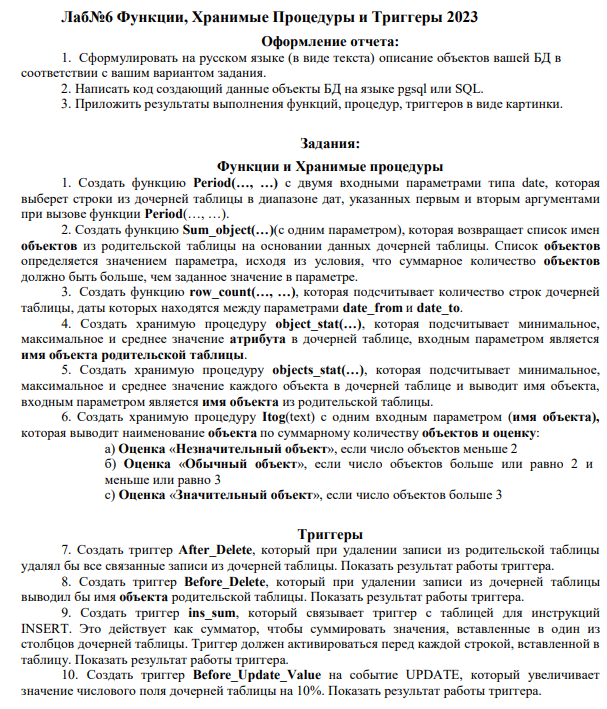
**Отчет по лабораторной работе 6.**

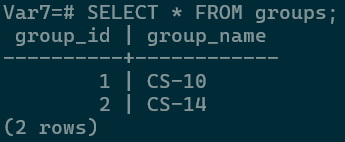
Выполнил: Золотухин Андрей Александрович КС-36

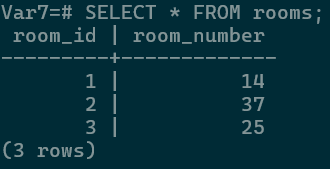
Преподаватель: Семёнов Геннадий Николаевич

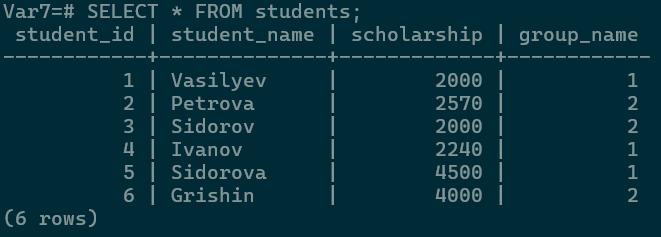
**ЗАДАНИЕ**

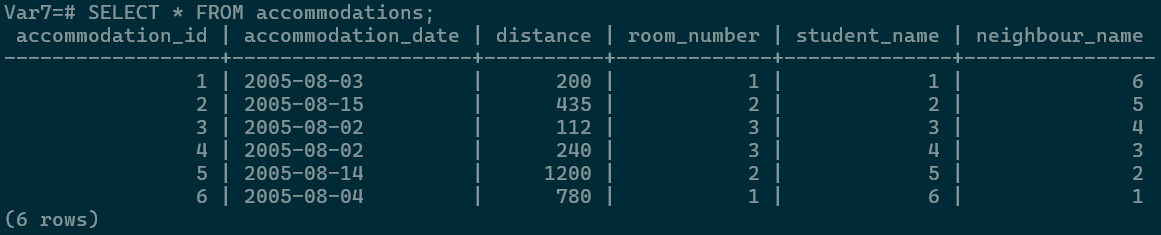
****

**Результат создания таблиц лабораторной работы 1**









**Выполнение задания**

**ФУНКЦИИ И ХРАНИМЫЕ ПРОЦЕДУРЫ**

1. **Создать функцию Period(..., …) с двумя входными параметрами типа date, которая выберет строки из дочерней таблицы в диапазоне дат, указанных первым и вторым аргументами при вызове функции Period(..., …):**

*Формирование списка размещений студентов в заданном диапазоне дат*:

CREATE OR REPLACE FUNCTION Period(start\_date DATE, end\_date DATE)

RETURNS TABLE(accommodation\_id INT, accommodation\_date DATE, distance INT, room\_number INT, student\_name INT, neighbour\_name INT) AS $$

BEGIN

RETURN QUERY

SELECT a.\*

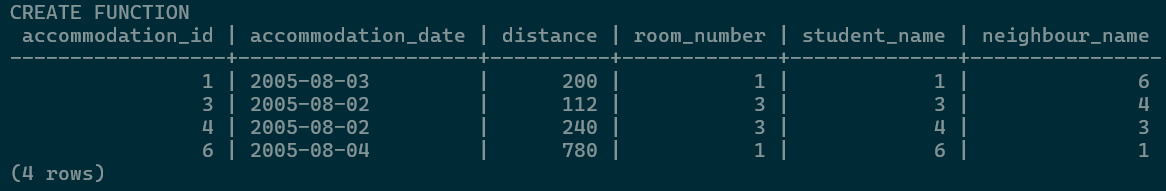
FROM public.accommodations a

WHERE a.accommodation\_date BETWEEN start\_date AND end\_date;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM Period('2005-08-01', '2005-08-10');



1. **Создать функцию Sum\_object(…)(с одним параметром), которая возвращает список имен объектов из родительской таблицы на основании данных дочерней таблицы. Список объектов определяется значением параметра, исходя из условия, что суммарное количество объектов должно быть больше, чем заданное значение в параметре:**

*Формирование списка групп, суммарная стипендия студентов в которых превышает заданное значение*:

CREATE OR REPLACE FUNCTION Sum\_object(min\_sum INTEGER)

RETURNS TABLE(group\_name VARCHAR) AS $$

BEGIN

RETURN QUERY

SELECT g.group\_name

FROM public.groups g

INNER JOIN public.students s ON g.group\_id = s.group\_name

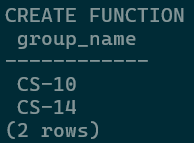
GROUP BY g.group\_name

HAVING SUM(s.scholarship) > min\_sum;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM Sum\_object(8000);



1. **Создать функцию row\_count(…, …), которая подсчитывает количество строк дочерней таблицы, даты которых находятся между параметрами date\_from и date\_to:**

*Формирование количества размещений студентов, которые произошли в заданный период*:

CREATE OR REPLACE FUNCTION row\_count(date\_from DATE, date\_to DATE)

RETURNS INT AS $$

DECLARE

count\_rows INT;

BEGIN

SELECT COUNT(\*)

INTO count\_rows

FROM public.accommodations

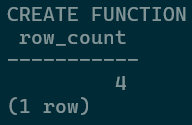
WHERE accommodation\_date BETWEEN date\_from AND date\_to;

RETURN count\_rows;

END;

$$ LANGUAGE plpgsql;

SELECT row\_count('2005-08-01', '2005-08-10');



1. **Создать хранимую процедуру object\_stat(…), которая подсчитывает минимальное, максимальное и среднее значение атрибута в дочерней таблице, входным параметром является имя объекта родительской таблицы:**

*Формирование статистики (минимальное, максимальное и среднее значение) для различных объектов (студенты, комнаты, размещения)*:

CREATE OR REPLACE PROCEDURE object\_stat(object\_name VARCHAR)

LANGUAGE plpgsql

AS $$

DECLARE

min\_value INTEGER;

max\_value INTEGER;

avg\_value FLOAT;

BEGIN

IF object\_name = 'students' THEN

SELECT MIN(scholarship), MAX(scholarship), AVG(scholarship)

INTO min\_value, max\_value, avg\_value

FROM students;

ELSIF object\_name = 'rooms' THEN

SELECT MIN(room\_number), MAX(room\_number), AVG(room\_number)

INTO min\_value, max\_value, avg\_value

FROM rooms;

ELSIF object\_name = 'accommodations' THEN

SELECT MIN(distance), MAX(distance), AVG(distance)

INTO min\_value, max\_value, avg\_value

FROM accommodations;

ELSE

RAISE EXCEPTION 'Unknown object name: %', object\_name;

END IF;

RAISE NOTICE 'Минимальное значение: %, Максимальное значение: %, Среднее значение: %', min\_value, max\_value, avg\_value;

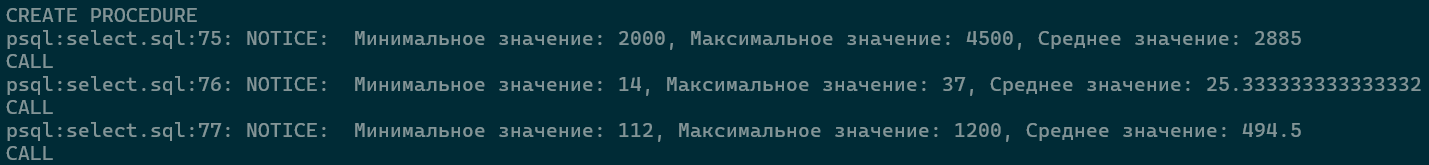
END;

$$;

CALL object\_stat('students');

CALL object\_stat('rooms');

CALL object\_stat('accommodations');



1. **Создать хранимую процедуру objects\_stat(…), которая подсчитывает минимальное, максимальное и среднее значение каждого объекта в дочерней таблице и выводит имя объекта, входным параметром является имя объекта из родительской таблицы:**

*Формирование статистики (минимальное, максимальное и среднее значение) для различных объектов (группы, комнаты)*:

CREATE OR REPLACE PROCEDURE objects\_stat(object\_name VARCHAR)

LANGUAGE plpgsql

AS $$

DECLARE

rec RECORD;

BEGIN

IF object\_name = 'groups' THEN

FOR rec IN

SELECT g.group\_name, MIN(s.scholarship) AS min\_scholarship,

MAX(s.scholarship) AS max\_scholarship,

AVG(s.scholarship) AS avg\_scholarship

FROM public.groups g

INNER JOIN public.students s ON g.group\_id = s.group\_name

GROUP BY g.group\_name

LOOP

RAISE NOTICE 'Группа: %, Минимальная стипендия: %, Максимальная стипендия: %, Средняя стипендия: %',

rec.group\_name, rec.min\_scholarship, rec.max\_scholarship, rec.avg\_scholarship;

END LOOP;

ELSIF object\_name = 'rooms' THEN

FOR rec IN

SELECT r.room\_number, MIN(a.distance) AS min\_distance,

MAX(a.distance) AS max\_distance,

AVG(a.distance) AS avg\_distance

FROM public.rooms r

INNER JOIN public.accommodations a ON r.room\_id = a.room\_number

GROUP BY r.room\_number

LOOP

RAISE NOTICE 'Комната: %, Минимальное расстояние: %, Максимальное расстояние: %, Среднее расстояние: %',

rec.room\_number, rec.min\_distance, rec.max\_distance, rec.avg\_distance;

END LOOP;

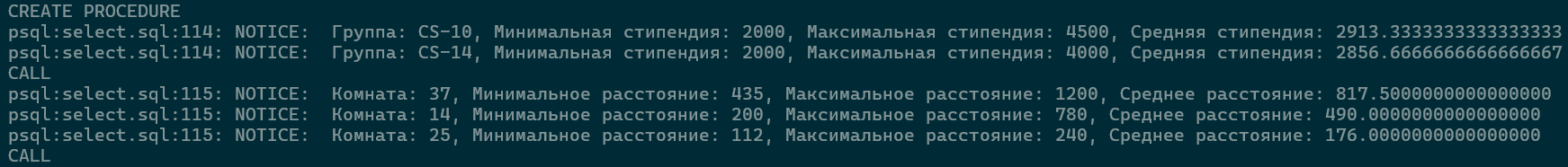
END IF;

END;

$$;

CALL objects\_stat('groups');

CALL objects\_stat('rooms');



1. **Создать хранимую процедуру Itog(text) с одним входным параметром (имя объекта), которая выводит наименование объекта по суммарному количеству объектов и оценку:** 
   1. **Оценка «Незначительный объект», если число объектов меньше 2**
   2. **Оценка «Обычный объект», если число объектов больше или равно 2 и меньше или равно 3**
   3. **Оценка «Значительный объект», если число объектов больше 3:**

*Формирование количества объектов и их оценки в зависимости от этого количества*:

CREATE OR REPLACE PROCEDURE Itog(IN object\_name TEXT)

LANGUAGE plpgsql

AS $$

DECLARE

object\_count INTEGER;

evaluation TEXT;

BEGIN

EXECUTE format('SELECT COUNT(\*) FROM %I', object\_name) INTO object\_count;

IF object\_count < 2 THEN

evaluation := 'Незначительный объект';

ELSIF object\_count >= 2 AND object\_count <= 3 THEN

evaluation := 'Обычный объект';

ELSE

evaluation := 'Значительный объект';

END IF;

RAISE NOTICE 'Объект: %, Количество: %, Оценка: %', object\_name, object\_count, evaluation;

END;

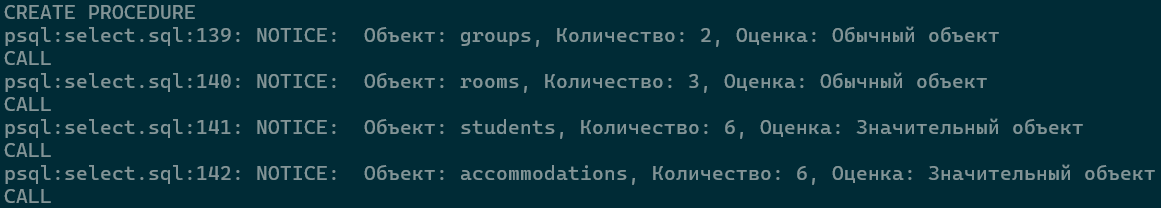
$$;

CALL Itog('groups');

CALL Itog('rooms');

CALL Itog('students');

CALL Itog('accommodations');



**ТРИГГЕРЫ**

1. **Создать триггер After\_Delete, который при удалении записи из родительской таблицы удалял бы все связанные записи из дочерней таблицы. Показать результат работы триггера:**

*Формирование удалений размещений студентов, связанных с удаляемой комнатой*:

Примечание: поскольку в 1-ой лаб. работе я поставил на ограничителе внешних ключей таблицы “размещений” флаг NO ACTION на ON DELETE, т.е. не производить никаких действий при этой операции, то я исправлю это значение на CASCADE, чтобы я мог удалять записи из родительской и дочерней таблиц одновременно.

ALTER TABLE public.accommodations

DROP CONSTRAINT room\_room\_number\_fkey;

ALTER TABLE public.accommodations

DROP CONSTRAINT student\_student\_name\_fkey;

ALTER TABLE public.accommodations

DROP CONSTRAINT neighbour\_neighbour\_name\_fkey;

ALTER TABLE public.accommodations

ADD CONSTRAINT room\_room\_number\_fkey FOREIGN KEY (room\_number)

REFERENCES public.rooms(room\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

ALTER TABLE public.accommodations

ADD CONSTRAINT student\_student\_name\_fkey FOREIGN KEY (student\_name)

REFERENCES public.students(student\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

ALTER TABLE public.accommodations

ADD CONSTRAINT neighbour\_neighbour\_name\_fkey FOREIGN KEY (neighbour\_name)

REFERENCES public.students(student\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE CASCADE;

CREATE OR REPLACE FUNCTION delete\_related\_accommodations()

RETURNS TRIGGER AS $$

BEGIN

DELETE FROM accommodations

WHERE room\_number = OLD.room\_id;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER after\_delete\_room

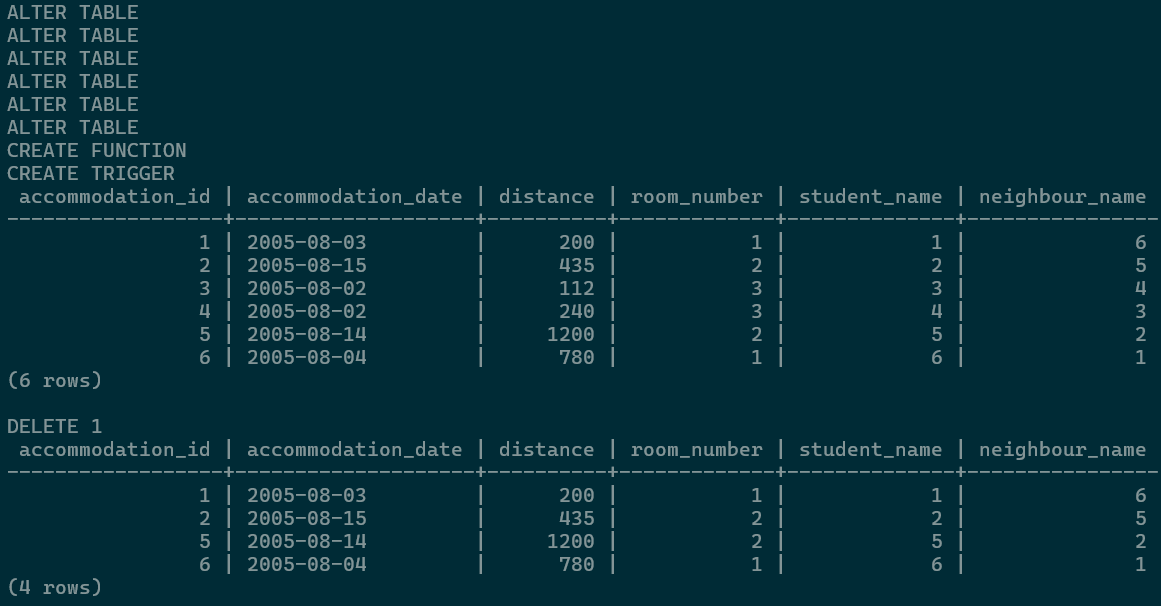
AFTER DELETE ON rooms

FOR EACH ROW EXECUTE FUNCTION delete\_related\_accommodations();

SELECT \* FROM accommodations;

DELETE FROM rooms WHERE room\_id = 3;

SELECT \* FROM accommodations;



1. **Создать триггер Before\_Delete, который при удалении записи из дочерней таблицы выводил бы имя объекта родительской таблицы. Показать результат работы триггера:**

*Формирование уведомлений перед удалением размещения студента*:

CREATE OR REPLACE FUNCTION notify\_before\_delete\_accommodation()

RETURNS TRIGGER AS $$

BEGIN

RAISE NOTICE 'Удаляется размещение студента: %', OLD.accommodation\_id;

IF EXISTS (SELECT 1 FROM rooms WHERE room\_id = OLD.room\_number) THEN

RAISE NOTICE 'Удаляется комната с номером: %', OLD.room\_number;

END IF;

IF EXISTS (SELECT 1 FROM students WHERE student\_id = OLD.student\_name) THEN

RAISE NOTICE 'Удаляется студент с ID: %', OLD.student\_name;

END IF;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER before\_delete\_accommodation

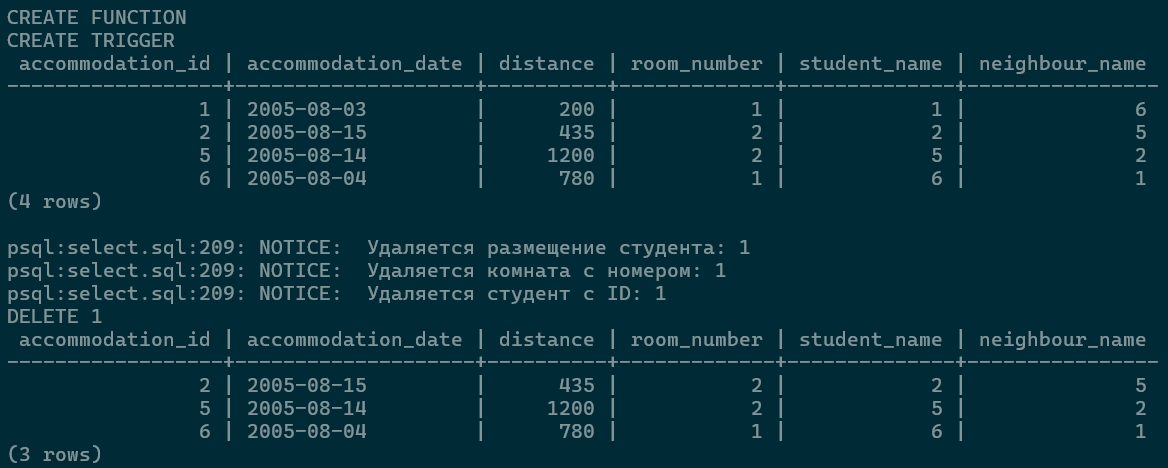
BEFORE DELETE ON accommodations

FOR EACH ROW EXECUTE FUNCTION notify\_before\_delete\_accommodation();

SELECT \* FROM accommodations;

DELETE FROM accommodations WHERE accommodation\_id = 1;

SELECT \* FROM accommodations;



1. **Создать триггер ins\_sum, который связывает триггер с таблицей для инструкций INSERT. Это действует как сумматор, чтобы суммировать значения, вставленные в один из столбцов дочерней таблицы. Триггер должен активироваться перед каждой строкой, вставленной в таблицу. Показать результат работы триггера:**

*Формирование суммы значений дистанции от общежития при добавлении новых размещений*:

CREATE OR REPLACE FUNCTION sum\_distance()

RETURNS TRIGGER AS $$

BEGIN

NEW.distance := NEW.distance + (SELECT COALESCE(SUM(distance), 0)

FROM accommodations);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER ins\_sum

BEFORE INSERT ON accommodations

FOR EACH ROW

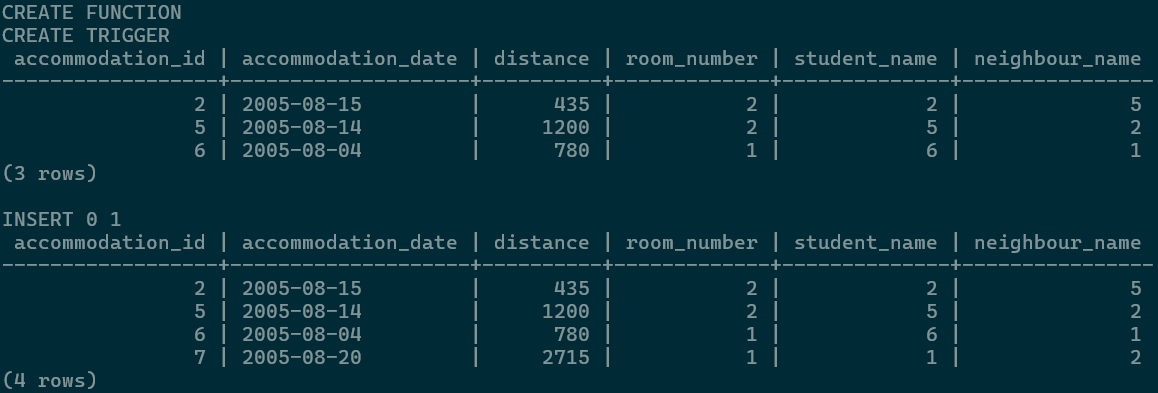
EXECUTE FUNCTION sum\_distance();

SELECT \* FROM accommodations;

INSERT INTO accommodations (accommodation\_date, distance, room\_number, student\_name, neighbour\_name) VALUES

('2005-08-20', 300, 1, 1, 2);

SELECT \* FROM accommodations;



1. **Создать триггер Before\_Update\_Value на событие UPDATE, который увеличивает значение числового поля дочерней таблицы на 10%. Показать результат работы триггера:**

*Формирование увеличения стипендии студентов на 10% перед обновлением*:

CREATE OR REPLACE FUNCTION before\_update\_value()

RETURNS TRIGGER AS $$

BEGIN

NEW.scholarship := NEW.scholarship \* (1.0 + (10.0 / 100.0));

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER Before\_Update\_Value

BEFORE UPDATE ON students

FOR EACH ROW

EXECUTE FUNCTION before\_update\_value();

UPDATE students SET scholarship = 2000 WHERE student\_id = 1;

SELECT \* FROM students WHERE student\_id = 1;

