# **Работа с базой данных**

1. **Установка XAMPP**
   1. **Подготовка к установке**

Приложение уже установлено

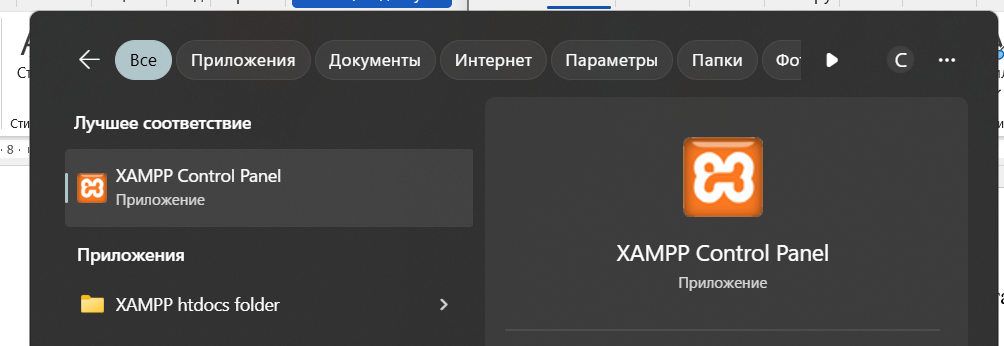


Рисунок 1 – Установленное приложение

* 1. **Запуск приложения**

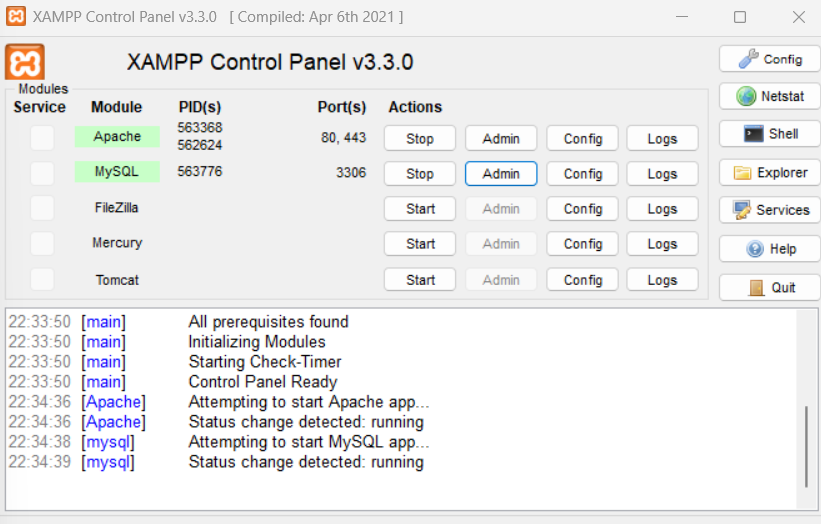


Рисунок 2 - Панель XAMPP

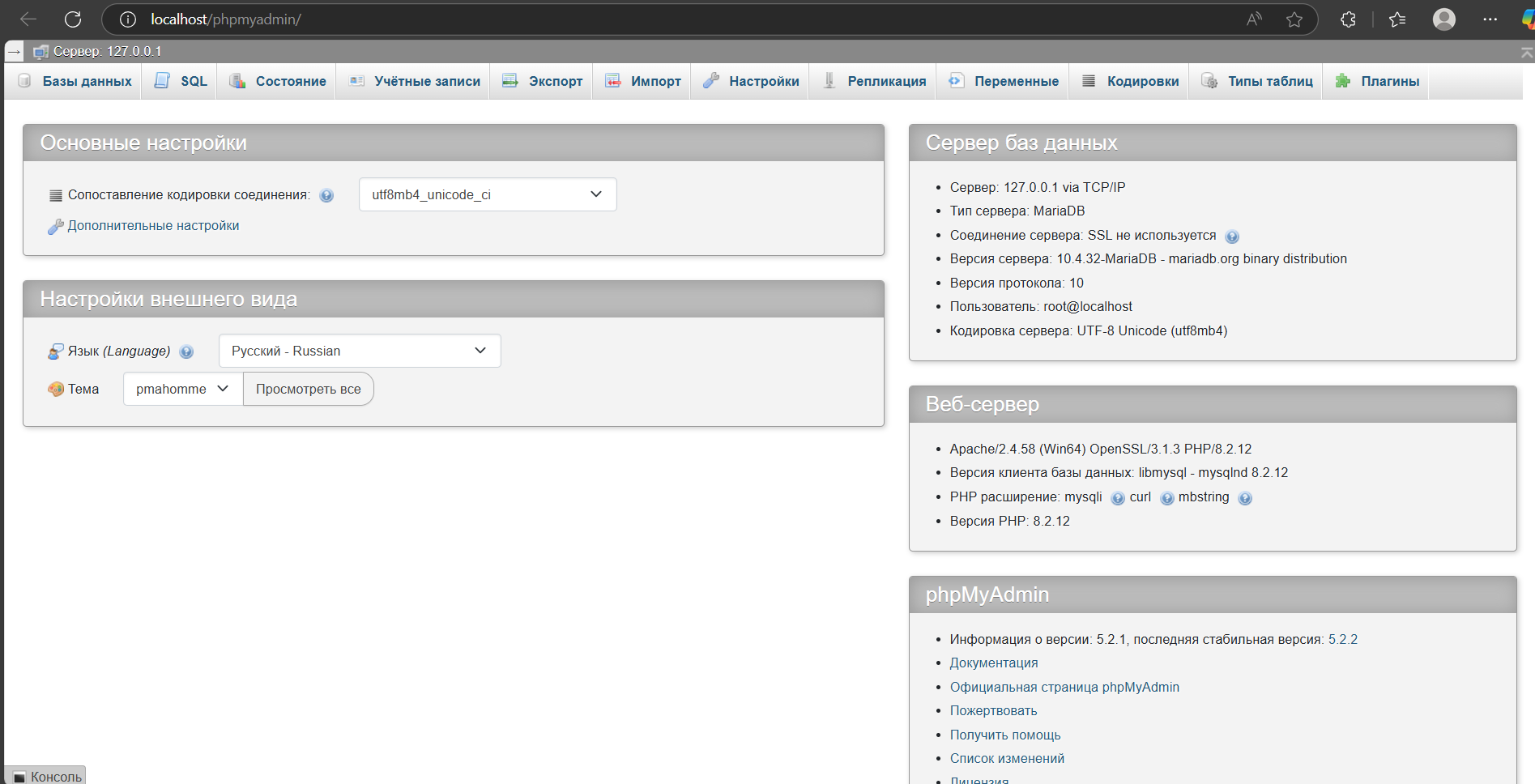


Рисунок 3 - Панель phpMyAdmin

**2. Работа с БД**

**2.1. Создание БД**

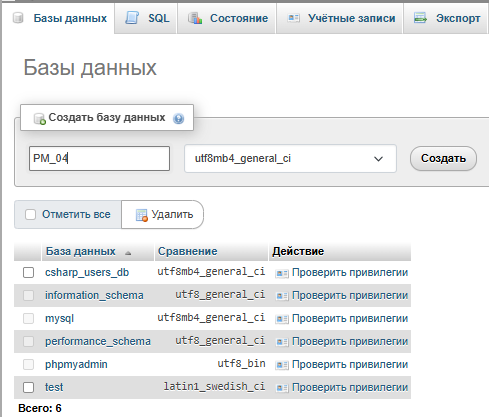


Рисунок 4 - Создание новой БД

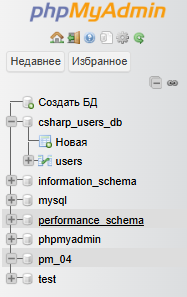


Рисунок 5 - Выбор созданной БД

**2.2. Импорт тестовой БД**

Задача: настроить структуру базы данных и заполнить ее предоставленными данными.

1. Вставка операторов CREATE TABLE:

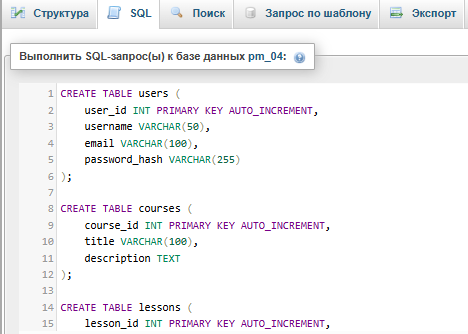


Рисунок 6 – Пример работы с SQL-запросом

1. Вставка операторов INSERT INTO:

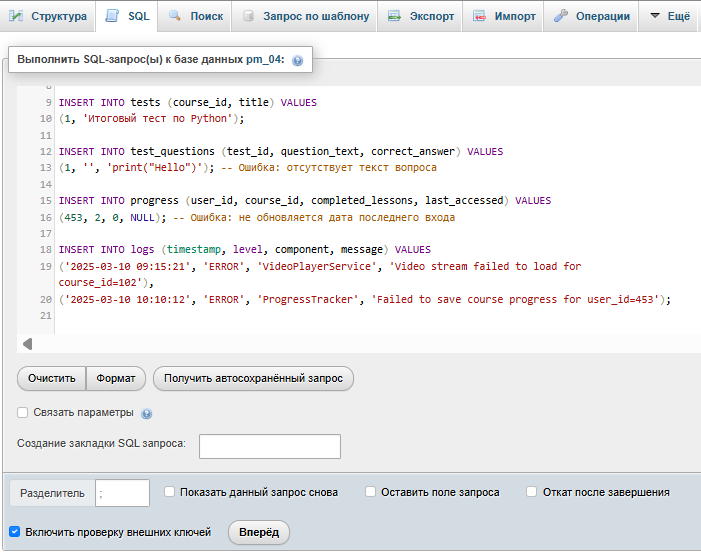


Рисунок 7 - Пример данных для вставки

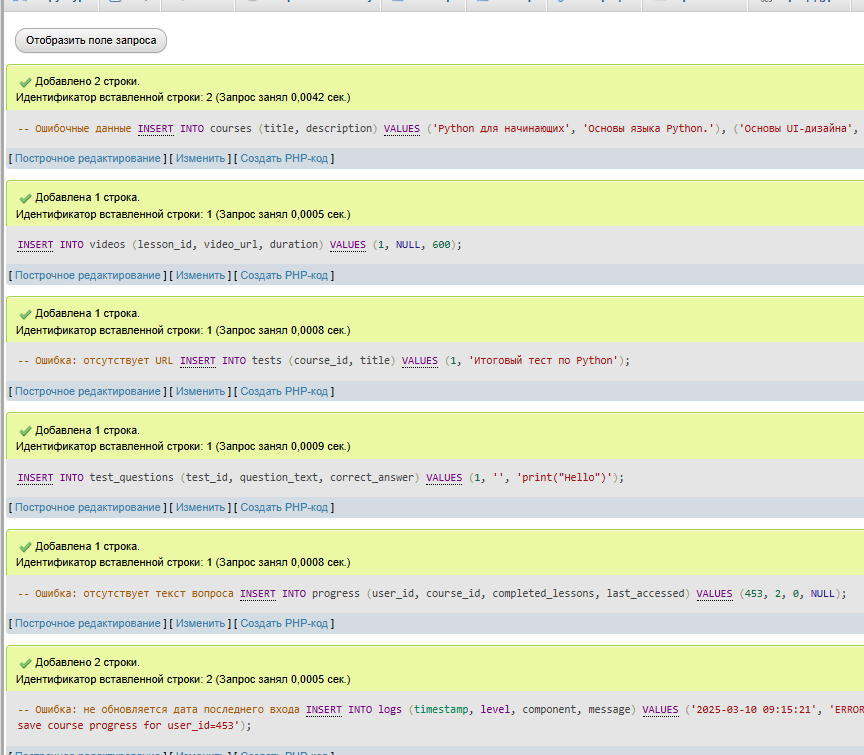


Рисунок 8 - Успешная вставка

1. Проверка данных

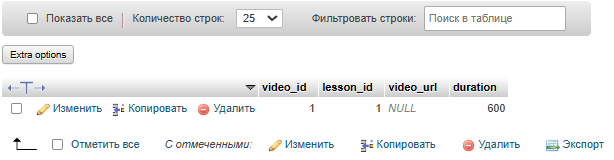


Рисунок 9 – Пример проблем в БД

**2.3. Пример запросов на создание и вставку в БД**

Запрос на создание таблиц:

-- Создание таблицы пользователей

CREATE TABLE users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50),

email VARCHAR(100),

password\_hash VARCHAR(255));

-- Создание таблицы платежей

CREATE TABLE payments (

payment\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

amount DECIMAL(10, 2),

status VARCHAR(20),

transaction\_date DATETIME,

transaction\_details TEXT,

FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

-- Создание таблицы логов ошибок

CREATE TABLE logs (

log\_id INT PRIMARY KEY AUTO\_INCREMENT,

timestamp DATETIME,

level VARCHAR(20),

component VARCHAR(50),

message TEXT

);

-- Создание таблицы кодов ошибок

CREATE TABLE error\_codes (

error\_code\_id INT PRIMARY KEY AUTO\_INCREMENT,

error\_code VARCHAR(50),

description TEXT

);

-- Создание таблицы истории транзакций

CREATE TABLE transaction\_history (

history\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

payment\_id INT,

previous\_status VARCHAR(20),

FOREIGN KEY (user\_id) REFERENCES users(user\_id),

FOREIGN KEY (payment\_id) REFERENCES payments(payment\_id)

);

-- Создание таблицы возвратов

CREATE TABLE refunds (

refund\_id INT PRIMARY KEY AUTO\_INCREMENT,

payment\_id INT,

amount DECIMAL(10, 2),

refund\_date DATETIME,

error\_message TEXT,

FOREIGN KEY (payment\_id) REFERENCES payments(payment\_id)

);

-- Вставка пользователей

INSERT INTO users (username, email, password\_hash) VALUES

('user101', 'user101@example.com', 'hashedpassword1'),

('user102', 'user102@example.com', 'hashedpassword2'),

('user103', 'user103@example.com', 'hashedpassword3');

-- Вставка данных о платежах

INSERT INTO payments (user\_id, amount, status, transaction\_date, transaction\_details) VALUES

(1, 100.00, 'failed', '2025-07-01 10:00:00', 'Ошибка: некорректные реквизиты'),

(2, 50.00, 'success', '2025-07-01 10:05:00', 'Платеж прошел успешно'),

(3, 200.00, 'failed', '2025-07-01 10:10:00', 'Ошибка: недостаточно средств');

-- Вставка логов ошибок

INSERT INTO logs (timestamp, level, component, message) VALUES

('2025-07-01 09:15:21', 'ERROR', 'PaymentProcessor', 'Ошибка при обработке платежа для user\_id=1, некорректные реквизиты'),

('2025-07-01 09:45:12', 'ERROR', 'RefundService', 'Ошибка при обработке возврата для payment\_id=3, недостаточно средств');

-- Вставка возвратов

INSERT INTO refunds (payment\_id, amount, refund\_date, error\_message) VALUES

(3, 50.00, '2025-07-02 10:00:00', 'Ошибка при возврате средств, недостаточно средств для возврата');

**2.4. Анализ проблем с БД**

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM logs WHERE error\_code [IS](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/comparison-operators.html%23operator_is) NULL;

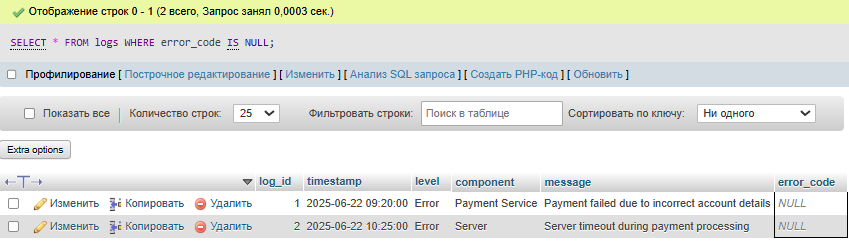


Рисунок 10. Ожидаемый результат: возвращается строка total\_price=NULL

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM logs WHERE level = 'Error' [AND](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/logical-operators.html%23operator_and) component = 'Payment Service';

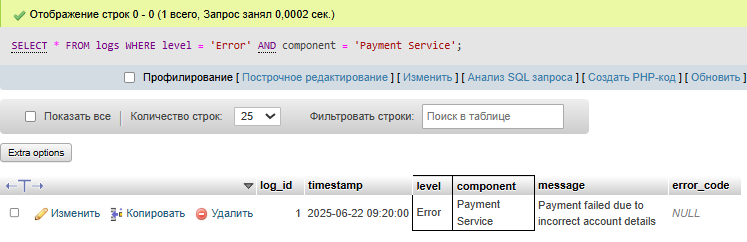


Рисунок 11

SELECT \* FROM clients WHERE phone\_number like '+8%';

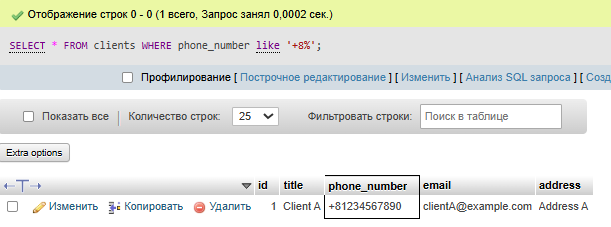


Рисунок 12. Ожидаемый результат: возвращается строка phone\_number like ‘+8%’

**2.5. Проверка отсутствия ограничений**

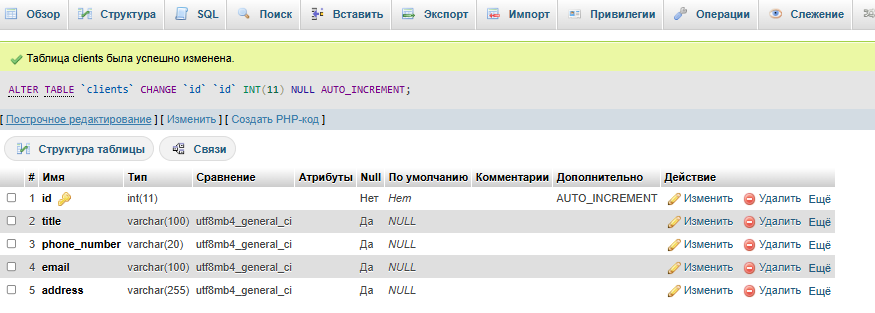


Рисунок 13 - Пример работы со структурой

**2.6. Проверка производительности**

[EXPLAIN](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/explain.html) [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM clients WHERE id = 4;

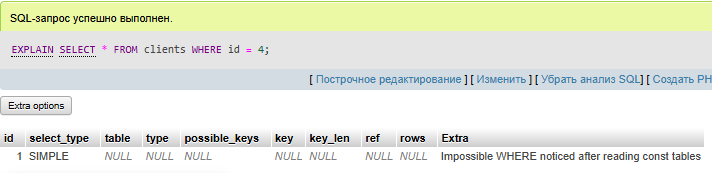


Рисунок 14 – Пример полного сканирования таблиц

**2.7. Устранение проблем в БД**

Цель: исправить ошибочные данные и добавить ограничения для предотвращения проблем в будущем.

1. Заполнение пустых значений в обязательных полях:

UPDATE clients

SET phone\_number = '+0000000000', email = 'default@example.com', address = 'Unknown'

WHERE phone\_number IS NULL OR email IS NULL OR address IS NULL;

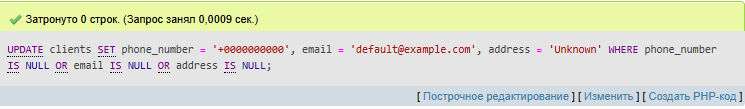


Рисунок 15

1. Исправление некорректных данных (например, неправильный формат email):

UPDATE clients

SET email = CONCAT('user', id, '@example.com')

WHERE email NOT REGEXP '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$';



Рисунок 16

1. Добавление ограничения для обязательных полей

ALTER TABLE clients

MODIFY phone\_number VARCHAR(20) NOT NULL,

MODIFY email VARCHAR(100) NOT NULL,

MODIFY address VARCHAR(255) NOT NULL;

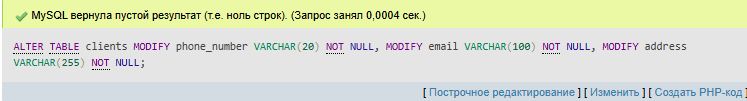


Рисунок 17

1. Добавление уникальности для email (если email должен быть уникальным):

ALTER TABLE clients

ADD CONSTRAINT unique\_email UNIQUE (email);

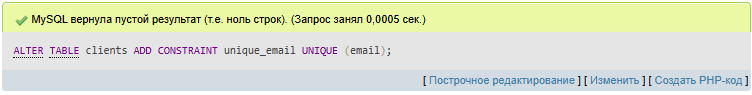


Рисунок 18

1. Заполнение пустых таблиц:

INSERT INTO logs (timestamp, level, component, message, error\_code) VALUES

('2025-06-22 09:20:00', 'Error', 'Payment Service', 'Payment failed due to incorrect account details', 'ERR001'),

('2025-06-22 10:25:00', 'Error', 'Server', 'Server timeout during payment processing', 'ERR002'),

('2025-06-22 11:30:00', 'Warning', 'Database', 'High load detected during transaction processing', 'WARN001');

INSERT INTO users (first\_name, last\_name, email, username, password\_hash, role) VALUES

('John', 'Doe', 'john.doe@example.com', 'johndoe', 'hashed\_password\_123', 'Admin'),

('Jane', 'Smith', 'jane.smith@example.com', 'janesmith', 'hashed\_password\_456', 'User'),

('Robert', 'Brown', 'robert.brown@example.com', 'robertbrown', 'hashed\_password\_789', 'User');

INSERT INTO error\_codes (error\_code, description) VALUES

('ERR001', 'Incorrect account details provided'),

('ERR002', 'Server timeout during payment processing'),

('WARN001', 'High load detected during transaction processing');



Рисунок 19

**2.8. Добавление ограничений внешнего ключа**

1) Добавление внешнего ключа в таблицу payments:

ALTER TABLE payments

ADD CONSTRAINT fk\_user\_id

FOREIGN KEY (user\_id) REFERENCES clients(id)

ON DELETE CASCADE

ON UPDATE CASCADE;

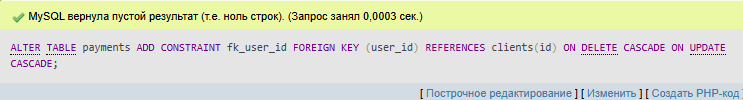


Рисунок 20

2) Добавление индексов производительности.

CREATE UNIQUE INDEX idx\_unique\_email ON clients (email);

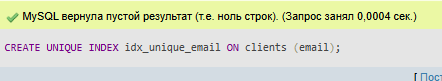


Рисунок 21

CREATE INDEX idx\_phone\_number ON clients (phone\_number);

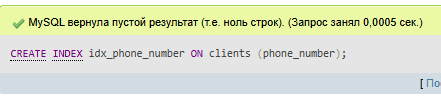


Рисунок 22

CREATE INDEX idx\_user\_id ON payments (user\_id);

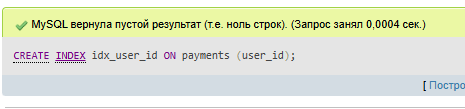


Рисунок 23

**2.9. Тестирование исправлений**

Цель: убедиться, что исправления в базе данных устраняют проблемы.

SELECT \* FROM clients WHERE phone\_number IS NULL OR email IS NULL OR address IS NULL;

SELECT \* FROM payments WHERE user\_id IS NULL OR amount IS NULL OR status IS NULL OR transaction\_date IS NULL;

SELECT \* FROM logs WHERE timestamp IS NULL OR level IS NULL OR component IS NULL OR message IS NULL;

SELECT \* FROM users WHERE first\_name IS NULL OR last\_name IS NULL OR email IS NULL OR username IS NULL OR password\_hash IS NULL;

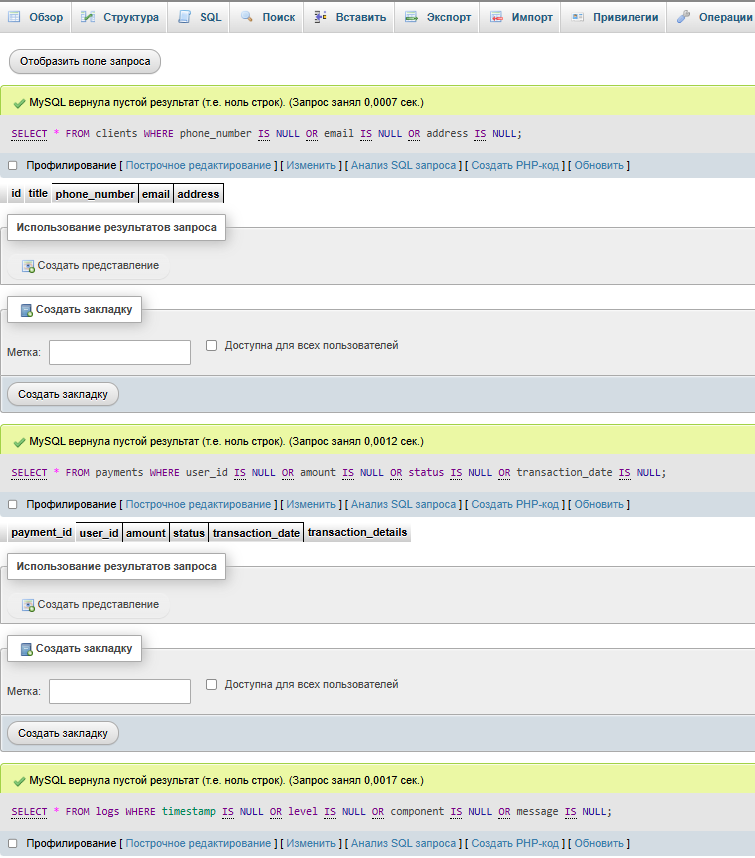


Рисунок 24

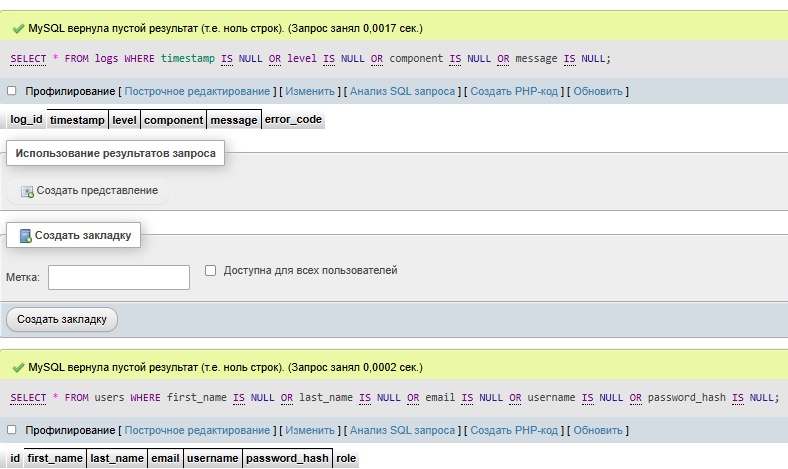


Рисунок 25

**2.10. Имитировать запросы приложений**

1) Проверка целостности данных

INSERT INTO payments (user\_id, amount, status, transaction\_date, transaction\_details)

VALUES (10, 100.00, 'Completed', '2025-06-01 10:00:00', 'Payment for invalid client');

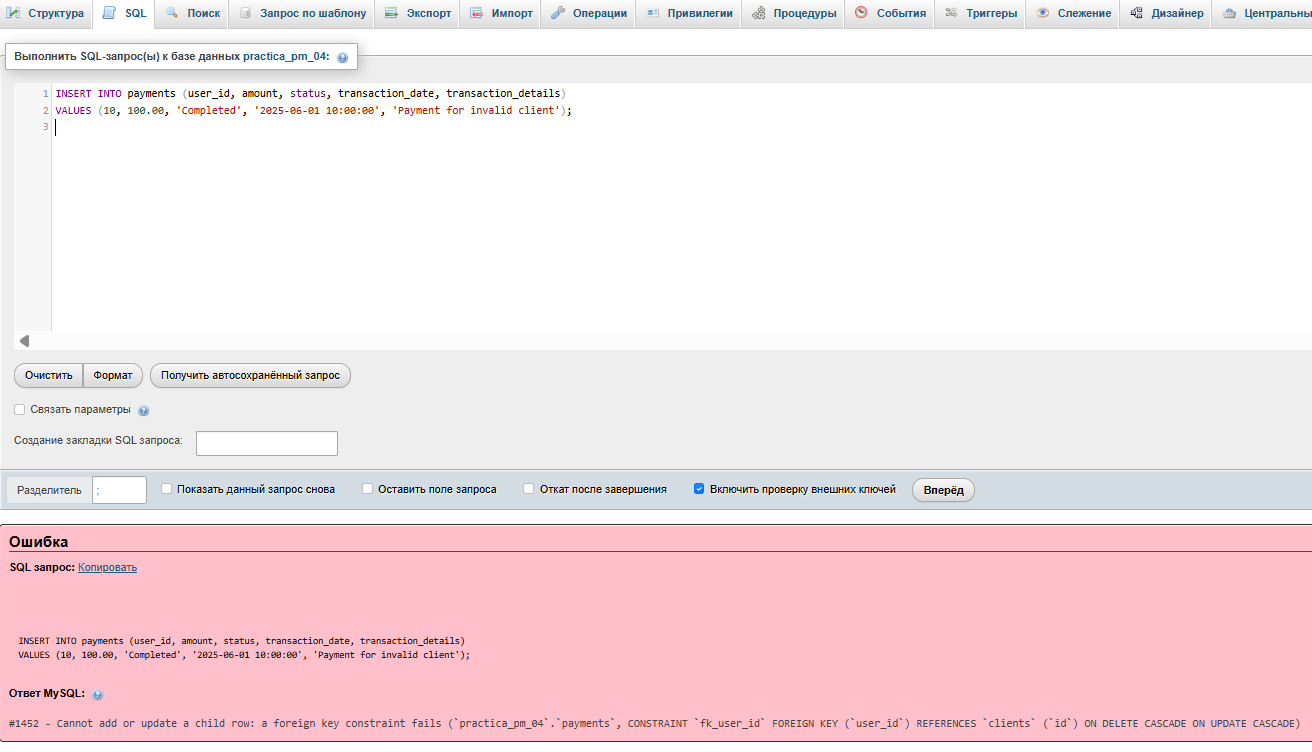


Рисунок 26

2) Запрос для регистрации клиента:

INSERT INTO clients (title, phone\_number, email, address)

VALUES ('Client D', '+79998887766', 'clientD@example.com', '7890 Birch Street, City D');

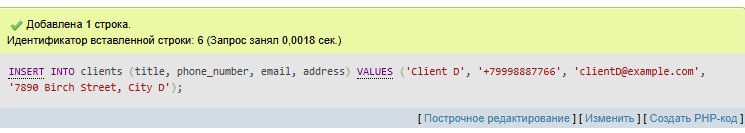


Рисунок 27

3) Проверка возврата:

SELECT r.refund\_id, r.amount, r.refund\_date, p.transaction\_details

FROM refunds r

JOIN payments p ON r.payment\_id = p.payment\_id

WHERE p.user\_id = 4;

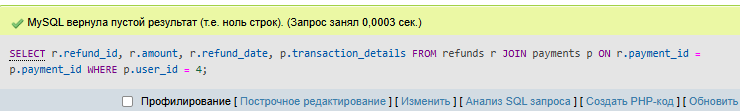


Рисунок 28

4) Создание возврата для платежа клиента:

INSERT INTO refunds (payment\_id, amount, refund\_date, error\_message)

VALUES (1, 100.00, '2025-06-06 15:00:00', 'Partial refund for order 987');

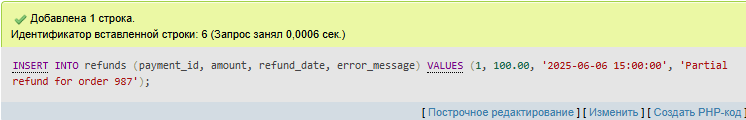


Рисунок 29

5) Проверка работоспособности:

SELECT \* FROM clients WHERE phone\_number IS NULL OR email IS NULL OR address IS NULL;

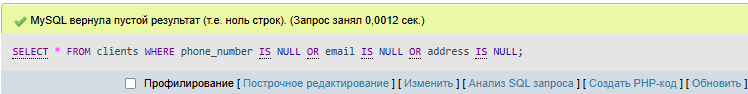


Рисунок 30

**2.11. Экспорт базы данных:**

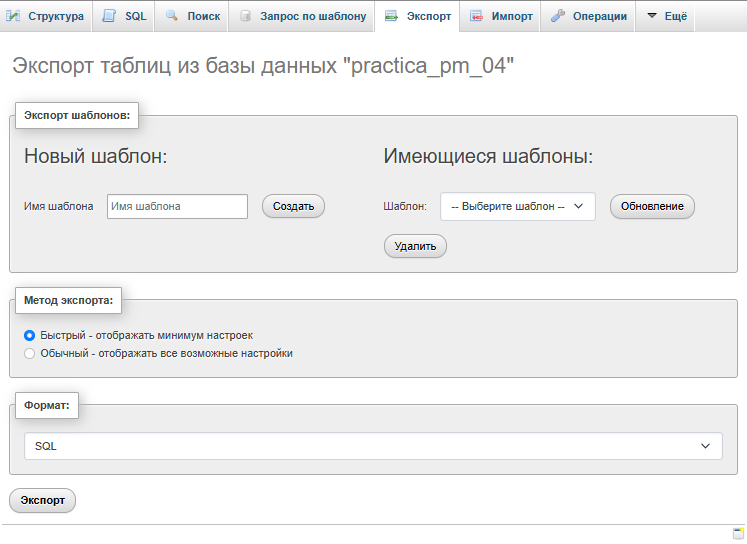


Рисунок 31 – Экспорт данных

