CREATE OR REPLACE FUNCTION check\_student\_email()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.email !~\* '^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$' THEN

RAISE EXCEPTION 'Некорректный email: %', NEW.email;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION create\_payment\_notification()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO Notification (Student\_ID, Text, Notification\_Date)

VALUES (NEW.Student\_ID, 'Ваша оплата за курс "' (SELECT Name FROM Course WHERE ID = NEW.Course\_ID) '" подтверждена.', NEW.Payment\_Date);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION check\_grade\_range()

RETURNS TRIGGER AS $$

BEGIN

-- Проверка, что оценка находится в диапазоне от 1 до 5

IF NEW.Grade < 1 OR NEW.Grade > 5 THEN

RAISE EXCEPTION 'Оценка должна быть в диапазоне от 1 до 5: %', NEW.Grade;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION delete\_student\_grades()

RETURNS TRIGGER AS $$

BEGIN

-- Удаление всех оценок учащегося

DELETE FROM Grade WHERE Student\_ID = OLD.ID;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION update\_payment\_amount()

RETURNS TRIGGER AS $$

BEGIN

-- Обновление суммы оплаты для всех учащихся, записанных на курс

UPDATE Payment

SET Amount = NEW.Price

WHERE Course\_ID = NEW.ID;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION create\_enrollment\_notification()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO Notification (Student\_ID, Text, Notification\_Date)

VALUES (NEW.Student\_ID, 'Вы записаны на курс "' (SELECT Name FROM Course WHERE ID = NEW.Course\_ID) '".', NEW.Payment\_Date);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION check\_payment\_date()

RETURNS TRIGGER AS $$

BEGIN

-- Проверка, что дата оплаты не является будущей датой

IF NEW.Payment\_Date > CURRENT\_DATE THEN

RAISE EXCEPTION 'Дата оплаты не может быть будущей датой: %', NEW.Payment\_Date;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION check\_notification\_date()

RETURNS TRIGGER AS $$

BEGIN

-- Проверка, что дата уведомления не является будущей датой

IF NEW.Notification\_Date > CURRENT\_DATE THEN

RAISE EXCEPTION 'Дата уведомления не может быть будущей датой: %', NEW.Notification\_Date;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION update\_payment\_amount()

RETURNS TRIGGER AS $$

BEGIN

-- Обновление суммы оплаты для всех учащихся, записанных на курс

UPDATE Payment

SET Amount = NEW.Price

WHERE Course\_ID = NEW.ID;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION check\_student\_existence()

RETURNS TRIGGER AS $$

BEGIN

-- Проверка, что учащийся существует в базе данных

IF NOT EXISTS (SELECT 1 FROM Student WHERE ID = NEW.Student\_ID) THEN

RAISE EXCEPTION 'Учащийся с ID % не существует в базе данных.', NEW.Student\_ID;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

/\*1. Триггер BEFORE INSERT для проверки корректности email учащегося

Задание: Создайте триггерную функцию и триггер, которые будут проверять, что

email учащегося соответствует формату email (например, содержит символ @ и

домен).\*/

CREATE TRIGGER before\_insert\_student\_email

BEFORE INSERT ON student

FOR EACH ROW

EXECUTE FUNCTION check\_student\_email();

INSERT INTO student (full\_name, email,phone) VALUES ('Ваня иванов', 'хз\_email',757578483);

/\*2. Триггер AFTER INSERT для автоматического создания уведомления об оплате

Задание: Создайте триггерную функцию и триггер, которые будут автоматически

создавать уведомление об оплате курса для учащегося.\*/

CREATE TRIGGER after\_insert\_payment

AFTER INSERT ON Payment

FOR EACH ROW

EXECUTE FUNCTION create\_payment\_notification();

INSERT INTO Payment (Student\_ID, Course\_ID, Amount, Payment\_Date) VALUES (1, 1, 5000, '2023-10-05');

SELECT \* FROM Notification;

/\*3. Триггер BEFORE UPDATE для проверки изменения оценки

Задание: Создайте триггерную функцию и триггер, которые будут проверять, что

оценка учащегося находится в диапазоне от 1 до 5.\*/

CREATE TRIGGER before\_update\_grade

BEFORE UPDATE ON Grade

FOR EACH ROW

EXECUTE FUNCTION check\_grade\_range();

UPDATE Grade SET Grade = 6 WHERE ID = 1;

/\*4. Триггер AFTER DELETE для автоматического удаления связанных оценок

Задание: Создайте триггерную функцию и триггер, которые будут автоматически

удалять все оценки учащегося, если учащийся удаляется из базы данных.\*/

CREATE TRIGGER after\_delete\_student

AFTER DELETE ON Student

FOR EACH ROW

EXECUTE FUNCTION delete\_student\_grades();

DELETE FROM Student WHERE ID = 5;

SELECT \* FROM Grade WHERE Student\_ID = 1;

/\*5. Триггер FOR EACH STATEMENT для автоматического обновления суммы

оплаты при изменении стоимости курса

Задание: Создайте триггерную функцию и триггер, которые будут автоматически

обновлять сумму оплаты для всех учащихся, если стоимость курса изменяется.\*/

CREATE TRIGGER after\_update\_course\_price

AFTER UPDATE OF Price ON Course

FOR EACH ROW

EXECUTE FUNCTION update\_payment\_amount();

UPDATE Course SET Price = 5500 WHERE ID = 1;

SELECT \* FROM Payment WHERE Course\_ID = 1;

/\*6. Триггер AFTER INSERT для автоматического создания уведомления о записи

на курс

Задание: Создайте триггерную функцию и триггер, которые будут автоматически

создавать уведомление о записи на курс для учащегося.\*/

CREATE TRIGGER after\_insert\_payments

AFTER INSERT ON Payment

FOR EACH ROW

EXECUTE FUNCTION create\_enrollment\_notification();

INSERT INTO Payment (Student\_ID, Course\_ID, Amount, Payment\_Date) VALUES (1, 1, 5000, '2023-10-05');

SELECT \* FROM Notification WHERE Student\_ID = 1;

/\*7. Триггер BEFORE UPDATE для проверки изменения даты оплаты

Задание: Создайте триггерную функцию и триггер, которые будут проверять, что

дата оплаты не может быть изменена на будущую дату.\*/

CREATE TRIGGER before\_update\_payment\_date

BEFORE UPDATE ON Payment

FOR EACH ROW

EXECUTE FUNCTION check\_payment\_date();

UPDATE Payment SET Payment\_Date = '2025-12-01' WHERE ID = 1;

UPDATE Payment SET Payment\_Date = '2023-10-05' WHERE ID = 1;

/\*8. Триггер BEFORE UPDATE для проверки изменения даты уведомления

Задание: Создайте триггерную функцию и триггер, которые будут проверять, что

дата уведомления не может быть изменена на будущую дату.\*/

CREATE TRIGGER before\_update\_notification\_date

BEFORE UPDATE ON Notification

FOR EACH ROW

EXECUTE FUNCTION check\_notification\_date();

UPDATE Notification SET Notification\_Date = '2025-12-01' WHERE ID = 1;

UPDATE Notification SET Notification\_Date = '2023-10-05' WHERE ID = 1;

/\*9. Триггер FOR EACH STATEMENT для автоматического обновления суммы

оплаты при изменении стоимости курса

Задание: Создайте триггерную функцию и триггер, которые будут автоматически

обновлять сумму оплаты для всех учащихся, если стоимость курса изменяется.\*/

CREATE TRIGGER after\_update\_course\_prices

AFTER UPDATE OF Price ON Course

FOR EACH ROW

EXECUTE FUNCTION update\_payment\_amount();

UPDATE Course SET Price = 5500 WHERE ID = 1;

SELECT \* FROM Payment WHERE Course\_ID = 1;

/\*10. Триггер BEFORE INSERT для проверки наличия учащегося

Задание: Создайте триггерную функцию и триггер, которые будут проверять, что

учащийся, который хочет получить оценку, существует в базе данных.\*/

CREATE TRIGGER before\_insert\_grade

BEFORE INSERT ON Grade

FOR EACH ROW

EXECUTE FUNCTION check\_student\_existence();

INSERT INTO Grade (Lesson\_ID, Student\_ID, Grade) VALUES (1, 999, 5);

INSERT INTO Grade (Lesson\_ID, Student\_ID, Grade) VALUES (1, 1, 5);