

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.20
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Хачатрян Владимир Владимирович
2 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики: Воронкин
Р. А., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: Основы работы с SQLite3.

Цель: исследовать базовые возможности системы управления базами данных SQLite3.

Выполнение:

Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер.

```
C:\Users\vovax>git clone https://github.com/GitVolodya/2.20.git
Cloning into '2.20'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
C:\Users\vovax>_
```

Рисунок 1. Клонирование репозитория

Создал виртуальное окружение conda и активировал его, также установил необходимые пакеты isort, black, flake8.

```
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.1.0
  latest version: 23.10.0

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

    conda install conda=23.10.0

## Package Plan ##

environment location: C:\Users\Gaming-PC\.conda\envs\2.20

added / updated specs:
- python=3.10
```

Рисунок 2. Создание ВО

Решить задачу: выполнить в песочнице команды:

```
create table customer(name);  
  
select *  
from customer;  
  
.schema customer
```

Рисунок 3. Команды для выполнения задачи

```
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);  
sqlite> _
```

Рисунок 4. Выполнение

Вот что здесь происходит:

Первая команда `create table` создает таблицу `customer` с единственным столбцом `name`.

Команда `select` показывает содержимое таблицы `customer` (она пустая).

Команда `.schema` показывает список и структуру всех таблиц в базе.

`create` и `select` — это SQL-запросы, часть стандарта SQL. Запрос может занимать несколько строк, а в конце всегда ставится точка с запятой.

`.schema` — это специальная команда SQLite, не часть стандарта SQL.

Решить задачу: с помощью команды `.help` найти в песочнице команду, которая отвечает за вывод времени выполнения запроса.

```
sqlite> create table city(name);  
sqlite> .timer on  
sqlite> select count(*) from city;  
0  
Run Time: real 0.000 user 0.000154 sys 0.000000  
sqlite>
```

Рисунок 5. Выполнение

Решить задачу: загрузить файл city.csv в песочнице, затем выполнить запрос, который вернет число, получить число после выполнения запроса.

```
sqlite> .import --csv city.csv city
sqlite> select max(length(city)) from city;
25
sqlite>
```

Рисунок 6. Выполнение

Решить задачу: загрузить файл city.csv в песочницу 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.

```
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .mode csv
sqlite> .import city.csv city
sqlite>
```

Рисунок 7. Выполнение

Решить задачу: написать в песочнице запрос, который посчитает количество городов для каждого часового пояса в Сибирском и Приволжском федеральных округах. Вывести столбцы timezone и city_count, отсортируйте по значению часового пояса:

```
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .import --csv city.csv city
sqlite> select timezone, COUNT(*) as city_count
...> from city
...> where federal_district in ('Сибирский', 'Приволжский')
...> group by timezone
...> order by timezone;
UTC+3|101
UTC+4|41
UTC+5|58
UTC+6|6
UTC+7|86
UTC+8|22
sqlite> █
```

Рисунок 8. Выполнение

Решить задачу: написать в песочнице запрос, который найдет три ближайших к Самаре города, не считая саму Самару. Указать в ответе названия этих трех городов через запятую в порядке удаления от Самары.

```
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> CREATE TABLE cities (
...> city_name TEXT,
...> latitude REAL,
...> longitude REAL
...> );
sqlite> INSERT INTO cities (city_name, latitude, longitude)
...> VALUES
...> ('Самара', 53.195873, 50.100193),
...> ('Ульяновск', 54.308067, 48.374867),
...> ('Пенза', 53.200066, 45.004944),
...> ('Саранск', 54.180760, 45.186226);
sqlite> SELECT city_name
...> FROM cities
...> WHERE city_name != 'Самара'
...> ORDER BY ABS(latitude - (SELECT latitude FROM cities WHERE city_name = 'Самара')) + ABS(longitude - (SELECT longitude FROM cities WHERE city_name = 'Самара'))
...> LIMIT 3;
Ульяновск
Пенза
Саранск
sqlite> █
```

Рисунок 9. Выполнение

Решить задачу: написать в песочнице запрос, который посчитает количество городов в каждом часовом поясе. Отсортировать по количеству городов по убыванию.

```
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .import --csv city.csv city
sqlite> SELECT timezone, COUNT(*) AS city_count
...> from city
...> GROUP BY timezone
...> ORDER BY city_count 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 10. Выполнение

Выполнить этот же запрос, но так, чтобы результат был

- в формате CSV
- с заголовками

```

Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .import --csv city.csv city
sqlite> .headers on
sqlite> .mode csv
sqlite> .separator |
sqlite> SELECT timezone, COUNT(*) AS city_count
...> FROM city
...> GROUP BY timezone
...> ORDER BY city_count 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 11. Выполнение седьмого общего с изменениями

Индивидуальное задание

Условие задания: Загрузить в SQLite выбранный датасет в формате CSV (датасет можно найти на сайте Kaggle). Сформировать более пяти запросов к таблицам БД. Выгрузить результат выполнения запросов в форматы CSV и JSON.

Скачал датасет с сайта Kaggle под названием Billionaires Statistics Dataset.

Создадим запросы к таблицам БЗ.

Для добавления файла json используем команду:

```

sqlite> .mode json
sqlite> .output billionaires_min.json

```

1. Выполнение запроса: подсчитать количество миллиардеров в каждой стране:

```
sqlite> .mode csv
sqlite> .headers on
sqlite> .output billionaires_country.csv
sqlite> SELECT country, COUNT(*) as count
...> FROM billionaires
...> GROUP BY country;
sqlite>
```

Рисунок 12. Выполнение первого запроса

country	count
1	0,2
2	30,1
3	33,1
4	36,1
5	36,2
6	39,2
7	41,2
8	43,4
9	45,1
10	47,3
11	49,3
12	50,3
13	51,6
14	52,5
15	53,3
16	54,2
17	55,5
18	56,7
19	57,6
20	58,8
21	59,8
22	60,7
23	61,4
24	62,2
25	63,3
26	64,3
27	65,6
28	66,4
29	67,6
30	68,6
31	69,7
32	70,4
33	71,3
34	72,8
35	73,4
36	74,9
37	75,5
38	76,1

Рисунок 13. Файл CSV для первого запроса

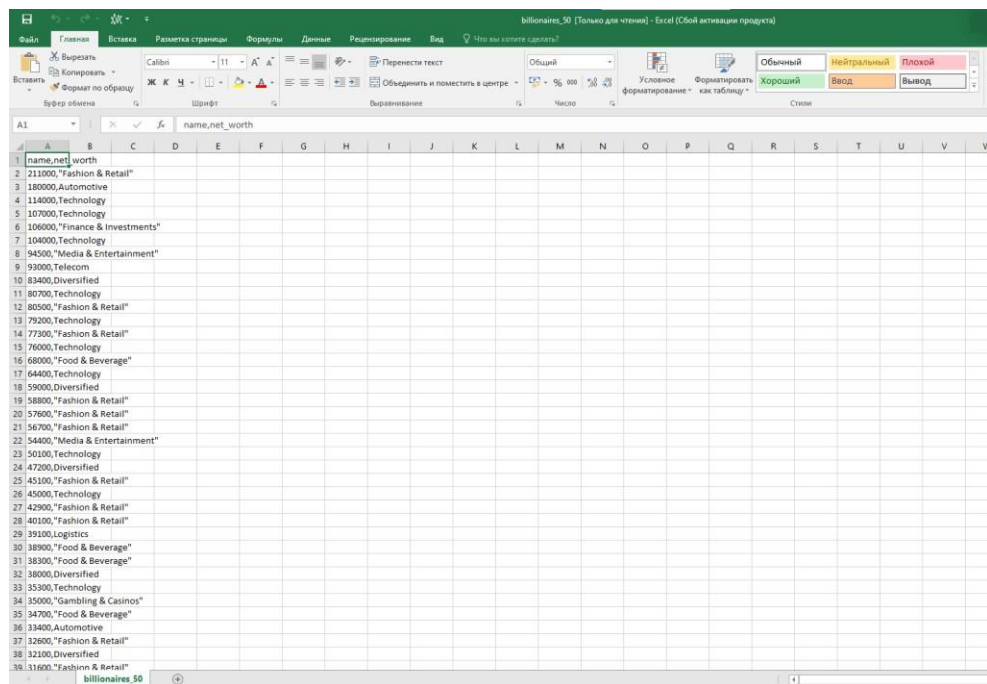
```
1 [{"country": "...", "count": 2},
2 [{"country": "30", "count": 1},
3 [{"country": "33", "count": 1},
4 [{"country": "36", "count": 1},
5 [{"country": "38", "count": 2},
6 [{"country": "39", "count": 2},
7 [{"country": "41", "count": 2},
8 [{"country": "43", "count": 4},
9 [{"country": "45", "count": 1},
10 [{"country": "47", "count": 3},
11 [{"country": "49", "count": 3},
12 [{"country": "50", "count": 3},
13 [{"country": "51", "count": 6},
14 [{"country": "52", "count": 5},
15 [{"country": "53", "count": 3},
16 [{"country": "54", "count": 2},
17 [{"country": "55", "count": 5},
18 [{"country": "56", "count": 7},
19 [{"country": "57", "count": 6},
20 [{"country": "58", "count": 8},
21 [{"country": "59", "count": 8},
22 [{"country": "60", "count": 7},
23 [{"country": "61", "count": 4},
24 [{"country": "62", "count": 2},
25 [{"country": "63", "count": 3},
26 [{"country": "64", "count": 3},
27 [{"country": "65", "count": 6},
28 [{"country": "66", "count": 4},
29 [{"country": "67", "count": 6},
30 [{"country": "68", "count": 6},
31 [{"country": "69", "count": 7},
32 [{"country": "70", "count": 4},
33 [{"country": "71", "count": 3},
34 [{"country": "72", "count": 8},
```

Рисунок 14. Файл JSON для первого запроса

2. Выполнение запроса: вывести имена и состояния (net worth) миллиардеров с состоянием более 50 миллиардов:

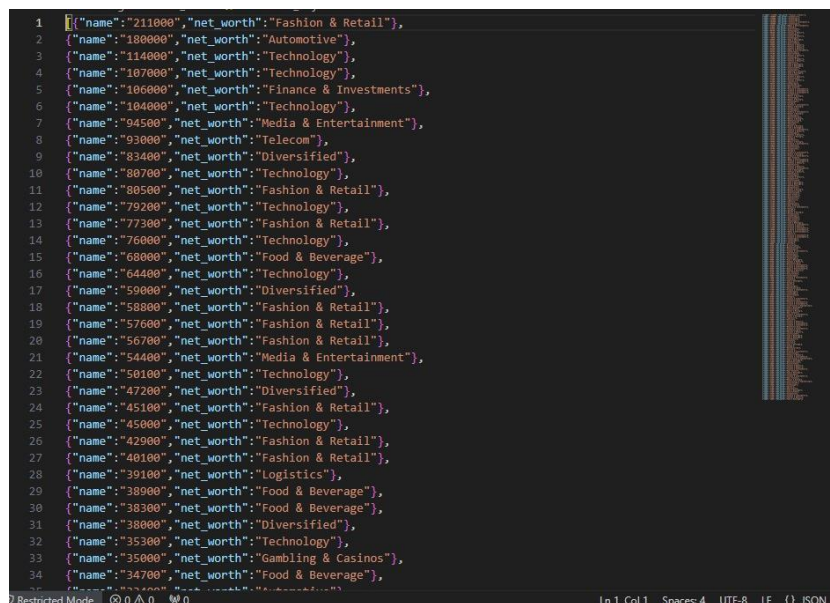
```
sqlite> .mode csv
sqlite> .headers on
sqlite> .output billionaires_50.csv
sqlite> SELECT name, net_worth
...> FROM billionaires
...> WHERE net_worth > 50.0;
sqlite> _
```

Рисунок 15. Выполнение второго запроса



A	B
1	name,net_worth
2	211000,"Fashion & Retail"
3	180000,"Automotive"
4	114000,"Technology"
5	107000,"Technology"
6	106000,"Finance & Investments"
7	104000,"Technology"
8	94500,"Media & Entertainment"
9	93000,"Telecom"
10	83400,"Diversified"
11	80700,"Technology"
12	80500,"Fashion & Retail"
13	79200,"Technology"
14	77300,"Fashion & Retail"
15	76000,"Technology"
16	68000,"Food & Beverage"
17	64400,"Technology"
18	59000,"Diversified"
19	58800,"Fashion & Retail"
20	57600,"Fashion & Retail"
21	56700,"Fashion & Retail"
22	54400,"Media & Entertainment"
23	50100,"Technology"
24	47200,"Diversified"
25	45100,"Fashion & Retail"
26	45000,"Technology"
27	42900,"Fashion & Retail"
28	40100,"Fashion & Retail"
29	39100,"Logistics"
30	38900,"Food & Beverage"
31	38300,"Food & Beverage"
32	38000,"Diversified"
33	35300,"Technology"
34	35000,"Gambling & Casinos"
35	34700,"Food & Beverage"
36	33400,"Automotive"
37	32600,"Fashion & Retail"
38	32100,"Diversified"
39	31600,"Fashion & Retail"

Рисунок 16. Файл CSV для второго запроса



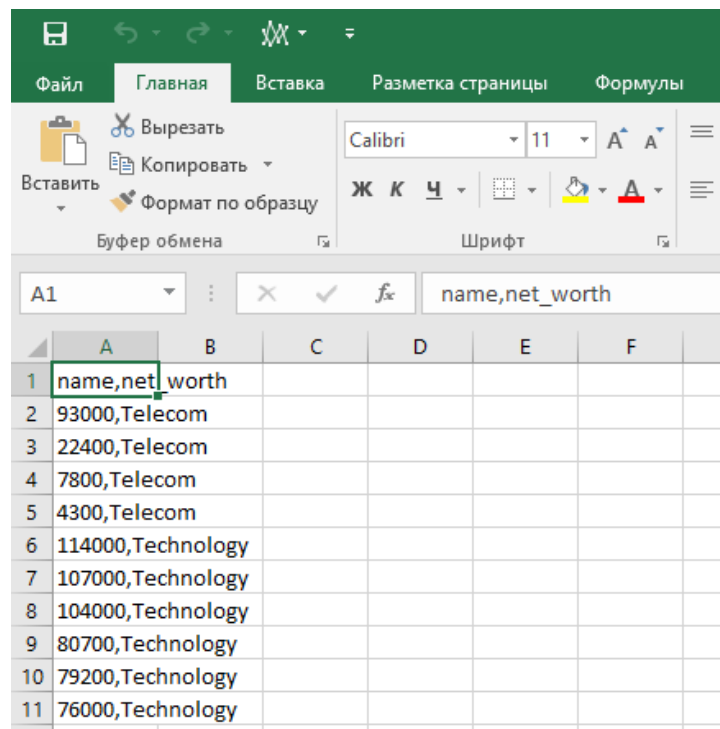
```
1 [{"name": "211000", "net_worth": "Fashion & Retail"},
2 [{"name": "180000", "net_worth": "Automotive"},
3 [{"name": "114000", "net_worth": "Technology"},
4 [{"name": "107000", "net_worth": "Technology"},
5 [{"name": "106000", "net_worth": "Finance & Investments"},
6 [{"name": "104000", "net_worth": "Technology"},
7 [{"name": "94500", "net_worth": "Media & Entertainment"},
8 [{"name": "93000", "net_worth": "Telecom"},
9 [{"name": "83400", "net_worth": "Diversified"},
10 [{"name": "80700", "net_worth": "Technology"},
11 [{"name": "80500", "net_worth": "Fashion & Retail"},
12 [{"name": "79200", "net_worth": "Technology"},
13 [{"name": "77300", "net_worth": "Fashion & Retail"},
14 [{"name": "76000", "net_worth": "Technology"},
15 [{"name": "68000", "net_worth": "Food & Beverage"},
16 [{"name": "64400", "net_worth": "Technology"},
17 [{"name": "59000", "net_worth": "Diversified"},
18 [{"name": "58800", "net_worth": "Fashion & Retail"},
19 [{"name": "57600", "net_worth": "Fashion & Retail"},
20 [{"name": "56700", "net_worth": "Fashion & Retail"},
21 [{"name": "54400", "net_worth": "Media & Entertainment"},
22 [{"name": "50100", "net_worth": "Technology"},
23 [{"name": "47200", "net_worth": "Diversified"},
24 [{"name": "45100", "net_worth": "Fashion & Retail"},
25 [{"name": "45000", "net_worth": "Technology"},
26 [{"name": "42900", "net_worth": "Fashion & Retail"},
27 [{"name": "40100", "net_worth": "Fashion & Retail"},
28 [{"name": "39100", "net_worth": "Logistics"},
29 [{"name": "38900", "net_worth": "Food & Beverage"},
30 [{"name": "38300", "net_worth": "Food & Beverage"},
31 [{"name": "38000", "net_worth": "Diversified"},
32 [{"name": "35300", "net_worth": "Technology"},
33 [{"name": "35000", "net_worth": "Gambling & Casinos"},
34 [{"name": "34700", "net_worth": "Food & Beverage"},
35 [{"name": "33400", "net_worth": "Automotive"},
36 [{"name": "32600", "net_worth": "Fashion & Retail"},
37 [{"name": "32100", "net_worth": "Diversified"},
38 [{"name": "31600", "net_worth": "Fashion & Retail"}]
```

Рисунок 17. Файл JSON для второго запроса

3. Выполнение запроса: вывести миллиардеров с наибольшим состоянием:

```
sqlite> .mode csv
sqlite> .headers on
sqlite> .output top10_billionaires.csv
sqlite> SELECT name, net_worth
...> FROM billionaires
...> ORDER BY net_worth DESC
...> LIMIT 10;
sqlite>
```

Рисунок 18. Выполнение третьего запроса



	A	B	C	D	E	F
1	name,net_worth					
2	93000,Telecom					
3	22400,Telecom					
4	7800,Telecom					
5	4300,Telecom					
6	114000,Technology					
7	107000,Technology					
8	104000,Technology					
9	80700,Technology					
10	79200,Technology					
11	76000,Technology					

Рисунок 19. Файл CSV для третьего запроса

Рисунок 20. Файл JSON для третьего запроса

```
1 [{"name": "93000", "net_worth": "Telecom"},
2 {"name": "22400", "net_worth": "Telecom"},
3 {"name": "7800", "net_worth": "Telecom"},
4 {"name": "4300", "net_worth": "Telecom"},
5 {"name": "114000", "net_worth": "Technology"},
6 {"name": "107000", "net_worth": "Technology"},
7 {"name": "104000", "net_worth": "Technology"},
8 {"name": "80700", "net_worth": "Technology"},
9 {"name": "79200", "net_worth": "Technology"},
10 {"name": "76000", "net_worth": "Technology"}]
```

4. Выполнение запроса: Вывести среднее состояние миллиардеров для каждой страны:

```

sqlite> .mode csv
sqlite> .headers on
sqlite> .output billionaires_strani.csv
sqlite> SELECT country, AVG(net_worth) as average_net_worth
...> FROM billionaires
...> GROUP BY country;
sqlite>

```

Рисунок 21. Выполнение четвертого запроса

country	average_net_worth
	0,0
	30,0
	33,0
	36,0
	38,0
	39,0
	41,0
	43,0
	45,0
	47,0
	49,0
	50,0
	51,0
	52,0
	53,0
	54,0
	55,0
	56,0
	57,0
	58,0
	59,0
	60,0
	61,0
	62,0
	63,0
	64,0
	65,0
	66,0
	67,0
	68,0
	69,0
	70,0
	71,0
	72,0
	73,0
	74,0
	75,0

Рисунок 22. Файл CSV для четвертого запроса

```

1 [{"country":"","average_net_worth":0.0},
2 {"country":"30","average_net_worth":0.0},
3 {"country":"33","average_net_worth":0.0},
4 {"country":"36","average_net_worth":0.0},
5 {"country":"38","average_net_worth":0.0},
6 {"country":"39","average_net_worth":0.0},
7 {"country":"41","average_net_worth":0.0},
8 {"country":"43","average_net_worth":0.0},
9 {"country":"45","average_net_worth":0.0},
10 {"country":"47","average_net_worth":0.0},
11 {"country":"49","average_net_worth":0.0},
12 {"country":"50","average_net_worth":0.0},
13 {"country":"51","average_net_worth":0.0},
14 {"country":"52","average_net_worth":0.0},
15 {"country":"53","average_net_worth":0.0},
16 {"country":"54","average_net_worth":0.0},
17 {"country":"55","average_net_worth":0.0},
18 {"country":"56","average_net_worth":0.0},
19 {"country":"57","average_net_worth":0.0},
20 {"country":"58","average_net_worth":0.0},
21 {"country":"59","average_net_worth":0.0},
22 {"country":"60","average_net_worth":0.0},
23 {"country":"61","average_net_worth":0.0},
24 {"country":"62","average_net_worth":0.0},
25 {"country":"63","average_net_worth":0.0},
26 {"country":"64","average_net_worth":0.0},
27 {"country":"65","average_net_worth":0.0},
28 {"country":"66","average_net_worth":0.0},
29 {"country":"67","average_net_worth":0.0},
30 {"country":"68","average_net_worth":0.0},
31 {"country":"69","average_net_worth":0.0},
32 {"country":"70","average_net_worth":0.0},
33 {"country":"71","average_net_worth":0.0},
34 {"country":"72","average_net_worth":0.0},
35 {"country":"73","average_net_worth":0.0},
36 {"country":"74","average_net_worth":0.0},
37 {"country":"75","average_net_worth":0.0},
38 {"country":"76","average_net_worth":0.0},
39 {"country":"77","average_net_worth":0.0},
40 {"country":"78","average_net_worth":0.0},
41 {"country":"79","average_net_worth":0.0},
42 {"country":"80","average_net_worth":0.0},
43 {"country":"81","average_net_worth":0.0},
44 {"country":"82","average_net_worth":0.0},
45 {"country":"83","average_net_worth":0.0},
46 {"country":"84","average_net_worth":0.0},
47 {"country":"85","average_net_worth":0.0},
48 {"country":"86","average_net_worth":0.0},
49 {"country":"87","average_net_worth":0.0},
50 {"country":"88","average_net_worth":0.0},
51 {"country":"89","average_net_worth":0.0},
52 {"country":"90","average_net_worth":0.0},
53 {"country":"91","average_net_worth":0.0},
54 {"country":"92","average_net_worth":0.0},
55 {"country":"93","average_net_worth":0.0},
56 {"country":"94","average_net_worth":0.0},
57 {"country":"95","average_net_worth":0.0},
58 {"country":"96","average_net_worth":0.0},
59 {"country":"97","average_net_worth":0.0},
60 {"country":"98","average_net_worth":0.0},
61 {"country":"99","average_net_worth":0.0},
62 {"country":"","average_net_worth":0.0}]]

```

Рисунок 23. Файл JSON для четвертого запроса

5. Выполнение запроса: Вывести миллиардеров с минимальным состоянием:

```

sqlite> .mode csv
sqlite> .headers on
sqlite> .output billionaires_min.csv
sqlite> SELECT name, net_worth
...> FROM billionaires
...> WHERE net_worth = (SELECT MIN(net_worth) FROM billionaires);
sqlite>

```

Рисунок 24. Выполнение пятого запроса

name,net_worth						
1	name,net_worth					
2	180000,Automotive					
3	33400,Automotive					
4	27400,Automotive					
5	24600,Automotive					
6	19000,Automotive					
7	18700,Automotive					
8	15200,Automotive					
9	14500,Automotive					
10	13200,Automotive					
11	12100,Automotive					
12	1300,Automotive					
13						

Рисунок 25. Файл CSV для пятого запроса

```
1 [{"name": "180000", "net_worth": "Automotive"},
2 {"name": "33400", "net_worth": "Automotive"},
3 {"name": "27400", "net_worth": "Automotive"},
4 {"name": "24600", "net_worth": "Automotive"},
5 {"name": "19000", "net_worth": "Automotive"},
6 {"name": "18700", "net_worth": "Automotive"},
7 {"name": "15200", "net_worth": "Automotive"},
8 {"name": "14500", "net_worth": "Automotive"},
9 {"name": "13200", "net_worth": "Automotive"},
10 {"name": "12100", "net_worth": "Automotive"},
11 {"name": "1300", "net_worth": "Automotive"}]
```

Рисунок 26. Файл JSON пятого запроса

Создал файл `envirement.yml` и деактивировал виртуальное окружение.
Отправил на удаленный репозиторий.

Рисунок 27. Деактивация ВО

Ссылка: <https://github.com/GitVolodya/2.20.git>

Контрольные вопросы:

1. *Каково назначение реляционных баз данных и СУБД?*

- Реляционные базы данных предназначены для организации и хранения данных в виде таблиц с отношениями между ними.
- Системы управления базами данных (СУБД) предоставляют средства для создания, управления и обращения с базами данных.

2. *Каково назначение языка SQL?*

SQL (Structured Query Language) используется для взаимодействия с реляционными базами данных. Он предоставляет стандартизированный способ создания, изменения, управления и запросов к данным в базе данных.

3. *Из чего состоит язык SQL?*

SQL состоит из нескольких подмножеств:

- DDL (Data Definition Language): Определение структуры базы данных (CREATE, ALTER, DROP).

- DML (Data Manipulation Language): Манипуляция данными (SELECT, INSERT, UPDATE, DELETE).

- DCL (Data Control Language): Управление доступом и правами (GRANT, REVOKE).

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

- SQLite является встроенной базой данных и хранится в виде одного файла.

- Клиент-серверные СУБД (например, MySQL, PostgreSQL) имеют отдельные серверы, к которым подключаются клиенты для доступа к данным.

5. Как установить SQLite в Windows и Linux?

- Windows: Можно загрузить исполняемый файл SQLite с официального сайта и выполнить установку.

- Linux: В большинстве дистрибутивов Linux SQLite уже установлен. Для установки можно воспользоваться менеджером пакетов (например, `sudo apt-get install sqlite` в Ubuntu).

6. Как создать базу данных SQLite?

- В командной строке SQLite: `sqlite3 имя_базы_данных.db`.

- Внутри SQLite: `CREATE DATABASE имя_базы_данных;`

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

- В командной строке SQLite: `.database` или `.dbinfo`.

- Внутри SQLite: `PRAGMA database_list;`

8. Как создать и удалить таблицу в SQLite?

Создание таблицы:

`CREATE TABLE название (`

`поле1 тип1,`

`...`

`);`

Удаление таблицы:

DROP TABLE название;

9. Что является первичным ключом в таблице?

Первичный ключ (Primary Key) в таблице — это уникальный идентификатор каждой записи. Он обеспечивает уникальность идентификации записей в таблице.

10. Как сделать первичный ключ таблицы автоинкрементным?

При создании таблицы в SQLite можно сделать поле первичного ключа автоинкрементным, используя ключевое слово AUTOINCREMENT

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

- NOT NULL: Гарантирует, что значение в столбце не может быть NULL (пустым).

- DEFAULT value: Устанавливает значение по умолчанию для столбца, если вставляемые данные не предоставляют значение для этого столбца.

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

Внешний ключ (Foreign Key) используется для связи двух таблиц по значениям в столбцах. Он обеспечивает целостность ссылочной целостности данных.

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

Используйте оператор INSERT INTO.

14. Как выбрать данные из таблицы SQLite?

Используйте оператор SELECT.

15. Как ограничить выборку данных с помощью условия WHERE?

Используйте WHERE для установки условий:

SELECT column1, column2, ...

FROM table_name

WHERE condition;

16. Как упорядочить выбранные данные?

Используйте ORDER BY.

17. Как выполнить обновление записей в таблице SQLite?

Используйте UPDATE.

18. Как удалить записи из таблицы SQLite?

Используйте DELETE.

19. Как сгруппировать данные из выборке из таблицы SQLite?

Используйте GROUP BY.

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

Используйте агрегатные функции, такие как MIN, MAX, SUM, AVG, и т. д.:

21. Как выполнить объединение нескольких таблиц в операторе SELECT?

Используйте JOIN.

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

Подзапросы используются для вложенных запросов, а шаблоны предоставляют средства создания более обобщенных запросов.

23. Каково назначение представлений VIEW в SQLite?

Представления позволяют создавать виртуальные таблицы на основе результатов запросов, что облегчает повторное использование и улучшает структуру запросов.

24. Какие существуют средства для импорта данных в SQLite?

QLite предоставляет команды .import и .read для импорта данных из внешних источников, а также можно использовать SQL-запросы с оператором INSERT.

25. Каково назначение команды .schema ?

Команда `.schema` используется для вывода SQL-кода, описывающего структуру базы данных, включая определение таблиц, индексов и других объектов

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

Используйте `GROUP BY` для объединения строк по значениям в одном или нескольких столбцах.

27. Каково назначение "табличных выражений" в SQLite?

Табличные выражения (Common Table Expressions или CTE) позволяют создавать временные результаты запросов, которые можно использовать внутри других запросов. Это обеспечивает более чистый и структурированный код SQL.

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

Экспорт в CSV: Используйте команду `.mode csv` перед выполнением запроса, а затем `.output filename.csv` для указания файла вывода.

29. Какие еще форматы для экспорта данных Вам известны?

Дополнительно к CSV и JSON, SQLite поддерживает экспорт в форматах XML, HTML, и SQL INSERT. Для экспорта в эти форматы также можно использовать соответствующие команды `.mode` и `.output`:

Вывод: исследовал базовые возможности системы управления базами данных SQLite3.