

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.20

Основы работы с SQLite3.

Выполнил студент группы

ИВТ-б-о-22-1

Пушкин Н.С. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Ставрополь 2023

Цель работы: исследовать базовые возможности системы управления базами данных SQLite3.

Порядок выполнения работы:

Задание 1. Выполнение команды.

```
sqlite> create table customer(name);
sqlite> select *
...> from customer;
sqlite> .schema customer
CREATE TABLE customer(name);
sqlite> |
```

Рисунок 1. Создание таблицы customer со столбцом name

Задание 2. Решите задачу: с помощью команды. help найдите в песочнице команду, которая отвечает за вывод времени выполнения запроса.

```
sqlite> select count(*) from city;

count(*)
1117

sqlite> .timer on
sqlite> select count(*) from city;

count(*)
1117

Run Time: real 0.001 user 0.000000 sys 0.000000
sqlite> |
```

Рисунок 2. Время выполнения запроса

Задание 3. Решите задачу: загрузите файл city.csv. Затем выполните такой запрос:

```
sqlite> select max(length(city)) from city
...> ;
```

max(length(city))
25

```
Run Time: real 0.001 user 0.000000 sys 0.000000
sqlite> |
```

Рисунок 3. Вывод запроса

Задание 4. Решите задачу: загрузите файл `city.csv` в песочнице с помощью команды `import`, но без использования опции `--csv`. Эта опция появилась только в недавней версии SQLite (3.32, май 2020), так что полезно знать способ, подходящий для старых версий. Вам поможет команда `help import`. Всего должно получиться две команды:

```
sqlite> .mode csv
sqlite> .import city.csv city
```

Рисунок 4. Добавления данных без использования опции csv

Задание 5. Решите задачу: напишите в песочнице запрос, который посчитает количество городов для каждого часового пояса в Сибирском и Приволжском федеральных округах. Выведите столбцы `timezone` и `city_count`, отсортируйте по значению часового пояса:

```
sqlite> select timezone,
...> count(*) city_count
...> from city
...> where federal_district
...> in ('Сибирский', 'Приволжский')
...> group by 1
...> order by 1 asc;
```

timezone	city_count
UTC+3	101
UTC+4	41
UTC+5	58
UTC+6	6
UTC+7	86
UTC+8	22

Рисунок 5. Результат запроса

Задание 6. Решите задачу: напишите в песочнице запрос, который найдет три ближайших к Самаре города, не считая саму Самару.

Укажите в ответе названия этих трех городов через запятую в порядке удаления от Самары.

```
sqlite> with geo_las as (select geo_lat as geo_las from city where city = 'Самара'),  
...> geo_los as (select geo_lon as geo_los from city where city = 'Самара'),  
...> geo_lam as (select geo_lat as geo_lam, city from city),  
...> geo_lou as (select geo_lon as geo_lou from city)  
...> select sqrt((power((geo_las - geo_lam),2) + power((geo_los - geo_lou),2)))  
...> as distance, city from (geo_las ,geo_los ,geo_lam, geo_lou )  
...> where city != 'Самара'  
...> order by distance asc limit 3;
```

Рисунок 6. Запрос

```
0.001052999999999886 | Заречный  
0.00948430000000004 | Каменка  
0.01199310000000051 | Елизово
```

Рисунок 7. Результат запроса

Задание 7. Решите задачу: напишите в песочнице запрос, который посчитает количество городов в каждом часовом поясе. Отсортируйте по количеству городов по убыванию.

```
sqlite> select timezone,  
...> count(*) city_count  
...> from city  
...> group by 1  
...> order by 2 desc;
```

timezone	city_count
UTC+3	660
UTC+5	173
UTC+7	86
UTC+4	66
UTC+9	31
UTC+8	28
UTC+2	22
UTC+10	22
UTC+11	17
UTC+6	6
UTC+12	6

```
sqlite>
```

Рисунок 8. Результат запроса

Индивидуальное задание:

```
sqlite> select count(*) from happy;
```

count(*)
312

Рисунок 9. Первый запрос

```
sqlite> select Max(Score) from happy;
```

Max(Score)
7.769

```
sqlite> select "Country or region", Year FROM happy WHERE Score=7.769;
```

Country or region	Year
Finland	2019

```
sqlite> |
```

Рисунок 10. Второй запрос (Страна и год с самым высоким индексом счастья)

```
sqlite> select "Country or region", Year FROM happy WHERE Score=6.774;
```

Country or region	Year
South Sudan	2019

```
sqlite> |
```

Рисунок 11. (Страна и год с самым низким индексом счастья)

```
sqlite> select "Country or region", Year, Score FROM happy;
```

Country or region	Year	Score
United Arab Emirates	2018	6.774

Рисунок 12. Четвёртый запрос (Страна с самым высоким ВВП на душу населения и её уровень счастья)

```
sqlite> select "Country or region", year, Score from happy where year in ('2018') order by 3 DESC limit 10;
```

Country or region	Year	Score
Finland	2018	7.632
Norway	2018	7.594
Denmark	2018	7.555
Iceland	2018	7.495
Switzerland	2018	7.487
Netherlands	2018	7.441
Canada	2018	7.328
New Zealand	2018	7.324
Sweden	2018	7.314
Australia	2018	7.272

Рисунок 13. Пятый запрос (Топ 10 стран с самым высоким уровнем счастья в 2018 году)

Вывод: в ходе работы были исследованы базовые возможности системы управления базами данных SQLite3.