

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-
СКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.20

Тема: «Основы
работы с SQLite3»

Выполнил студент группы

ИВТ-б-о-21-1

Богадунов В.И. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Ставрополь 2022

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

Ход работы:

1. Создал репозиторий в GitHub, дополнил правила в .gitignore для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.

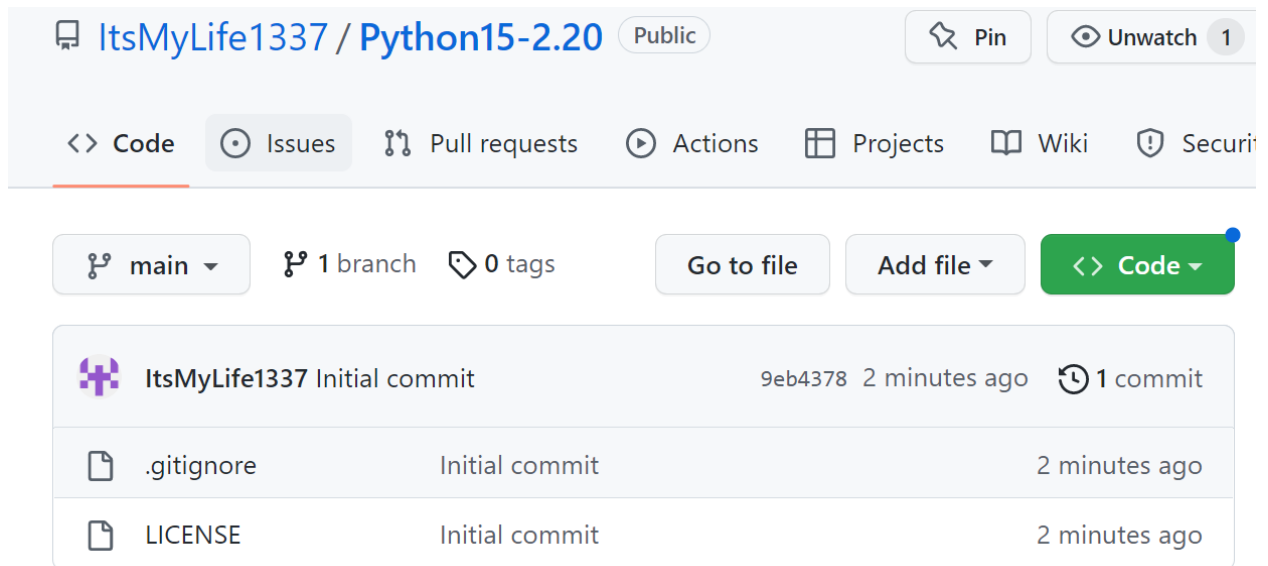


Рисунок 1.1 – Созданный репозиторий

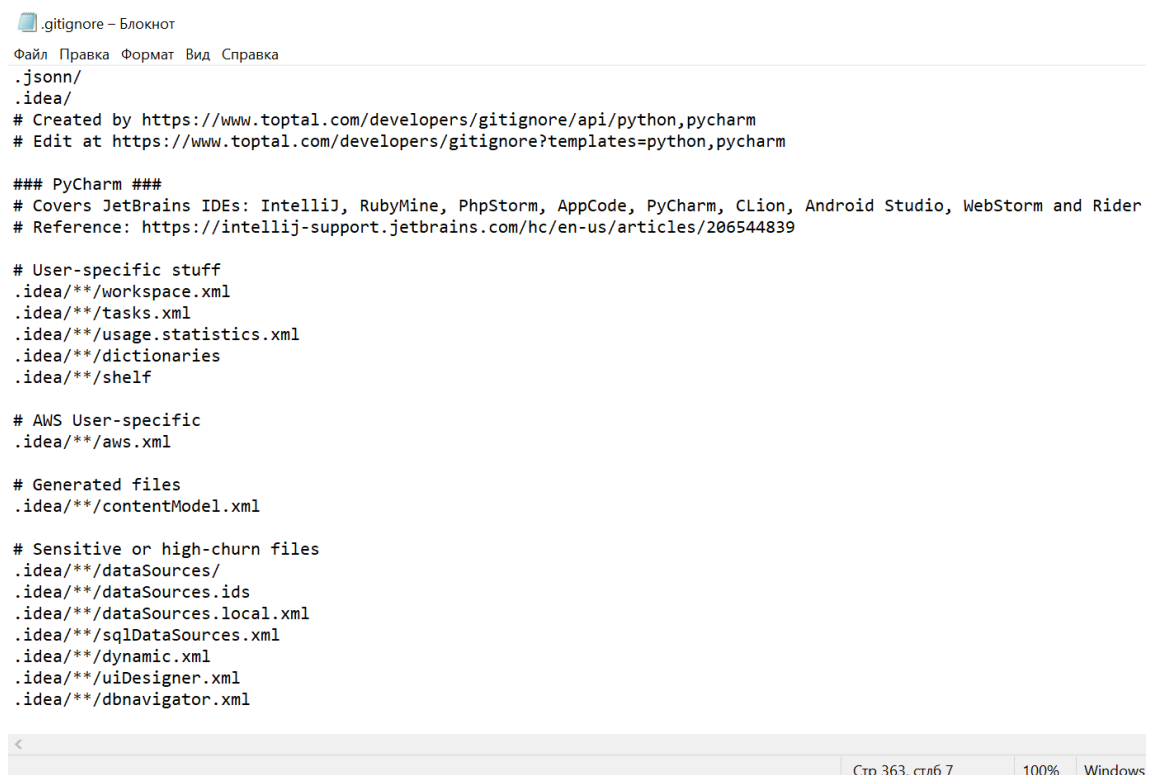


Рисунок 1.2 – Дополнил правила в .gitignore

```

c:\Users\Admin\Desktop\git\Python15-2.20>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Admin/Desktop/git/Python15-2.20/.git/hooks]

c:\Users\Admin\Desktop\git\Python15-2.20>

```

Рисунок 1.3 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Изучил работу примеров лабораторной работы в «Песочнице» и приступил к выполнению заданий.

Задание №1(7). Выполнение команд. Вот что здесь происходит:

```

SQLite version 3.38.0 2022-02-22 18:58:40
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> create table customer(name);
sqlite> select *
...> from customer;
sqlite> .schema customer
CREATE TABLE customer(name);

```

Рисунок 2 – Создание таблицы customer со столбцом (name)

Что вернула команда .schema? Ответ: данная команда показала какие столбцы есть в таблице.

Задание №2(8). Решите задачу: с помощью команды .help найдите в песочнице команду, которая отвечает за вывод времени выполнения запроса. Если ее включить, в результатах запроса добавится строка:

Например: Run Time: real XXX user XXX sys XXX

```
.system CMD ARGS...      Run CMD ARGS... in a system shell
.tables ?TABLE?          List names of tables matching LIKE pattern TA
.testcase NAME            Begin redirecting output to 'testcase-out.txt
.testctrl CMD ...         Run various sqlite3_test_control() operations
.timeout MS               Try opening locked tables for MS milliseconds
.timer on|off             Turn SQL timer on or off
.trace ?OPTIONS?          Output each SQL statement as it is run
.vfsinfo ?AUX?            Information about the top-level VFS
.vfslist                  List all available VFSes
.vfsname ?AUX?            Print the name of the VFS stack
.width NUM1 NUM2 ...      Set minimum column widths for columnar output
sqlite>
```

Рисунок 3 – С помощью команды .help нашёл команду, которая отвечает за время выполнения запроса

```
sqlite> .timer on
```

Рисунок 4 – Включаем таймер (чтобы увидеть время выполнения запросов)

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

count (*)
1117

```
Run Time: real 0.000 user 0.000255 sys 0.000000
sqlite>
```

Рисунок 5 – Вводим необходимый запрос и получаем время его выполнения

Задание №3(9). Решите задачу: загрузите файл city.csv в песочнице. Затем выполните такой запрос: `select max(length(city)) from city;`.

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

max(length(city))
25

```
sqlite>
```

Рисунок 6 – Вывод запроса

Какое число он вернул? **Ответ: 25**

Задание №4(10). 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.

```
Last login: Sun Dec 25 14:34:06 2022 from 127.0.0.1
SQLite version 3.38.0 2022-02-22 18:58:40
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .help import
.import FILE TABLE      Import data from FILE into TABLE
Options:
  --ascii                Use \037 and \036 as column and row separators
  --csv                  Use , and \n as column and row separators
  --skip N               Skip the first N rows of input
  --schema S             Target table to be S.TABLE
  -v                     "Verbose" - increase auxiliary output
Notes:
  * If TABLE does not exist, it is created. The first row of input
    determines the column names.
  * If neither --csv or --ascii are used, the input mode is derived
    from the ".mode" output mode
  * If FILE begins with "|" then it is a command that generates the
    input text.
sqlite> .mode csv
sqlite> .import city.csv city
sqlite> 
```

Рисунок 7 – Добавление данных без использования опции `--csv`

Задание №5(11). 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.

Запрос:

Select

`timezone, count(city) as city_count from city where federal_district = 'Приволжский' or federal_district = 'Сибирский' group by timezone order by timezone ASC;`

```
sqlite> select
...> timezone, count(city) as city_count
...> from city where federal_district = 'Приволжский' or federal_district = 'Сибирский'
...> group by timezone
...> order by timezone ASC;
```

Рисунок 8 – Написанный запрос

```
UTC+3 | 101
UTC+4 | 41
UTC+5 | 58
UTC+6 | 6
UTC+7 | 86
UTC+8 | 22
sqlite> 
```

Рисунок 9 – Ответ

Задание №6(12). 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.

Запрос:

```
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;
```

```
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 10 – Написанный запрос

```
0.00105299999999886 | Заречный
0.0094843000000004 | Каменка
0.01199310000000051 | Елизово
```

Рисунок 11 – Результат выполнения запроса

Задание №7(13). Решите задачу: напишите в песочнице запрос, который посчитает количество городов в каждом часовом поясе. Отсортируйте по количеству городов по убыванию.

А теперь выполните этот же запрос, но так, чтобы результат был

- в формате CSV,
- с разделителем «pipe» |

Как выглядит четвертая строка результата?

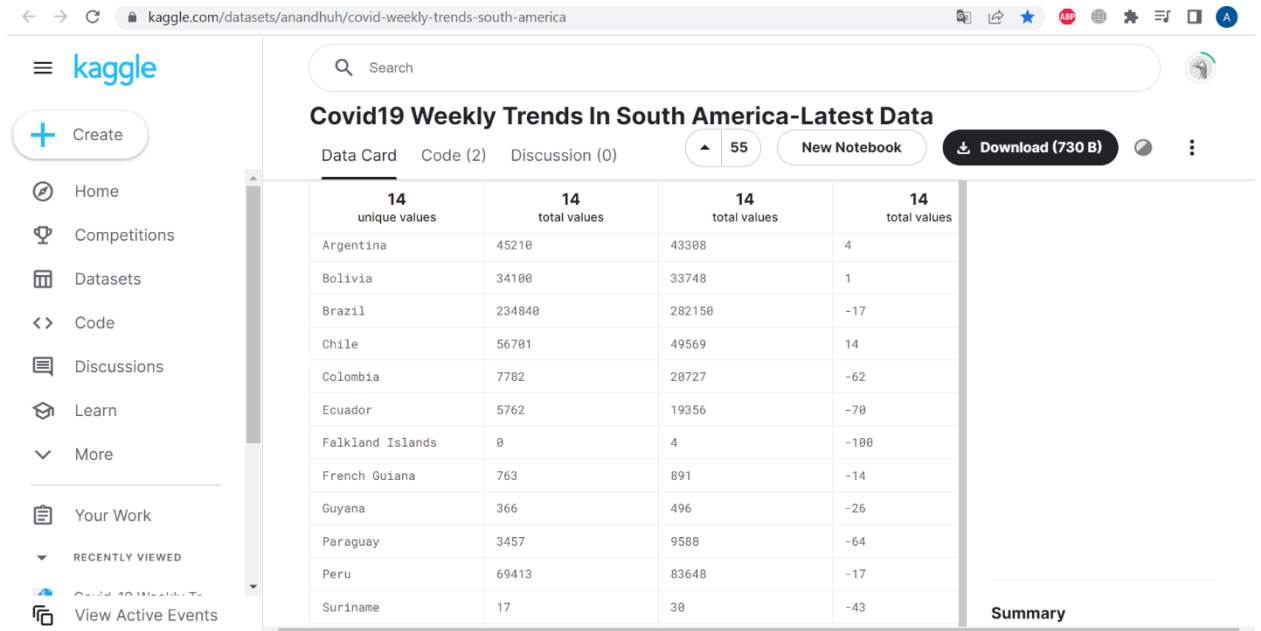
```
sqlite> select timezone,  
...> count (*) city_count  
...> from city  
...> group by 1  
...> order by 2 desc;  
UTC+3,660  
UTC+5,173  
UTC+7,86  
UTC+4,66  
UTC+9,31  
UTC+8,28  
UTC+2,22  
UTC+10,22  
UTC+11,17  
UTC+6,6  
UTC+12,6  
sqlite> |
```

Рисунок 12 – Результат выполнения запроса в формате csv с заголовками

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

Рисунок 13 – Результат выполнения запроса с «pipe» разделителем

Индивидуальное задание. Загрузите в SQLite выбранный Вами датасет в формате CSV (датасет можно найти на сайте Kaggle). Сформируйте более пяти запросов к таблицам БД. Выгрузите результат выполнения запросов в форматы CSV и JSON. Выбранный датасет.



Country	CasesLast7d	DeathsLast7d	RecoveriesLast7d
Argentina	45210	43308	4
Bolivia	34100	33748	1
Brazil	234840	282150	-17
Chile	56701	49569	14
Colombia	7782	20727	-62
Ecuador	5762	19356	-70
Falkland Islands	0	4	-100
French Guiana	763	891	-14
Guyana	366	496	-26
Paraguay	3457	9588	-64
Peru	69413	83648	-17
Suriname	17	30	-43

Рисунок 14 – Выбранный датасет с Kaggle

```
sqlite> .mode box
sqlite> .headers on
sqlite> select Country, CasesLast7d from covid where CasesLast7d between 2500 and 3000;
```

Country	CasesLast7d
Colombia	2512

```
sqlite>
```

Рисунок 15 – 1 Запрос зафиксированные за последние 7 дней случаи в промежутке между 2500 и 3000 человек, среди стран

```
sqlite> select Country, DeathsLast7d from covid where DeathsLast7d == 0 limit 2;
```

Country	DeathsLast7d
Falkland Islands	0
French Guiana	0

```
sqlite>
```

Рисунок 16 – 2 Запрос, выбираю страна где 0 смертность за последние 7 дней (выбрал 2)


```
sqlite> select Country, Population from covid order by Population asc limit 10;
```

Country	Population
Bolivia	11948668
Ecuador	18106806
Chile	19400316
Brazil	215175647
Venezuela	28296430
French Guiana	312076
Peru	33766958
Uruguay	3494576
Falkland Islands	3655
Argentina	45913837

```
sqlite>
```

Рисунок 17 – 3 Запрос

```
sqlite> select Country, CasesPreceding7d, Population from covid where CasesPreceding7d between 4000 and 4999 order by Population desc limit 3;
```

Country	CasesPreceding7d	Population
Colombia	4312	51821865
Peru	4901	33766958
Ecuador	4629	18106806

```
sqlite>
```

Рисунок 18 – 4 Запрос зафиксированные случаи за предыдущие 7 дней между 4000 и 4999 среди стран от общей численности

```
sqlite> select Country, WeeklyCasePerChange, WeeklyDeathPerChange from covid where WeeklyCasePerChange <= 0 and WeeklyDeathPerChange != 0 limit 8;
```

Country	WeeklyCasePerChange	WeeklyDeathPerChange
Argentina	-50	35
Bolivia	-31	-44
Brazil	-19	-22
Chile	-38	-33
Colombia	-42	-22
Ecuador	-29	-87
Guyana	-20	-100
Paraguay	-34	-42

Рисунок 19 – 5 Запрос еженедельные %-ы изменений (взял <=0 случаи заболевания) и !=0 % смертности

```
sqlite> select Country, CasesLast7d1MPop, CasesPreceding7d, Population from covid where CasesLast7d1Mpop between 0 and 250 and CasesPreceding7d != 0 order by Population desc;
```

Country	CasesLast7d1MPop	CasesPreceding7d	Population
Paraguay	111	1216	7284217
Peru	119	4901	33766958
French Guiana	1035	251	312076
Venezuela	16	1260	28296430
Ecuador	181	4629	18106806
Bolivia	133	2312	11948668

```
sqlite>
```

Рисунок 20 – Запрос 6, случаи заражения на 1 миллион (между 0 и 250) и зафиксированные за 7 дней













 request1	25.12.2022 16:40	Microsoft Excel Com...	1 КБ
 request1	25.12.2022 18:18	JSON File	1 КБ
 request2	25.12.2022 18:00	Microsoft Excel Com...	1 КБ
 request2	25.12.2022 18:18	JSON File	1 КБ
 request3	25.12.2022 17:59	Microsoft Excel Com...	1 КБ
 request3	25.12.2022 18:18	JSON File	1 КБ
 request4	25.12.2022 18:01	Microsoft Excel Com...	1 КБ
 request4	25.12.2022 18:18	JSON File	1 КБ
 request5	25.12.2022 18:05	Microsoft Excel Com...	1 КБ
 request5	25.12.2022 18:18	JSON File	1 КБ
 request6	25.12.2022 18:08	Microsoft Excel Com...	2 КБ
 request6	25.12.2022 18:16	JSON File	2 КБ

Рисунок 21 – Все 6 результатов выгруженные в формате .JSON и .csv

Вывод: в результате выполнения лабораторной работы были исследованы на практике базовые возможности системы управления базами данных SQLite3.

Ответы на контрольные вопросы:

1. Каково назначение реляционных баз данных и СУБД?

Главная функция СУБД – это управление данными (которые могут быть как во внешней, так и в оперативной памяти). СУБД обязательно поддерживает языки баз данных, а также отвечает за копирование и восстановление информации после каких-либо сбоев.

2. Каково назначение языка SQL?

Язык SQL предназначен для создания и изменения реляционных баз данных, а также извлечения из них данных. Другими словами, SQL – это инструмент, с помощью которого человек управляет базой данных.

3. Из чего состоит язык SQL?

Язык SQL состоит из операторов, инструкций и вычисляемых функций.

4. В чем отличие СУБД SQLite от клиент-серверных СУБД?

С помощью SQLite создаются базы данных, представляющие собой один кроссплатформенный текстовый файл. Файл базы данных, в отличие от SQLite, не встраивается в приложение, не становится его частью, он существует отдельно. Так можно создать базу данных, пользуясь консольным `sqlite3`, после чего использовать ее в программе с помощью библиотеки SQLite языка программирования. При этом файл базы данных также хранится на локальной машине.

5. Как установить SQLite в Windows и Linux?

В Ubuntu установить `sqlite3` можно командой `sudo apt install sqlite3`. Для операционной системы Windows скачивают свой архив (`sqlite-tools-win32-*.zip`) и распаковывают.

6. Как создать базу данных SQLite?

С помощью `sqlite3` создать или открыть существующую базу данных можно двумя способами. Во-первых, при вызове утилиты `sqlite3` в качестве аргумента можно указать имя базы данных. Если БД существует, она будет открыта. Если ее нет, она будет создана и открыта.

7. Как выяснить в SQLite какая база данных является текущей?

Выяснить, какая база данных является текущей, можно с помощью команды `.databases` утилиты `sqlite3`.

8. Как создать и удалить таблицу в SQLite?

Таблицы базы данных создаются с помощью директивы `CREATE`

`TABLE` языка SQL. После `CREATE TABLE` идет имя таблицы, после которого в скобках перечисляются имена столбцов и их тип. Для удаления целой таблицы из базы данных используется директива `DROP TABLE`, после которой идет имя удаляемой таблицы.

9. Что является первичным ключом в таблице?

PRIMARY KEY – ограничитель, который заставляет СУБД проверять уникальность значения данного поля у каждой добавляемой записи.

10. Как сделать первичный ключ таблицы автоинкрементным?

Добавить AUTOINCREMENT в столбце при создании таблицы.

11. Каково назначение инструкций NOT NULL и DEFAULT при создании таблиц?

Ограничитель NOT NULL используют, чтобы запретить оставление поля пустым.

DEFAULT задает значение по умолчанию.

12. Каково назначение внешних ключей в таблице? Как создать внешний ключ в таблице?

С помощью внешнего ключа устанавливается связь между записями разных таблиц.

Чтобы включить поддержку внешних ключей в sqlite3, надо выполнить команду PRAGMA foreign_keys = ON. После этого добавить в таблицу запись, в которой внешний ключ не совпадает ни с одним первичным из другой таблицы, не получится.

13. Как выполнить вставку строки в таблицу базы данных SQLite?

С помощью оператора INSERT языка SQL выполняется вставка данных в таблицу.

14. Как выбрать данные из таблицы SQLite?

С помощью оператора SELECT осуществляется выборочный просмотр данных из таблицы.

15. Как ограничить выборку данных с помощью условия WHERE?

Условие WHERE используется не только с оператором SELECT, также с UPDATE и DELETE. С помощью WHERE определяются строки, которые будут выбраны, обновлены или удалены. По сути это фильтр.

16. Как упорядочить выбранные данные?

При выводе данных их можно не только фильтровать с помощью WHERE, но и сортировать по возрастанию или убыванию с помощью оператора ORDER BY.

17. Как выполнить обновление записей в таблице SQLite?

UPDATE ... SET – обновление полей записи

18. Как удалить записи из таблицы SQLite?

DELETE FROM – удаление записей таблицы

19. Как сгруппировать данные из выборки из таблицы SQLite?

В SQL кроме функций агрегирования есть оператор GROUP BY, который выполняет группировку записей по вариациям заданного поля.

20. Как получить значение агрегатной функции (например: минимум, максимум, количество записей и т. д.) в выборке из таблицы SQLite?

Для этих целей в языке SQL предусмотрены различные функции агрегирования данных.

Наиболее используемые – count(), sum(), avr(), min(), max().

21. Как выполнить объединение нескольких таблиц в операторе SELECT?

После FROM указываются обе сводимые таблицы через JOIN. В данном случае неважно, какую указывать до JOIN, какую после. После ключевого слова ON записывается условие сведения. Условие сообщает, как соединять строки разных таблиц.

22. Каково назначение подзапросов и шаблонов при работе с таблицами SQLite?

Шаблоны реализуют поиск по таблице, если неизвестно полное название данных в строке.

Подзапросы помогают уменьшить работу путём создания дополнительного запроса внутри основного.

23. Каково назначение представлений VIEW в SQLite?

Бывает удобно сохранить результат выборки для дальнейшего использования. Для этих целей в языке SQL используется оператор CREATE VIEW, который создает представление – виртуальную таблицу. В эту виртуальную таблицу как бы сохраняется результатзапроса.

24. Какие существуют средства для импорта данных в SQLite?

```
.import --csv city.csv city
```

25. Каково назначение команды .schema ?

Показывает какие столбцы есть в таблице, тип их данных и прочие свойства.

26. Как выполняется группировка и сортировка данных в запросах SQLite?

```
select federal_district as district,count(*) as city_count from citygroup by 1  
order by 2 desc;
```

27. Каково назначение "табличных выражений" в SQLite?

Выражение with history as (...) создает именованный запрос. Название — history , а содержание — селект в скобках (век основания для каждого города).

К history можно обращаться поимени в остальном запросе, что мы и делаем.

28. Как осуществляется экспорт данных из SQLite в форматы CSV и JSON?

`.mode csv`

29. Какие еще форматы для экспорта данных Вам известны?

- `.mode list`
- `.mode json`