SQLite – работа в CLI mode

Задача: В таблице streams переименовать столбец даты начала обучения в started_at.

```
sqlite> ALTER TABLE streams RENAME COLUMN start_date TO started_at;
```

Задача: В таблице streams добавить столбец даты завершения обучения в finished_at.

```
sqlite> ALTER TABLE streams ADD COLUMN finished_at TEXT;
```

Задача: Привести данные в полное соответствие с таблицами 1-4 практического задания в методичке

```
sqlite> INSERT INTO teachers (surname, name, email)  VALUES
  ('Николай', 'Савельев', 'saveliev.n@mai.ru'),
  ('Наталья', 'Петрова', 'petrova.n@yandex.ru'),
  ('Елена', 'Малышева', 'malisheva.e@google.com');

sqlite> INSERT INTO courses (name)  VALUES
  ('Базы данных'),
  ('Основы Python'),
  ('Linux. Рабочая станция');

sqlite> ALTER TABLE streams RENAME COLUMN students TO students_amount;

sqlite> INSERT INTO streams (course_id, number, started_at, students_amount)  VALUES
  (3, 165, '18.08.2020', 34),
  (2, 178, '02.10.2020', 37),
  (1, 203, '12.11.2020', 35),
  (1, 210, '03.12.2020', 41);

sqlite> ALTER TABLE grades RENAME COLUMN grade TO performance;

sqlite> INSERT INTO grades (teacher_id, stream_id, performance)  VALUES
  (3, 1, 4.7),
```

(2, 2, 4.9),

(1, 3, 4.8),

(1, 4, 4.9);

Задача: Преобразовать дату начала потока к виду год-месяц-день.

```
UPDATE streams SET started_at = SUBSTR(started_at, 7, 4) || '-' || SUBSTR(started_at, 4, 2) || '-' || SUBSTR(started_at, 1, 2);
```

Задача: Получить идентификатор и номер потока, который запланирован на самую позднюю дату.

```
SELECT id, number FROM streams ORDER BY started_at DESC LIMIT 1;
```

Задача: Показать уникальные значения года дат начала потоков обучения

```
SELECT DISTINCT(SELECT SUBSTR(started_at, 1, 4) FROM streams);
```

Задача: Найти количество преподавателей в базе данных. Вывести искомое значение в столбец с именем total_teachers.

```
//SELECT id, MAX(id) AS 'total_teachers' FROM teachers;

SELECT COUNT(*) AS 'total_teachers' FROM teachers;
```

Задача: Показать даты начала двух последних по времени потоков.

```
SELECT * FROM streams ORDER BY started_at DESC LIMIT 2;
```

Задача: Найти среднюю успеваемости учеников по потоку преподавателя с идентификатором равным 1.

```
SELECT teacher_id, AVG(performance) FROM grades GROUP BY teacher_id HAVING teacher_id=1;
```

Задача: Найти потоки, количество учеников в которых больше или равно 40. В отчёт вывести номер потока, название курса и количество учеников.

```
SELECT
        number,
        (SELECT name FROM courses WHERE id = course_id) AS course_name,
        students_amount
FROM streams WHERE students_amount >= 40;
```

Задача: Найти два потока с самыми низкими оценками за успеваемость. В отчёт вывести номер потока, название курса, фамилию и имя преподавателя (одним столбцом), оценку успеваемости.

```
SELECT
        (SELECT number FROM streams WHERE id = stream_id) AS stream_number,
        (SELECT name FROM courses WHERE id =
           (SELECT course_id FROM streams WHERE id = stream_id)
         ) AS course_name,
        (SELECT name || ' ' || surname  FROM teachers WHERE id = teacher_id) AS teacher_name,
        performance
FROM grades ORDER BY performance ASC LIMIT 2;
```

Задача: Найти среднюю успеваемость всех потоков преподавателя Николая Савельева. В отчёт вывести идентификатор преподавателя и среднюю оценку по потокам.

```
SELECT
        teacher_id, AVG(performance)
FROM grades WHERE teacher_id = (SELECT id FROM teachers WHERE name = 'Савельев' AND surname
= 'Николай');
```

Задача: Найти потоки преподавателя Натальи Петровой, а также потоки, по которым успеваемость ниже 4.8. В отчёт вывести идентификатор потока, фамилию и имя преподавателя.

```
SELECT
        stream_id,
        (SELECT name || ' ' || surname  FROM teachers WHERE id = teacher_id) AS teacher_name
FROM grades WHERE teacher_id = (SELECT id FROM teachers WHERE name = 'Петрова' AND surname =
'Наталья')
UNION
SELECT
        stream_id,
        (SELECT name || ' ' || surname  FROM teachers WHERE id = teacher_id) AS teacher_name
FROM grades WHERE performance < 4.8;
```

Задача: Показать информацию по потокам. В отчет вывести номер потока, название курса и дату начала занятий.

```
SELECT number, name, started_at
FROM streams JOIN courses
ON streams.course_id = courses.id;
```

Задача: Найти общее количество учеников для каждого курса. В отчет вывести название курса и количество учеников по всем потокам курса.

```
SELECT
        name,
        SUM(students_amount)
FROM streams JOIN courses
ON streams.course_id = courses.id
GROUP BY name ;
```

Задача: Найти среднюю оценку по всем потокам для всех учителей. В отчет вывести идентификатор, фамилию и имя учителя, среднюю оценку по всем проведенным потокам. Учителя, у которых не было потоков, также должны попасть в выборку.

```
SELECT
teachers.id,
teachers.surname,
teachers.name,
AVG(grades.performance)
FROM teachers
        LEFT JOIN grades
                ON teachers.id = grades.teacher_id
        LEFT JOIN streams
                ON grades.stream_id = streams.id
GROUP BY teachers.id;
```

Задача: Создайте представление, которое для каждого курса выводит название, номер последнего потока, дату начала обучения последнего потока и среднюю успеваемость курса по всем потокам.

```
CREATE VIEW task1 AS SELECT
number,
name,
MAX(started_at),
AVG(performance)
FROM streams
        JOIN courses
        ON streams.course_id = courses.id
        JOIN grades
        ON streams.id = grades.stream_id
GROUP BY name;
```

Задача: Удалите из базы данных всю информацию, которая относится к преподавателю с идентификатором, равным 3. Используйте транзакцию.

```
BEGIN TRANSACTION;
        DELETE FROM teachers WHERE id = 3;
        DELETE FROM grades WHERE teacher_id = 3;
COMMIT;
```

Задача: Создайте триггер для таблицы успеваемости, который проверяет значение успеваемости на соответствие диапазону чисел от 0 до 5 включительно.

```
CREATE TRIGGER check_performance_format BEFORE INSERT
ON grades
BEGIN
 SELECT CASE
 WHEN
  (NEW.performance NOT BETWEEN 0 AND 5)
 THEN
```

```
   RAISE(ABORT, 'Wrong value for performance!')
  END;
END;
```

Задача: Создайте триггер для таблицы потоков, который проверяет, что дата начала потока больше текущей даты, а номер потока имеет наибольшее значение среди существующих номеров. При невыполнении условий необходимо вызвать ошибку с информативным сообщением.

```
CREATE TRIGGER check_data_and_number BEFORE INSERT
ON streams
BEGIN
  SELECT CASE
WHEN
    (CAST(strftime('%s', NEW.started_at)  AS  integer)
<= CAST(strftime('%s', date('now'))  AS  integer))
   OR (NEW.number <= (SELECT MAX(number) FROM streams))
   THEN
   RAISE(ABORT, 'Wrong value for date or number!')
  END;
END;
```