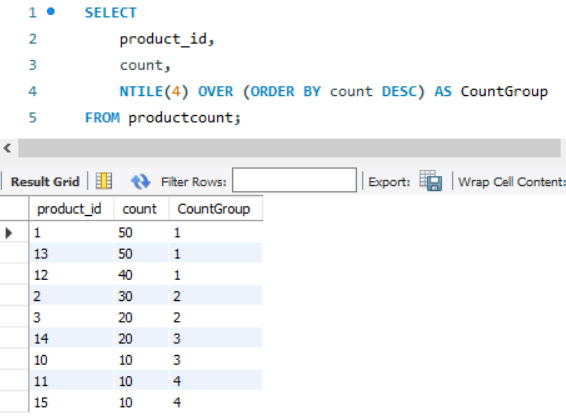


Рисунок 65 – Применение NTILE()

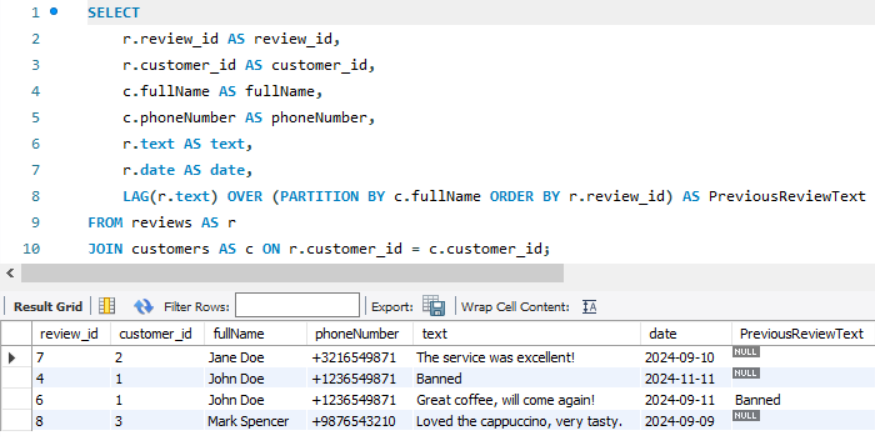


Рисунок 66 – Применение LAG()

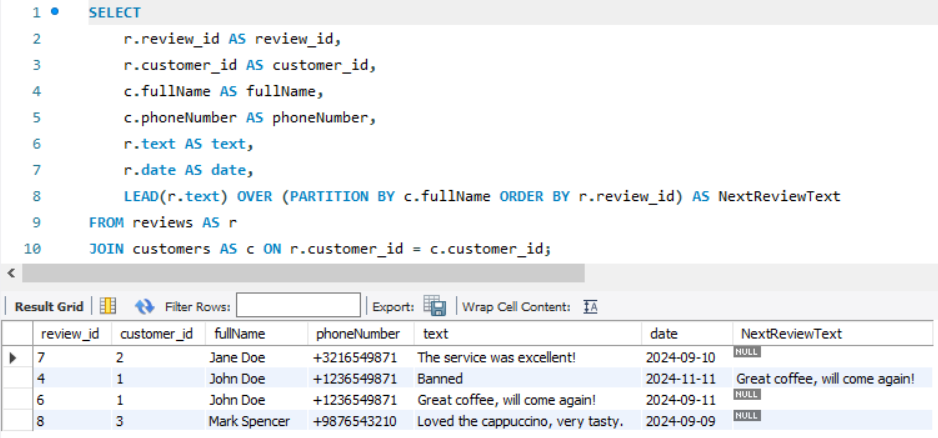


Рисунок 67 – Применение LEAD()

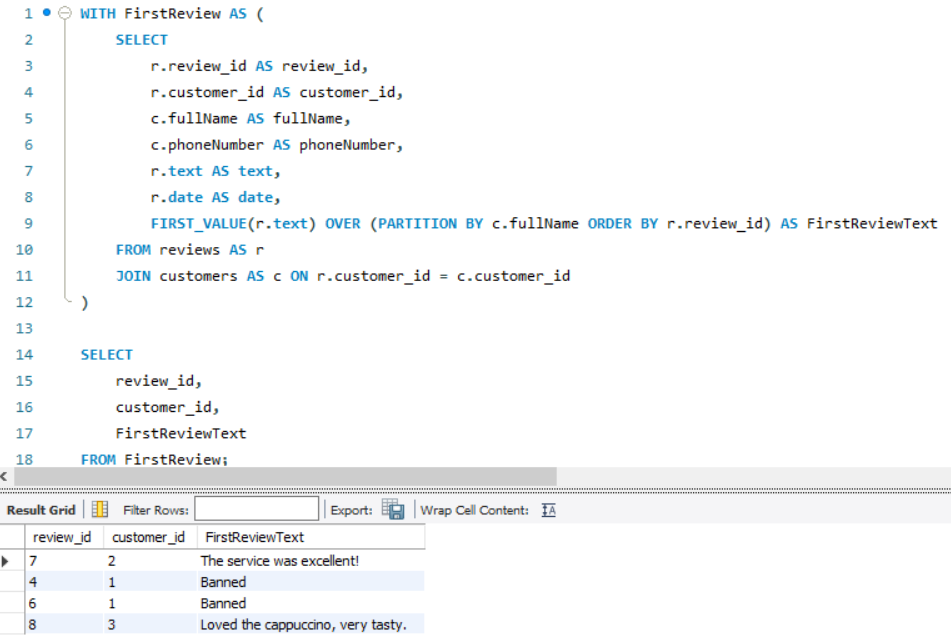


Рисунок 68 – Применение FIRST\_VALUE()

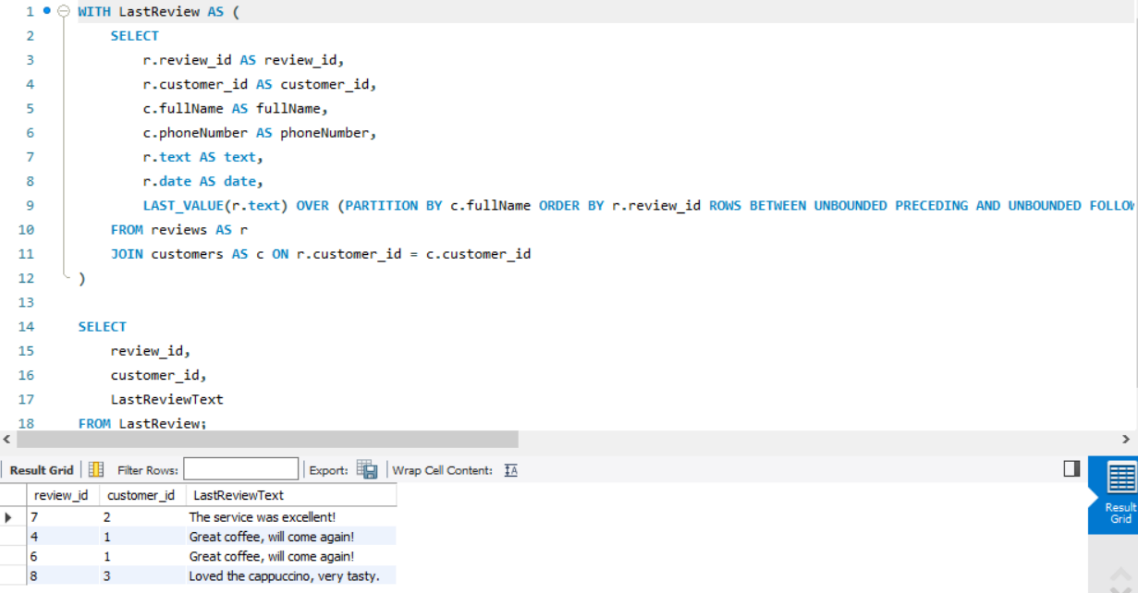


Рисунок 69 – Применение LAST\_VALUE()

# **Практическая работа №5**

**Задание:** Реализовать интерфейс для работы с БД

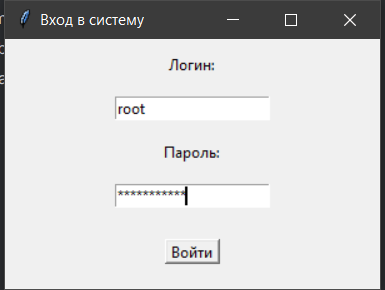


Рисунок 70 – Окно авторизации

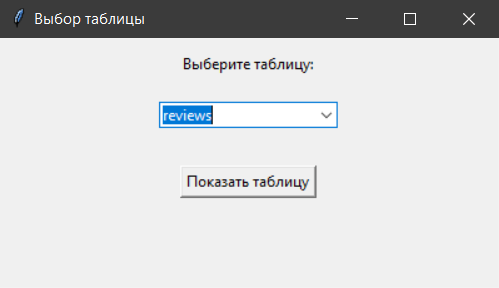


Рисунок 71 – Окно выбора таблицы

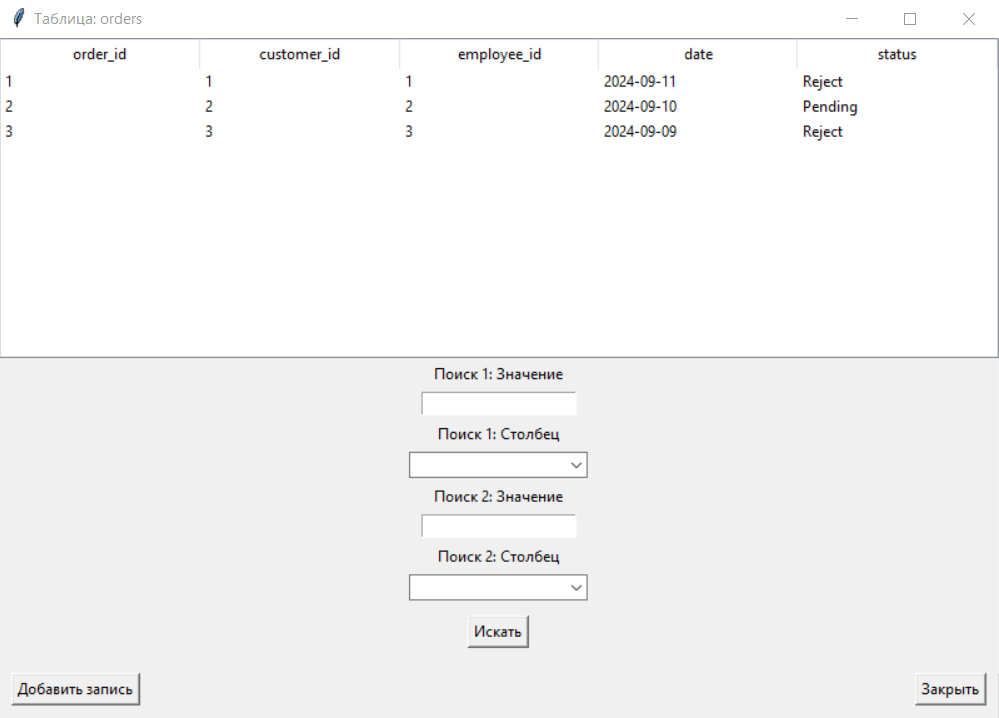


Рисунок 72 – Окно выбранной таблицы (от имени администратора)

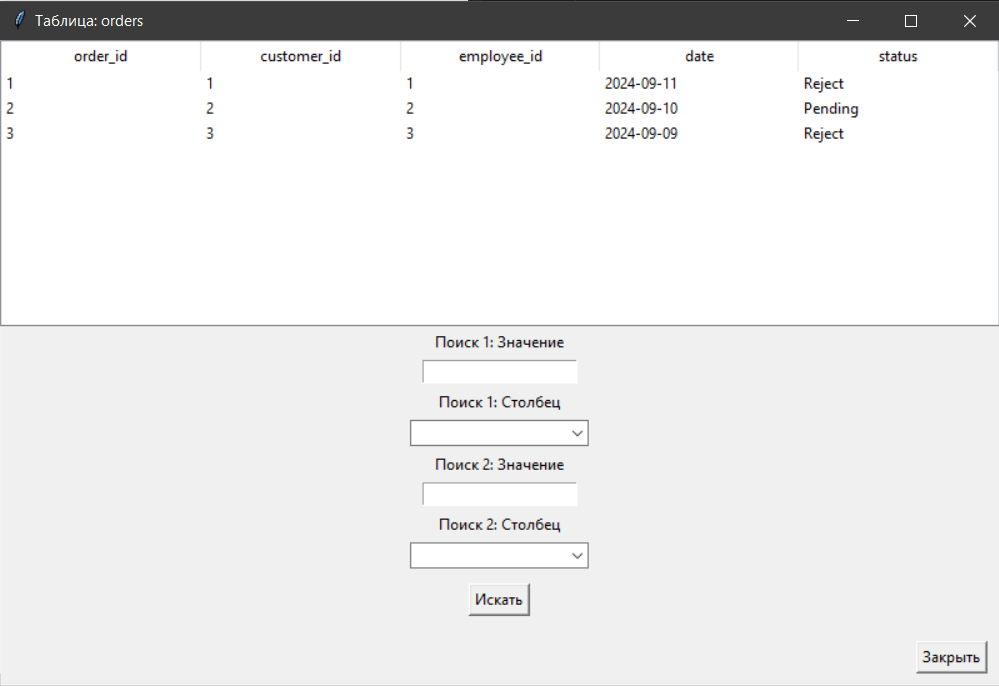


Рисунок 73 – Окно выбранной таблицы (от имени пользователя)

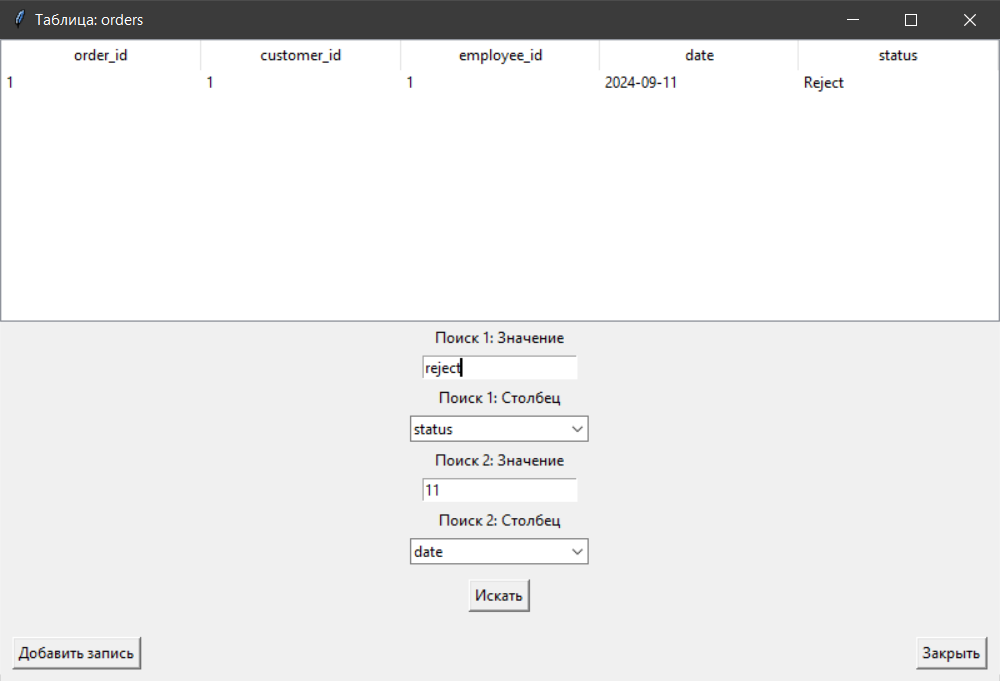
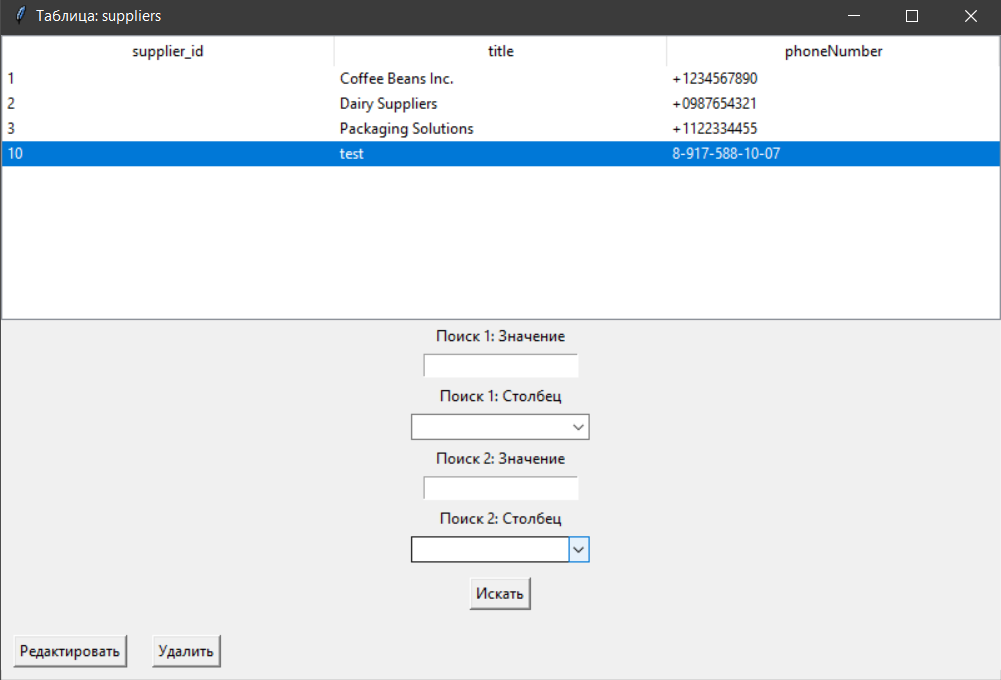
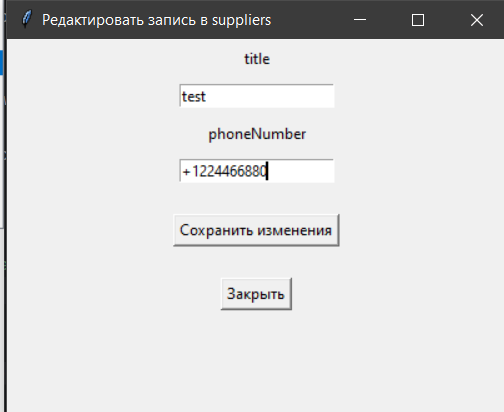
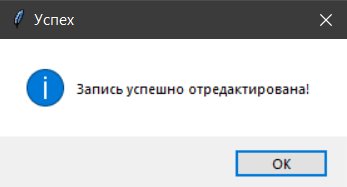


Рисунок 74 – Поиск записи в таблице по двум параметрам







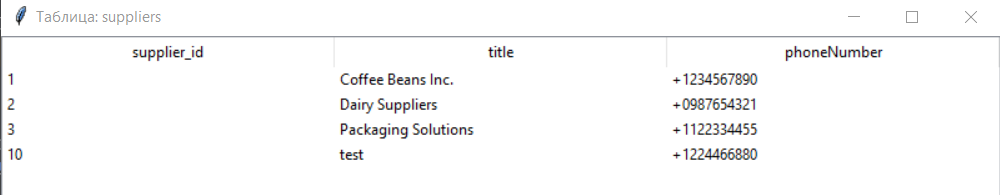
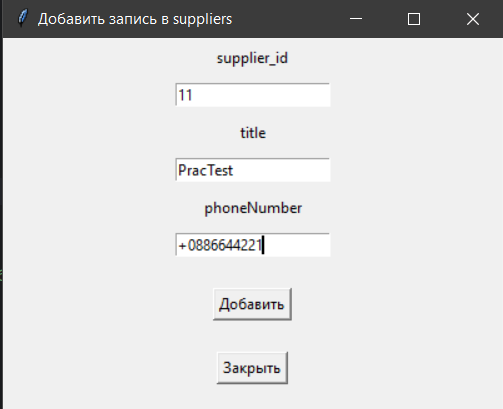
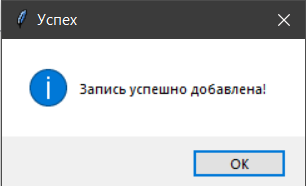


Рисунок 75 – Процесс редактирования записи







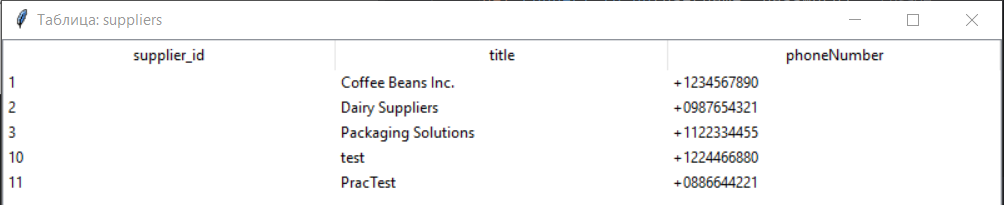
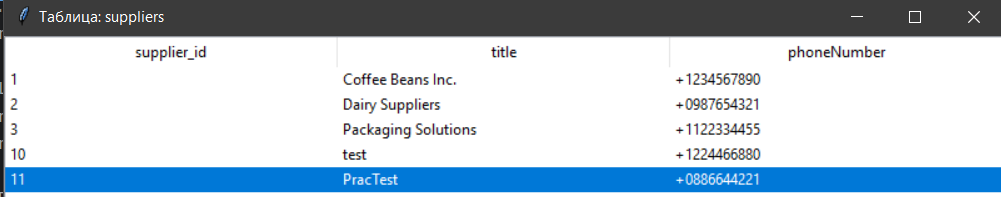
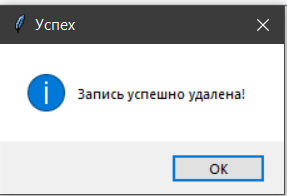


Рисунок 76 – Процесс добавления записи





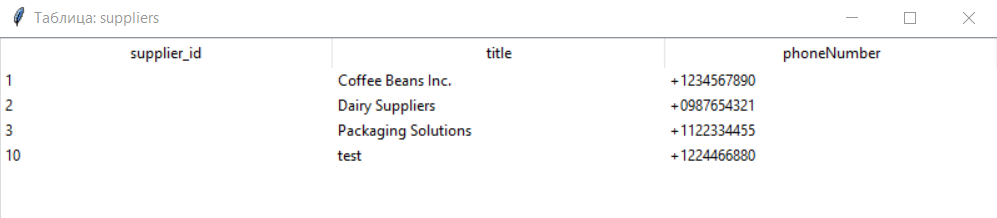


Рисунок 77 – Процесс удаления записи



Рисунок 78 – Триггер для проверки доступности продуктов перед добавлением заказа

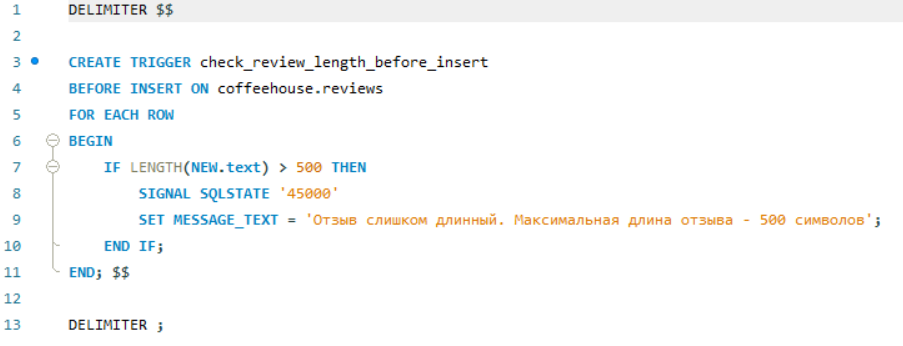


Рисунок 79 – Триггер для проверки длины отзыва

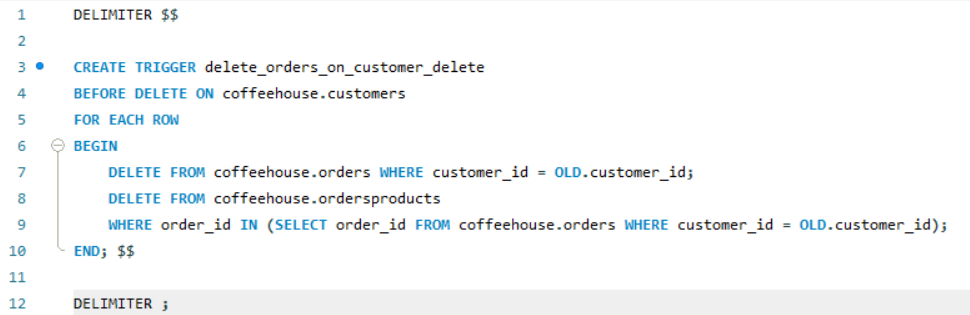


Рисунок 80 – Триггер для автоматического удаления заказов при удалении клиента

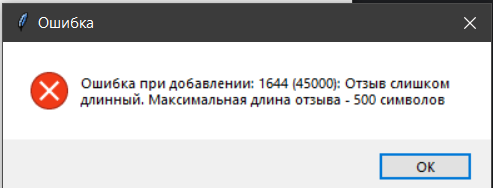


Рисунок 81 – Пример работы триггера

*Листинг 1 – код интерфейса*

|  |
| --- |
| import tkinter as tk  from tkinter import ttk, messagebox  import mysql.connector  def connect\_to\_db(username, password):  try:  connection = mysql.connector.connect(  host="localhost",  user=username,  password=password,  database="coffeehouse"  )  return connection  except mysql.connector.Error as e:  messagebox.showerror("Ошибка", f"Ошибка подключения: {e}")  return None  def check\_user\_role(connection, username):  try:  cursor = connection.cursor()  cursor.execute(f"SHOW GRANTS FOR '{username}'@'localhost'")  grants = cursor.fetchall()  is\_admin = any(  "ALL PRIVILEGES" in grant[0] or "GRANT OPTION" in grant[0] for grant in grants  )  return is\_admin  except mysql.connector.Error as e:  messagebox.showerror("Ошибка", f"Ошибка при проверке прав: {e}")  return False  def fetch\_table\_data(connection, table\_name):  try:  cursor = connection.cursor()  cursor.execute(f"SELECT \* FROM {table\_name}")  columns = [col[0] for col in cursor.description]  rows = cursor.fetchall()  cursor.close()  return columns, rows  except mysql.connector.Error as e:  messagebox.showerror("Ошибка", f"Ошибка запроса: {e}")  return [], []  def add\_row\_to\_table(connection, table\_name, columns, table\_window, user\_role):  def save\_row():  values = [entry.get() for entry in entries]  placeholders = ', '.join(['%s'] \* len(values))  query = f"INSERT INTO {table\_name} ({', '.join(columns)}) VALUES ({placeholders})"  try:  cursor = connection.cursor()  cursor.execute(query, tuple(values))  connection.commit()  messagebox.showinfo("Успех", "Запись успешно добавлена!")  table\_window.destroy()  display\_table(connection, table\_name, user\_role)  except mysql.connector.Error as e:  messagebox.showerror("Ошибка", f"Ошибка при добавлении: {e}")  add\_window = tk.Toplevel()  add\_window.title(f"Добавить запись в {table\_name}")  add\_window.geometry("400x300")  entries = []  for col in columns:  tk.Label(add\_window, text=col).pack(pady=5)  entry = tk.Entry(add\_window)  entry.pack(pady=5)  entries.append(entry)  tk.Button(add\_window, text="Добавить", command=save\_row).pack(pady=20)  tk.Button(add\_window, text="Закрыть", command=add\_window.destroy).pack(pady=5)  def edit\_row\_in\_table(connection, table\_name, columns, row\_id, table\_window, user\_role):  def save\_changes():  values = [entry.get() for entry in entries]  update\_query = f"UPDATE {table\_name} SET {', '.join([f'{col}=%s' for col in columns[1:]])} WHERE {columns[0]}=%s"  try:  cursor = connection.cursor()  cursor.execute(update\_query, tuple(values) + (row\_id,))  connection.commit()  messagebox.showinfo("Успех", "Запись успешно отредактирована!")  table\_window.destroy()  display\_table(connection, table\_name, user\_role)  except mysql.connector.Error as e:  messagebox.showerror("Ошибка", f"Ошибка при редактировании: {e}")  edit\_window = tk.Toplevel()  edit\_window.title(f"Редактировать запись в {table\_name}")  edit\_window.geometry("400x300")  cursor = connection.cursor()  cursor.execute(f"SELECT \* FROM {table\_name} WHERE {columns[0]} = %s", (row\_id,))  row\_data = cursor.fetchone()  cursor.close()  if not row\_data:  messagebox.showerror("Ошибка", "Запись не найдена!")  edit\_window.destroy()  return  entries = []  for idx, col in enumerate(columns[1:]):  tk.Label(edit\_window, text=col).pack(pady=5)  entry = tk.Entry(edit\_window)  entry.insert(0, row\_data[idx + 1])  entry.pack(pady=5)  entries.append(entry)  tk.Button(edit\_window, text="Сохранить изменения", command=save\_changes).pack(pady=20)  tk.Button(edit\_window, text="Закрыть", command=edit\_window.destroy).pack(pady=5)  def delete\_row\_from\_table(connection, table\_name, columns, row\_id, table\_window, user\_role):  try:  cursor = connection.cursor()  cursor.execute(f"DELETE FROM {table\_name} WHERE {columns[0]}=%s", (row\_id,))  connection.commit()  messagebox.showinfo("Успех", "Запись успешно удалена!")  table\_window.destroy()  display\_table(connection, table\_name, user\_role)  except mysql.connector.Error as e:  messagebox.showerror("Ошибка", f"Ошибка при удалении: {e}")  def search\_table(connection, table\_name, search\_term1, column1, search\_term2, column2):  cursor = connection.cursor()  # Формируем условие для поиска  conditions = []  params = []  if search\_term1 and column1:  conditions.append(f"`{column1}` LIKE %s")  params.append(f"%{search\_term1}%")  if search\_term2 and column2:  conditions.append(f"`{column2}` LIKE %s")  params.append(f"%{search\_term2}%")  if not conditions:  return [] # Если нет условий, возвращаем пустой результат  query = f"SELECT \* FROM {table\_name} WHERE {' AND '.join(conditions)}"  cursor.execute(query, tuple(params))  rows = cursor.fetchall()  return rows  def display\_table(connection, table\_name, user\_role):  columns, rows = fetch\_table\_data(connection, table\_name)  if not columns:  return  table\_window = tk.Toplevel()  table\_window.title(f"Таблица: {table\_name}")  table\_window.geometry("800x500")  tree = ttk.Treeview(table\_window, columns=columns, show="headings", selectmode="browse")  tree.pack(fill="both", expand=True)  for col in columns:  tree.heading(col, text=col)  tree.column(col, width=100)  for row in rows:  tree.insert("", "end", values=row)  tk.Label(table\_window, text="Поиск 1: Значение").pack(pady=2)  search\_entry1 = tk.Entry(table\_window)  search\_entry1.pack(pady=2)  tk.Label(table\_window, text="Поиск 1: Столбец").pack(pady=2)  search\_column1 = ttk.Combobox(table\_window, values=columns)  search\_column1.pack(pady=2)  tk.Label(table\_window, text="Поиск 2: Значение").pack(pady=2)  search\_entry2 = tk.Entry(table\_window)  search\_entry2.pack(pady=2)  tk.Label(table\_window, text="Поиск 2: Столбец").pack(pady=2)  search\_column2 = ttk.Combobox(table\_window, values=columns)  search\_column2.pack(pady=2)  def search\_action():  search\_term1 = search\_entry1.get()  column1 = search\_column1.get()  search\_term2 = search\_entry2.get()  column2 = search\_column2.get()  search\_results = search\_table(connection, table\_name, search\_term1, column1, search\_term2, column2)  for row in tree.get\_children():  tree.delete(row)  for row in search\_results:  tree.insert("", "end", values=row)  tk.Button(table\_window, text="Искать", command=search\_action).pack(pady=10)  def on\_select(event):  selected\_item = tree.selection()  if selected\_item:  row\_id = tree.item(selected\_item)["values"][0]  show\_edit\_delete\_buttons(row\_id)  tree.bind("<<TreeviewSelect>>", on\_select)  button\_frame = tk.Frame(table\_window)  button\_frame.pack(side="bottom", fill="x", pady=10)  if user\_role:  tk.Button(button\_frame, text="Добавить запись", command=lambda: add\_row\_to\_table(connection, table\_name, columns, table\_window, user\_role)).pack(side="left", padx=10)  def show\_edit\_delete\_buttons(row\_id):  for widget in button\_frame.winfo\_children():  widget.destroy()  if row\_id:  tk.Button(button\_frame, text="Редактировать", command=lambda: edit\_row\_in\_table(connection, table\_name, columns, row\_id, table\_window, user\_role)).pack(side="left", padx=10)  tk.Button(button\_frame, text="Удалить", command=lambda: delete\_row\_from\_table(connection, table\_name, columns, row\_id, table\_window, user\_role)).pack(side="left", padx=10)  tk.Button(button\_frame, text="Закрыть", command=table\_window.destroy).pack(side="right", padx=10)  def main\_window(connection, username):  user\_role = check\_user\_role(connection, username)  def show\_table():  table\_name = table\_selector.get()  if table\_name:  display\_table(connection, table\_name, user\_role)  cursor = connection.cursor()  cursor.execute("SHOW TABLES")  tables = [table[0] for table in cursor.fetchall()]  cursor.close()  root = tk.Tk()  root.title("Выбор таблицы")  root.geometry("400x200")  tk.Label(root, text="Выберите таблицу:").pack(pady=10)  table\_selector = ttk.Combobox(root, values=tables)  table\_selector.pack(pady=10)  tk.Button(root, text="Показать таблицу", command=show\_table).pack(pady=20)  root.mainloop()  def login\_window():  def try\_login():  username = entry\_username.get()  password = entry\_password.get()  if not username or not password:  messagebox.showerror("Ошибка", "Введите логин и пароль")  return  connection = connect\_to\_db(username, password)  if connection:  main\_window(connection, username)  login\_win.destroy()  login\_win = tk.Tk()  login\_win.title("Вход в систему")  login\_win.geometry("300x200")  tk.Label(login\_win, text="Логин:").pack(pady=10)  entry\_username = tk.Entry(login\_win)  entry\_username.pack(pady=5)  tk.Label(login\_win, text="Пароль:").pack(pady=10)  entry\_password = tk.Entry(login\_win, show="\*")  entry\_password.pack(pady=5)  tk.Button(login\_win, text="Войти", command=try\_login).pack(pady=20)  login\_win.mainloop()  if \_\_name\_\_ == "\_\_main\_\_":  login\_window() |