ДЗ 2: 1. Определил первичный как Id, не пустой, автоинкремент. Внешние ключи:
FOREIGN KEY("teacher_id") REFERENCES "teachers"("id"),
FOREIGN KEY("stream_id") REFERENCES "streams"("id"),
2. Про ограничение уникальности — согласен. Отчего-то я упустил из виду это задание.

ДЗ 4.2: Добавил в таблицу ещё данных. Результат тот же.
Обратный запрос даёт вполне предсказуемый результат. Я озадачен.

ДЗ 6.3: Справедливое замечание. Обдумал, понял.

8.1 SELECT DISTINCT
name AS course_name,
SUM(students_amount) over(Partition by course_id) AS total_students
FROM streams LEFT JOIN courses
on course_id = courses.id;

8.2 SELECT DISTINCT
teachers.id,
surname,
name,
avg(performance) OVER (PARTITION BY teachers.id) as AVG
from teachers LEFT JOIN academic_performance
ON teachers.id = teacher_id;

8.3 Без индексов запрос выполняется — 18 мс.

CREATE INDEX flow_number_idx on streams(flow_number); — 15 мс

CREATE INDEX surname_idx on teachers(surname);
CREATE INDEX name_idx on teachers(name);
CREATE INDEX performance_idx on academic_performance(performance); — 6мс.

CREATE INDEX flow_number_idx on streams(flow_number);
CREATE INDEX surname_idx on teachers(surname);
CREATE INDEX name_idx on teachers(name);
CREATE INDEX performance_idx on academic_performance(performance); — 3 мс.

8.4 Скриншоты. SQLiteStudio не получилось быстро собрать из исходников собрать. С установочным пакетом возникли сложности, возможно зависимости не все удовлетворены. Подумал, что мой вариант ничуть не хуже.