**Использование хранимых функций и процедур**

**«Использование триггеров**

**Вывести список всех специальностей и количество студентов в каждой специальности.**

**DELIMITER //**

**CREATE PROCEDURE specialnost\_count\_students()**

**BEGIN**

**SELECT specialnost, kol\_vo\_studentov**

**FROM specialnost;**

**END//**

**DELIMITER ;**

**Вывести список всех студентов из определенной группы**

**DELIMITER //**

**CREATE PROCEDURE students\_in\_group(IN group\_id VARCHAR(45))**

**BEGIN**

**SELECT imia\_studenta, nomer\_gr, strana, nomer\_stud\_bileta, sroc\_obucheniya**

**FROM student**

**WHERE nomer\_gr = group\_id;**

**END//**

**DELIMITER ;**

**Вывести список всех стран и столиц.**

**DELIMITER //**

**CREATE PROCEDURE countries\_capitals()**

**BEGIN**

**SELECT strana, stolica**

**FROM strana;**

**END//**

**DELIMITER ;**

**Вывести список всех религий и их основателей.**

**DELIMITER //**

**CREATE PROCEDURE religions\_founders()**

**BEGIN**

**SELECT vera, posledovateli**

**FROM vera;**

**END//**

**DELIMITER ;**

**Вывести список всех стран и их президентов в алфавитном порядке.**

**sql**

**DELIMITER //**

**CREATE PROCEDURE countries\_presidents()**

**BEGIN**

**SELECT strana, prezident**

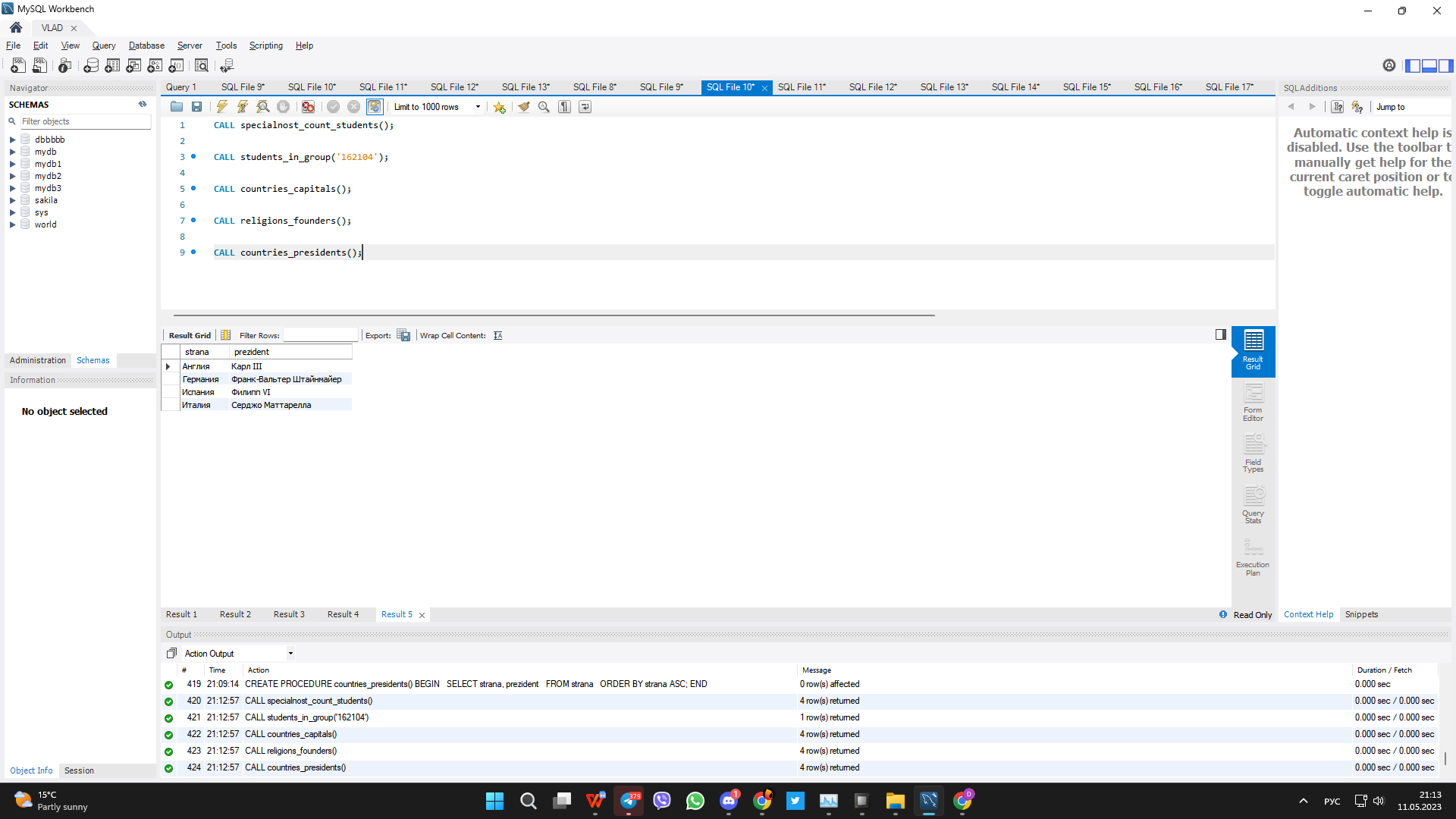
**FROM strana**

**ORDER BY strana ASC;**

**END//**

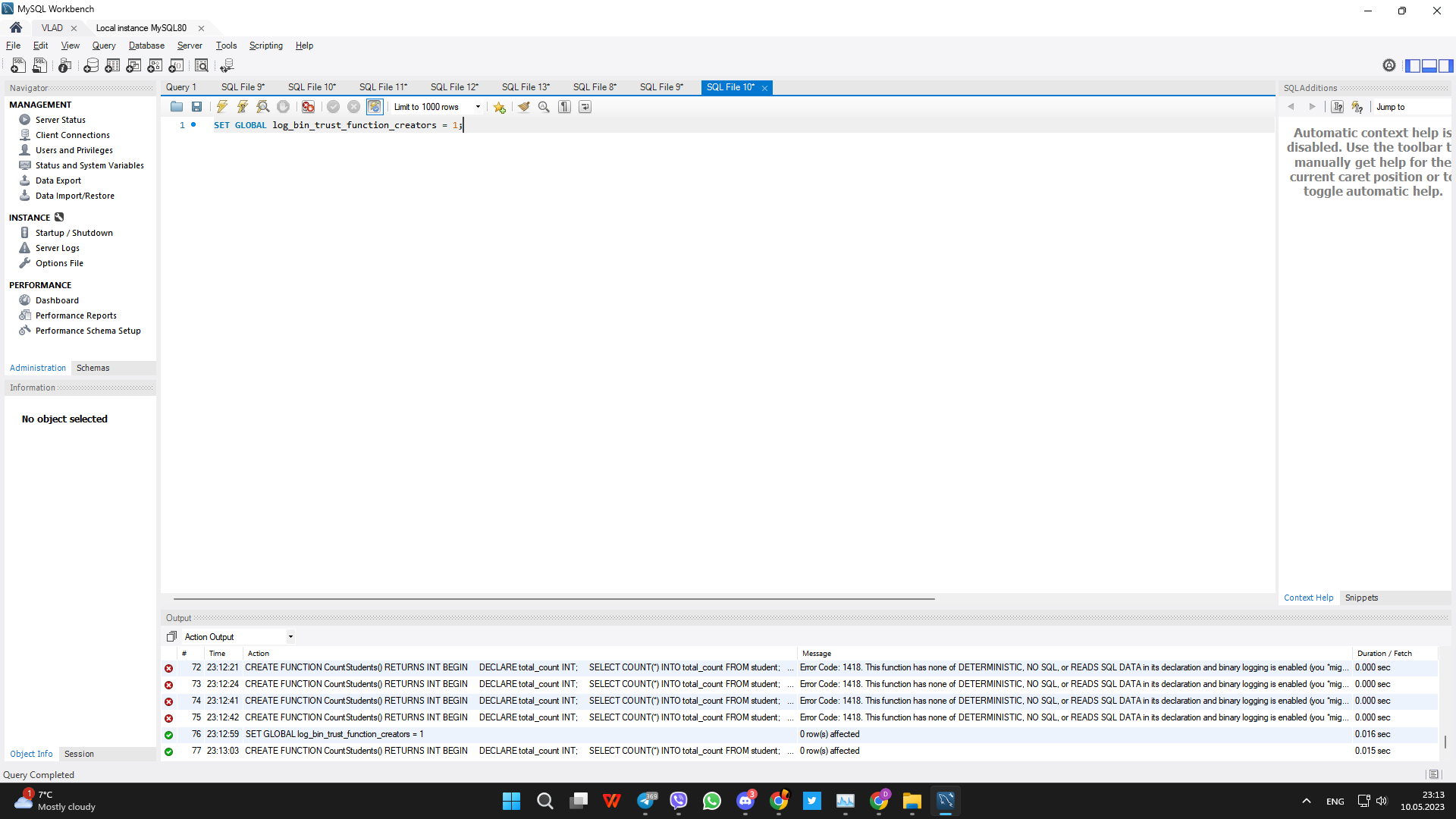
**DELIMITER ;**

**Пруфы вызова того что всё работает**



**Функции:**

**Прииспользовании функций что ю не возникала ошибка 1418 нам надо сделать вот это**

****

**Тогда эта ошибка уйдёт и можно использовать функции**

**Функция для подсчета общего количества студентов в базе данных:**

**DELIMITER //**

**CREATE FUNCTION CountStudents()**

**RETURNS INT**

**BEGIN**

**DECLARE total\_count INT;**

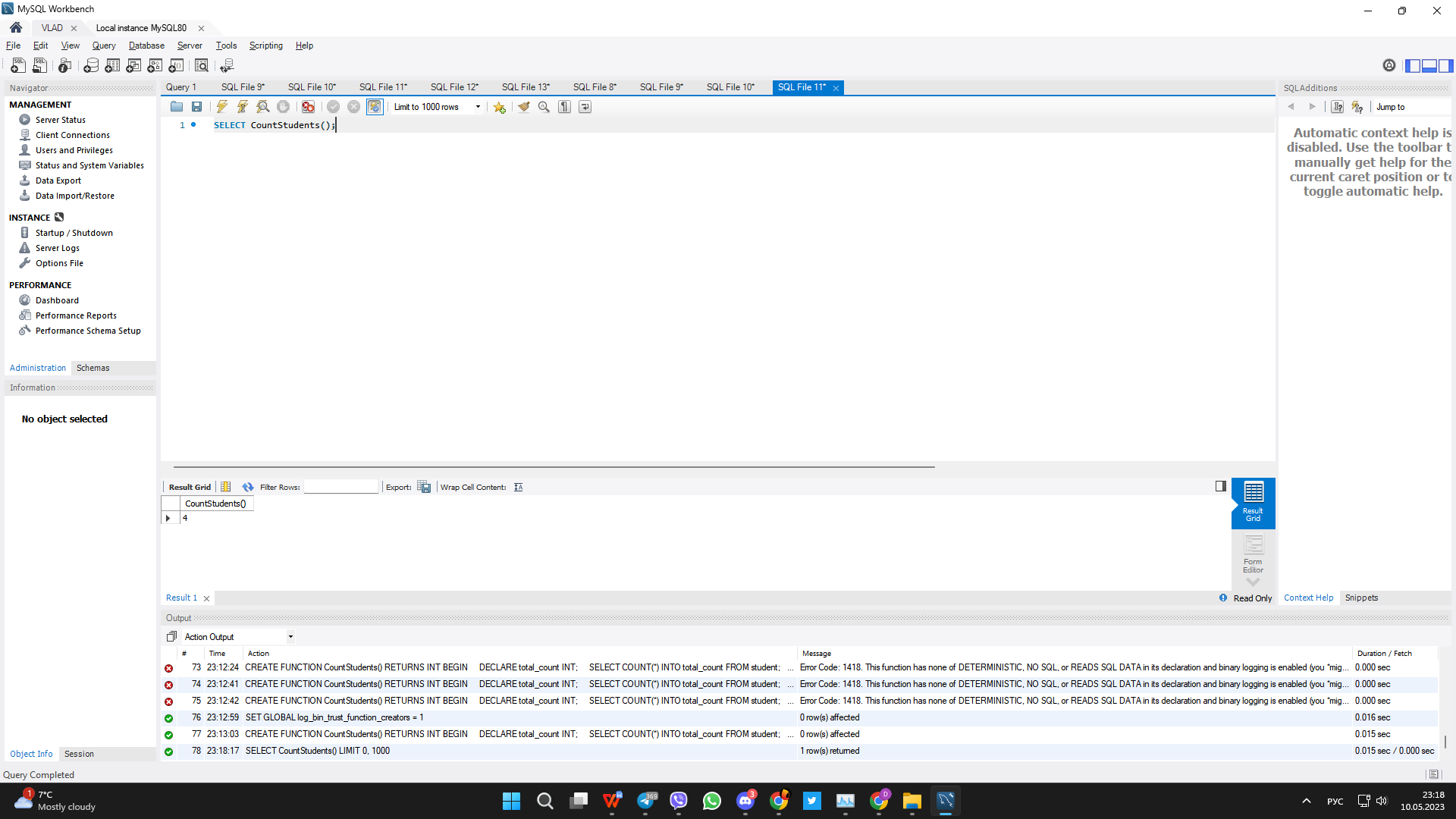
**SELECT COUNT(\*) INTO total\_count FROM student;**

**RETURN total\_count;**

**END //**

**DELIMITER ;**

**Вот пример использования этой функции**



**Ещё одна функция которая получает средний балл по определенной специальности:**

**DELIMITER //**

**CREATE FUNCTION GetAverageScore(specialty\_name VARCHAR(45))**

**RETURNS DECIMAL(4, 2)**

**BEGIN**

**DECLARE avg\_score DECIMAL(4, 2);**

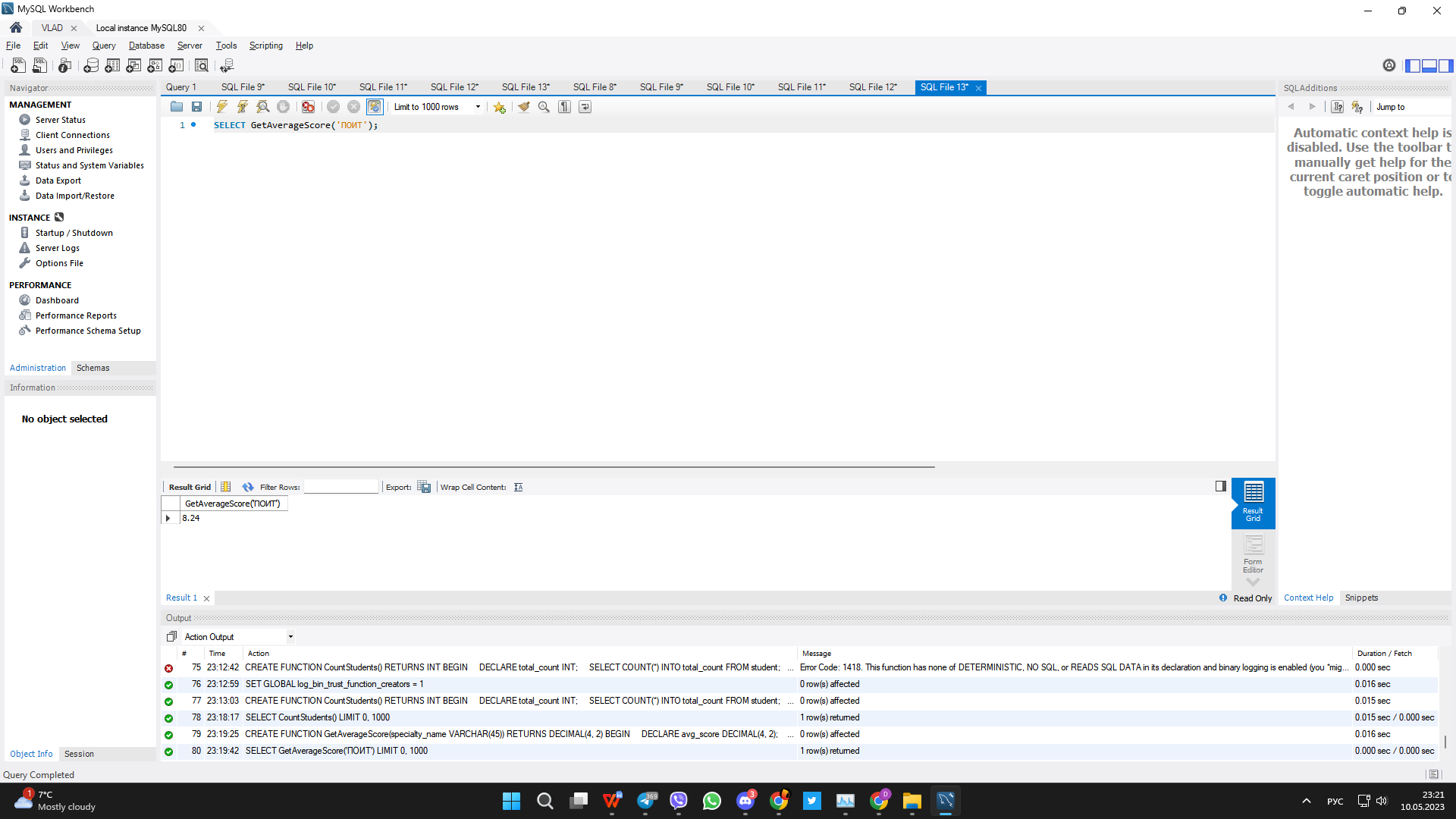
**SELECT sredniy\_ball INTO avg\_score FROM specialnost WHERE specialnost = specialty\_name;**

**RETURN avg\_score;**

**END //**

**DELIMITER ;**

**А вот пример её использования**



**Функция для получения среднего балла по определенной специальности.**

**DELIMITER //**

**CREATE FUNCTION get\_average\_grade(specialnost\_name VARCHAR(45))**

**RETURNS DECIMAL(10,2)**

**BEGIN**

**DECLARE avg\_grade DECIMAL(10,2);**

**SELECT sredniy\_ball INTO avg\_grade**

**FROM specialnost**

**WHERE specialnost = specialnost\_name;**

**RETURN avg\_grade;**

**END//**

**DELIMITER ;**

**Функция для получения количества студентов в определенной группе.**

**DELIMITER //**

**CREATE FUNCTION count\_students\_in\_group(group\_id VARCHAR(45))**

**RETURNS INT**

**BEGIN**

**DECLARE student\_count INT;**

**SELECT COUNT(\*) INTO student\_count**

**FROM student**

**WHERE nomer\_gr = group\_id;**

**RETURN student\_count;**

**END//**

**DELIMITER ;**

**Функция для получения названия специальности студента по его имени.**

**DELIMITER //**

**CREATE FUNCTION get\_student\_specialty(student\_name VARCHAR(45))**

**RETURNS VARCHAR(45)**

**BEGIN**

**DECLARE specialty\_name VARCHAR(45);**

**SELECT specialnost INTO specialty\_name**

**FROM nomer\_gr**

**WHERE nomer\_gr = (**

**SELECT nomer\_gr**

**FROM student**

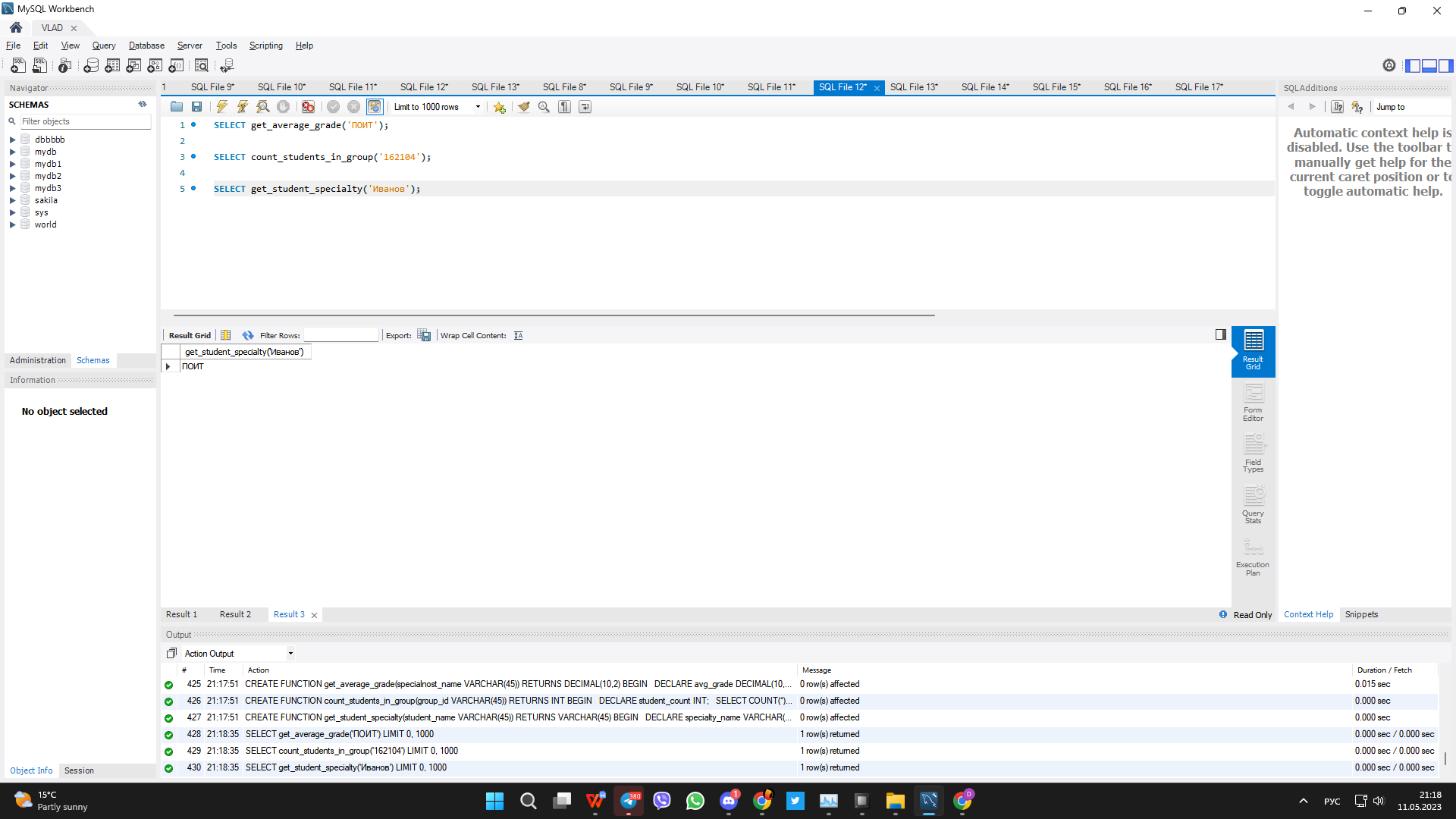
**WHERE imia\_studenta = student\_name**

**);**

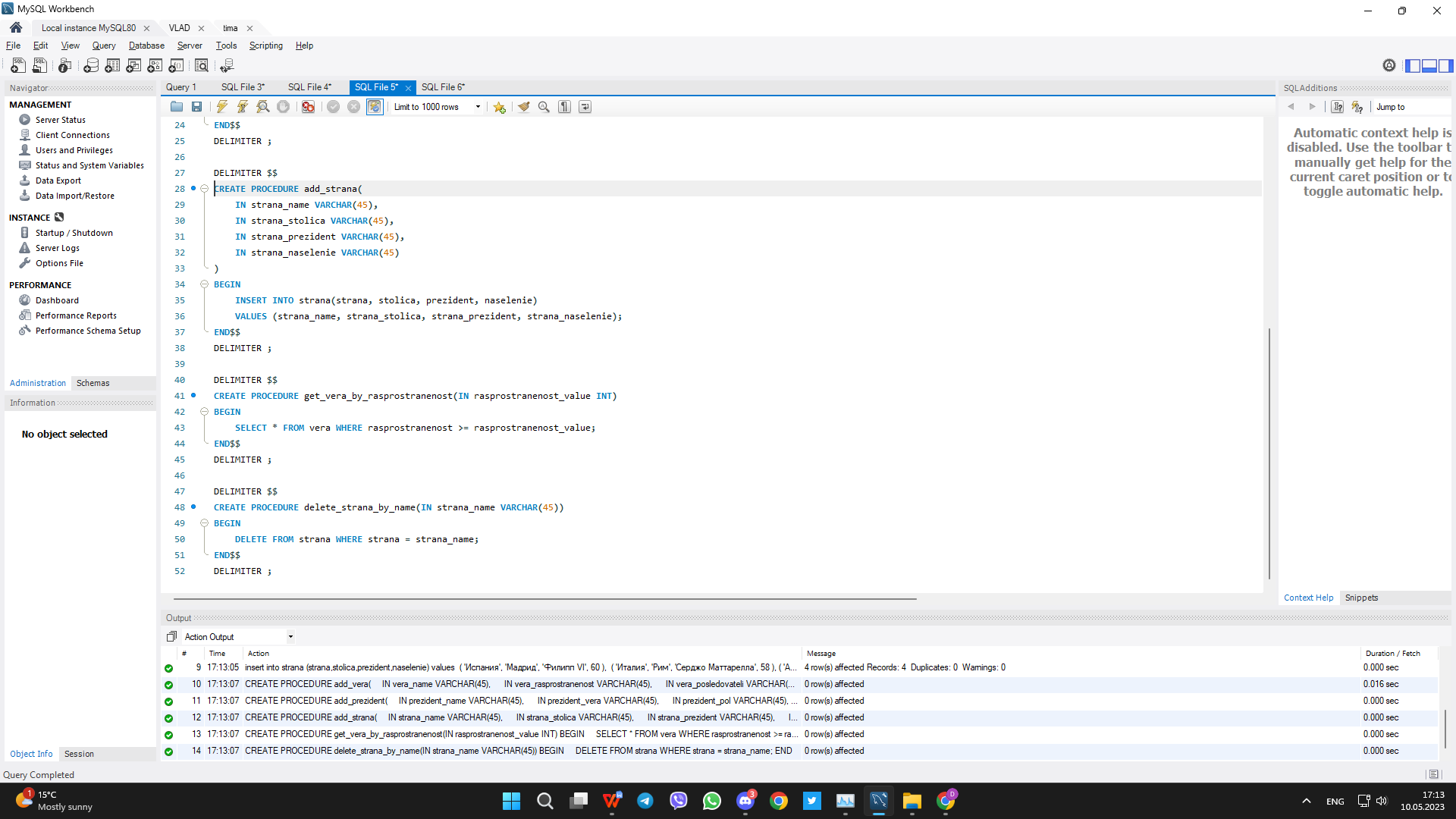
**RETURN specialty\_name;**

**END//**

**DELIMITER ;**



**А вот доказательный скриншот что там всё запускалось**



**Тригеры :**

**Триггер для проверки возраста студентов при добавлении записи в таблицу "student":**

**DELIMITER //**

**CREATE TRIGGER update\_avg\_grade AFTER INSERT ON student**

**FOR EACH ROW**

**BEGIN**

**UPDATE specialnost**

**SET sredniy\_ball = (**

**SELECT AVG(sroc\_obucheniya)**

**FROM student**

**WHERE nomer\_gr = NEW.nomer\_gr**

**)**

**WHERE specialnost = (**

**SELECT specialnost**

**FROM nomer\_gr**

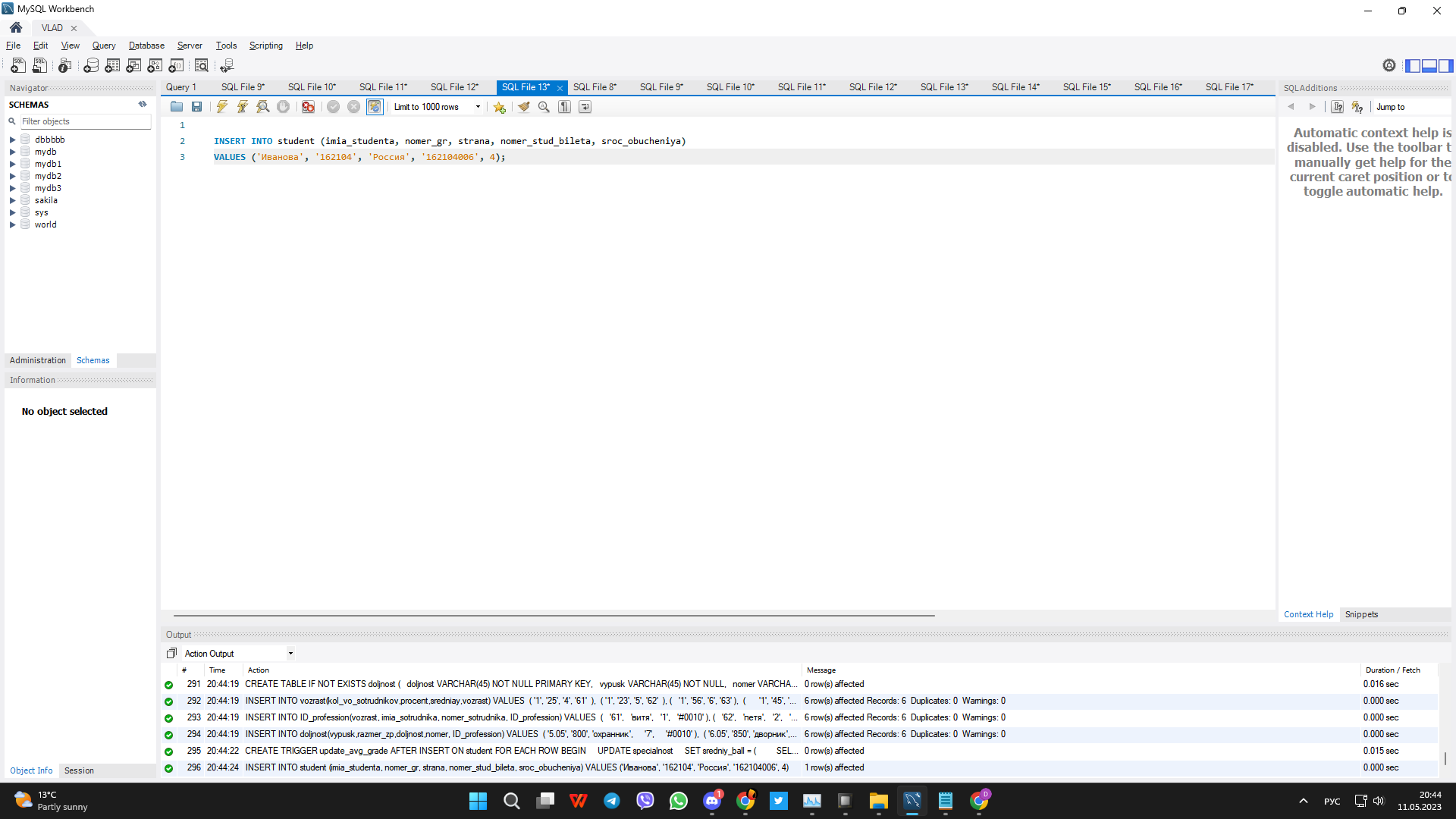
**WHERE nomer\_gr = NEW.nomer\_gr**

**);**

**END//**

**DELIMITER ;**

**Проверка работы:**



**Триггер, который запрещает изменять или удалять запись из таблицы "prezident", если в таблице "strana" есть запись с тем же значением поля "prezident" в качестве внешнего ключа. Такой триггер можно написать так:**

**DELIMITER $$**

**CREATE TRIGGER prezident\_trig1**

**BEFORE UPDATE ON prezident**

**FOR EACH ROW**

**BEGIN**

**IF EXISTS (SELECT 1 FROM strana WHERE prezident = OLD.prezident) THEN**

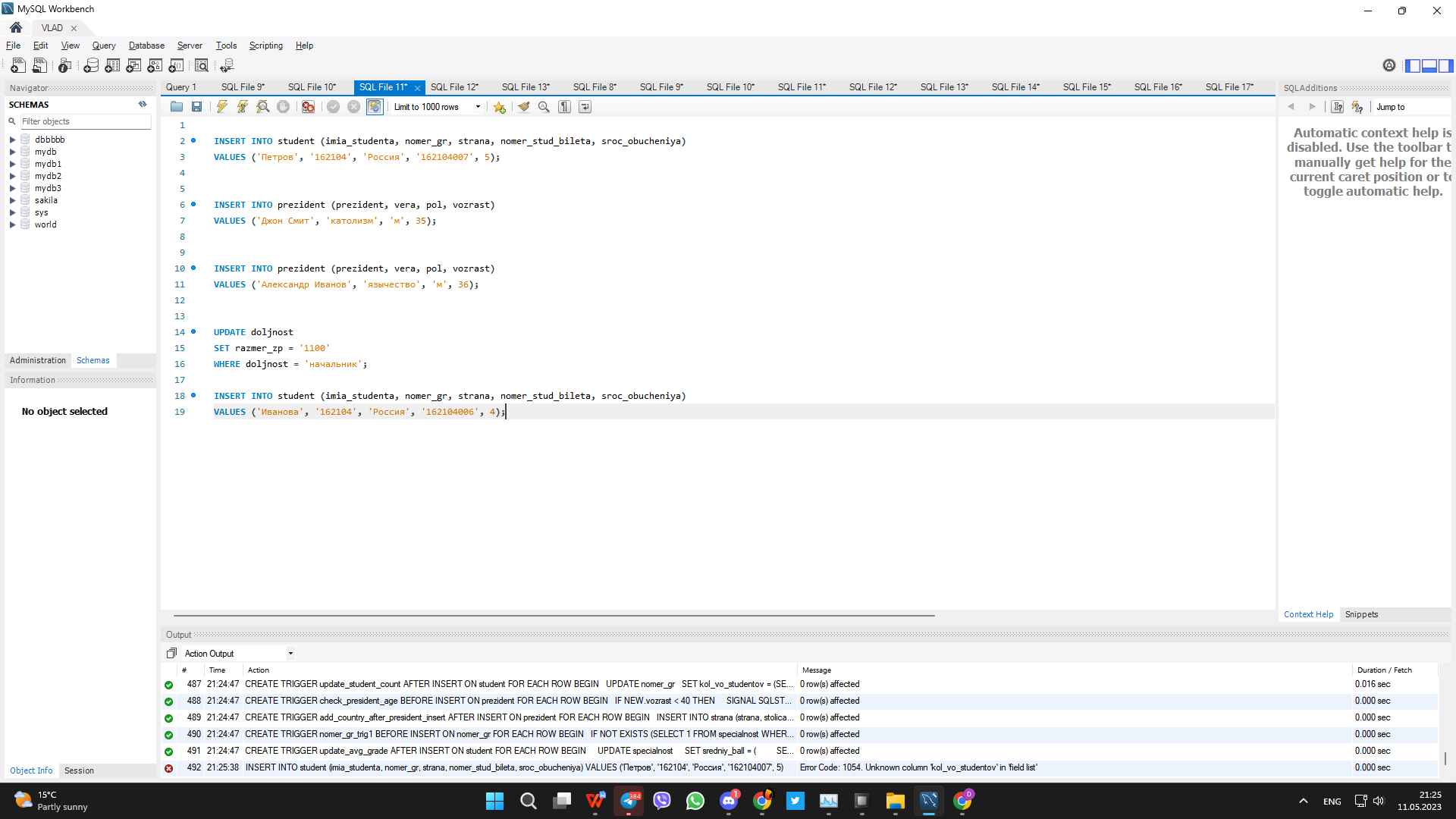
**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Cannot delete or update this row, because it is referenced by a row in strana table';**

**END IF;**

**END$$**

**DELIMITER ;**



**Триггер для проверки уникальности возраста в таблице "vozrast" при добавлении записи в таблицу "ID\_profession":**

**DELIMITER $$**

**CREATE TRIGGER check\_unique\_age**

**BEFORE INSERT ON ID\_profession**

**FOR EACH ROW**

**BEGIN**

**IF EXISTS (SELECT \* FROM ID\_profession WHERE vozrast = NEW.vozrast) THEN**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Возраст уже существует в таблице ID\_profession!';**

**END IF;**

**END$$**

**DELIMITER ;**

**Триггер для автоматического обновления размера заработной платы в таблице "doljnost" при изменении значения в таблице "vozrast":**

**DELIMITER $$**

**CREATE TRIGGER update\_salary**

**AFTER UPDATE ON vozrast**

**FOR EACH ROW**

**BEGIN**

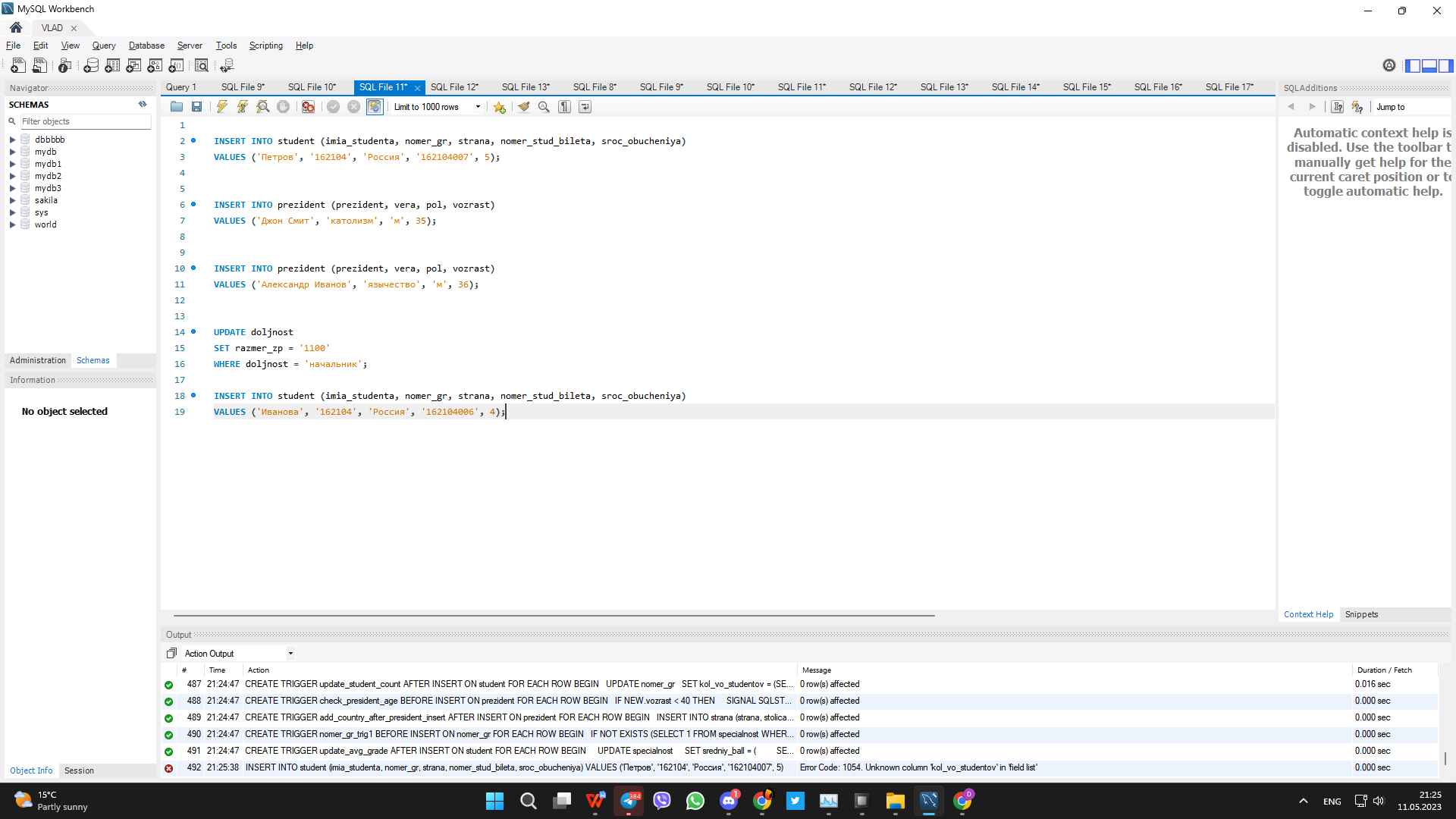
**UPDATE doljnost**

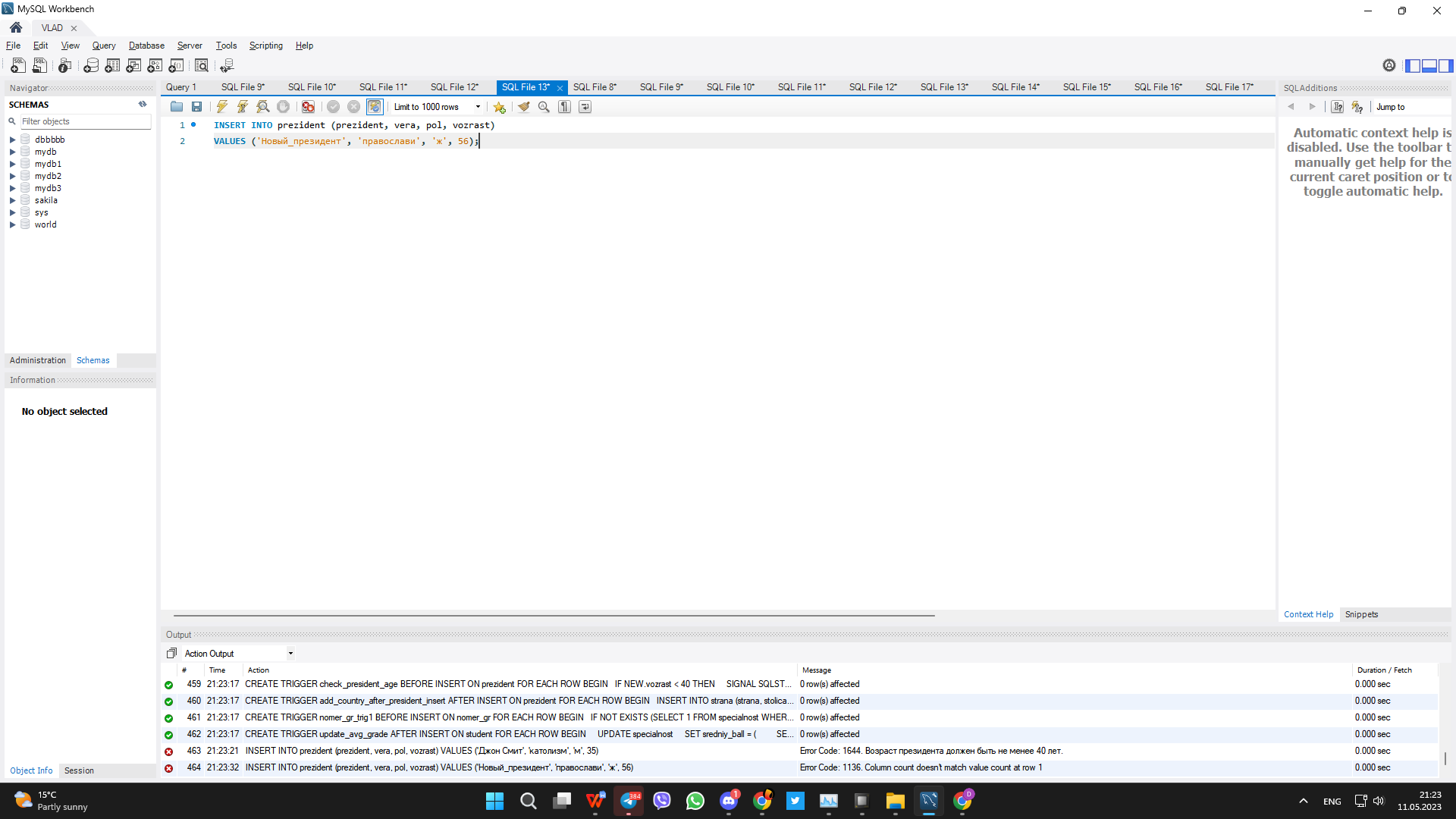
**SET razmer\_zp = NEW.sredniay \* 100**

**WHERE ID\_profession = (SELECT ID\_profession FROM ID\_profession WHERE vozrast = OLD.vozrast);**

**END$$**

**DELIMITER ;**





**Триггер, который запрещает добавлять запись в таблицу "nomer\_gr", если в таблице "specialnost" нет записи с таким же значением поля "specialnost" в качестве внешнего ключа. Такой триггер можно написать так:**

**DELIMITER $$**

**CREATE TRIGGER add\_country\_after\_president\_insert AFTER INSERT ON prezident**

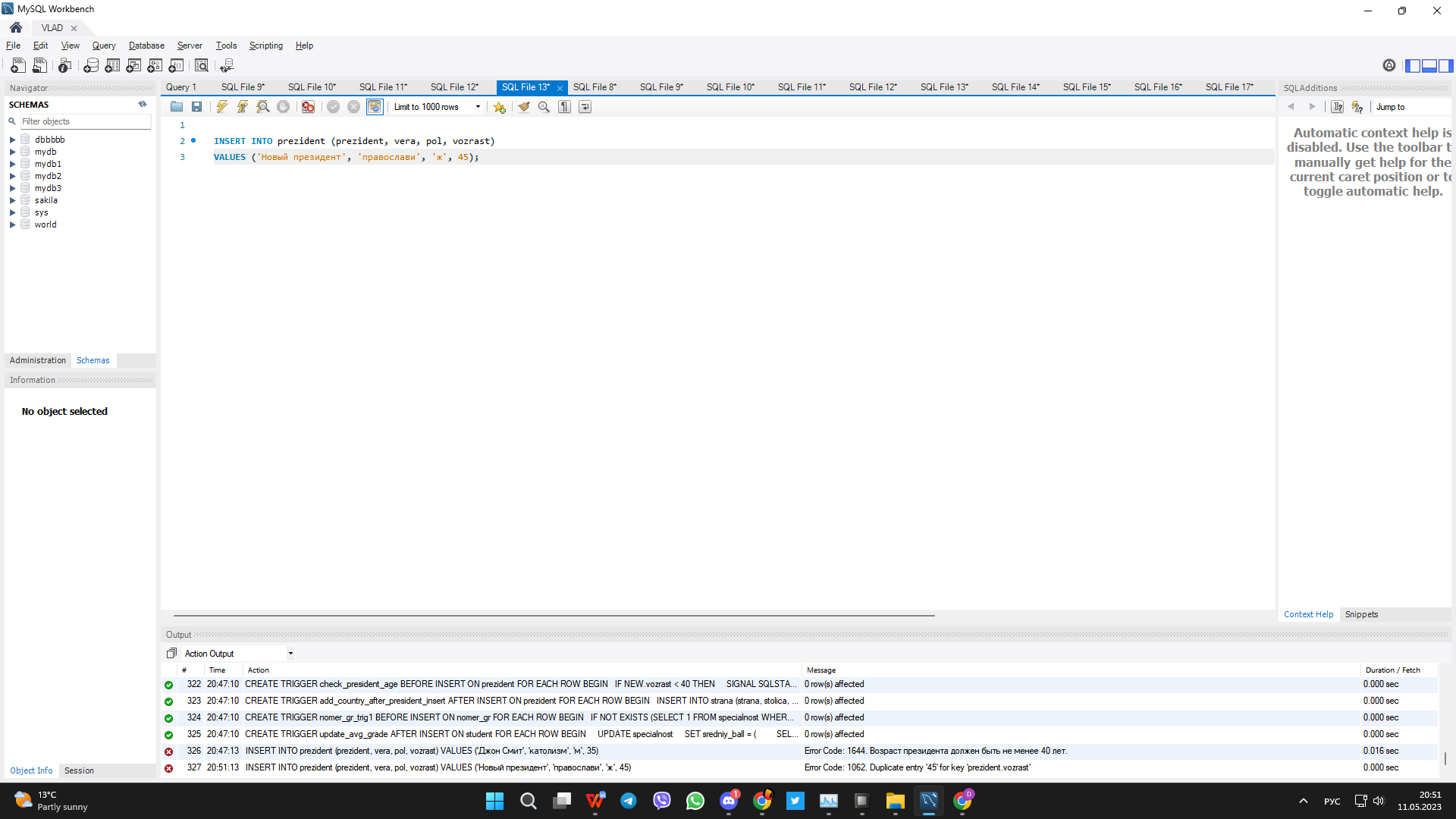
**FOR EACH ROW**

**BEGIN**

**INSERT INTO strana (strana, stolica, prezident)**

**VALUES (NEW.prezident, '', NEW.prezident);**

**END$$**

**DELIMITER ;**

**Тригер для проверки возраста призидента**

**DELIMITER $$**

**CREATE TRIGGER check\_president\_age BEFORE INSERT ON prezident**

**FOR EACH ROW**

**BEGIN**

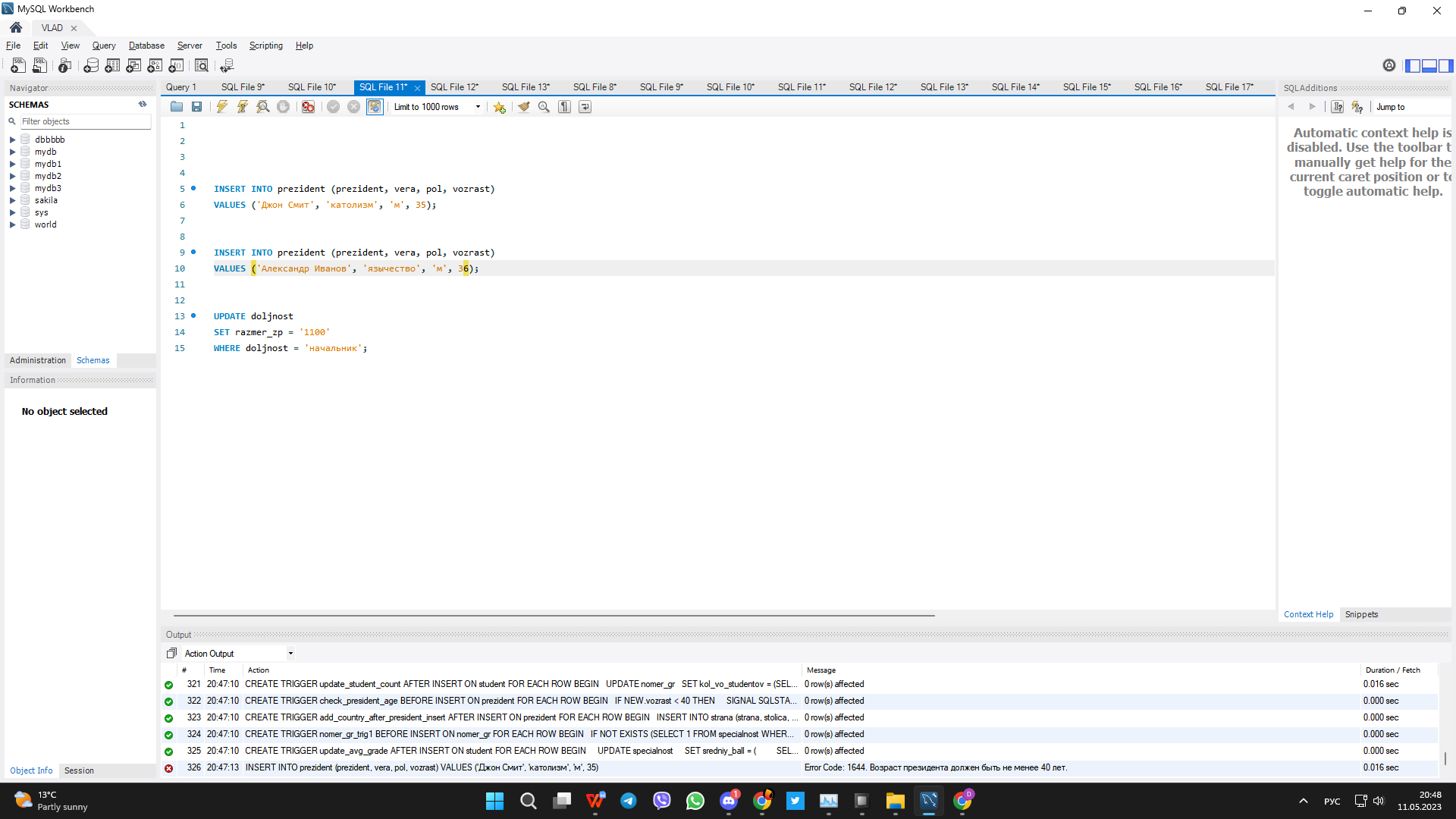
**IF NEW.vozrast < 40 THEN**

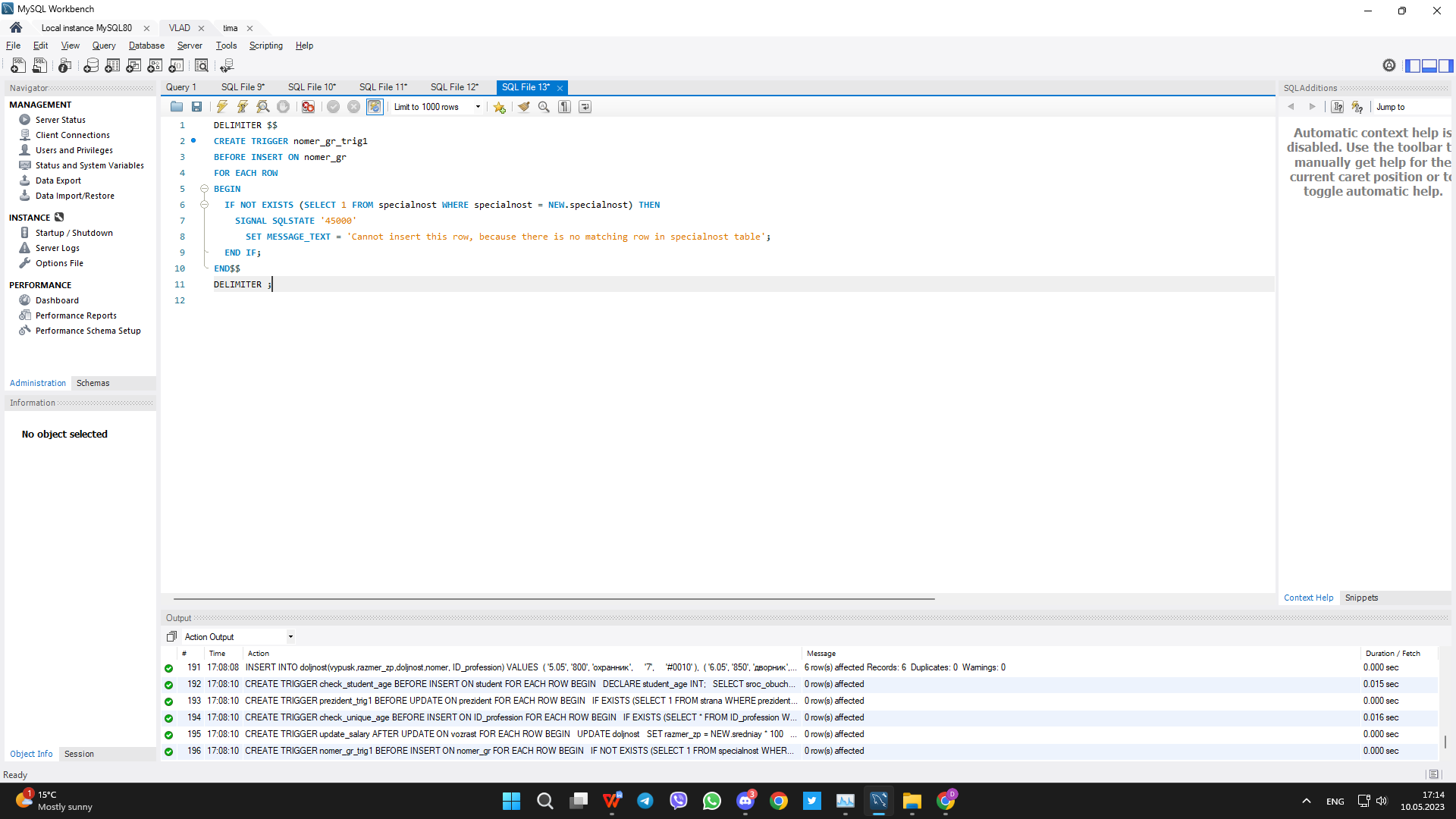
**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Возраст президента должен быть не менее 40 лет.';**

**END IF;**

**END$$**

**DELIMITER ;**





**Таким образом, с помощью создания процедур мы можем хранить запросы, логически их обрабатывать (с помощью if, for, while), а также вызывать функцию, когда нам нужно получить определённые значения (выходные параметры).**

В ходе этой работы мы научились создавать и выполнять процедуры и хранимые функции на языке запросов SQL/MySql.

В ходе этой работы мы изучили триггеры, изучили методы управления ими, а также научились создавать и использовать триггеры с использованием языка запросов SQL/MySql.