

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 №6**  
**дисциплины «Анализ данных»**

Выполнил:  
Гайчук Дарья Дмитриевна  
2 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика  
и вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», очная  
форма обучения

---

(подпись)

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

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2024 г.

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

### Порядок выполнения работы:

1. Создала новый репозиторий и клонировала его на свой компьютер.

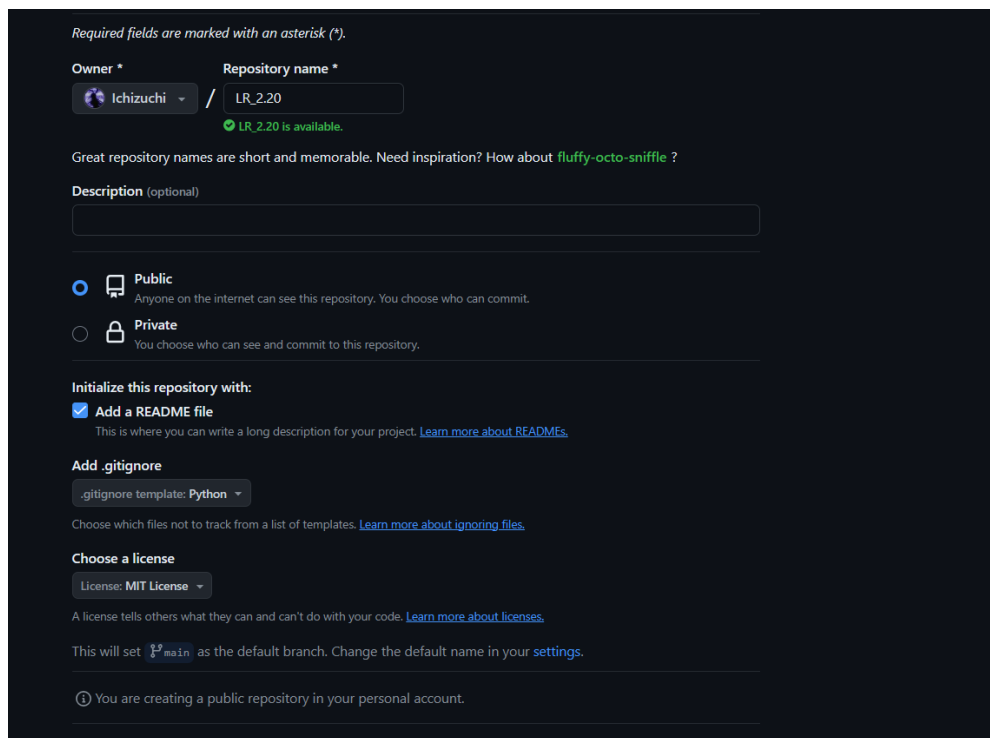


Рисунок 1 – Создан новый репозиторий

2. Клонировала репозиторий на свой компьютер. В ходе данной лабораторной работы работала с моделью ветвления git-flow.

```
● @Ichizuchi →/workspaces/LR_2.20 (main) $ git branch develop
● @Ichizuchi →/workspaces/LR_2.20 (main) $ git clone https://github.com/Ichizuchi/LR_2.20.git
Cloning into 'LR_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.
```

Рисунок 2 – Клонирование и модель ветвления git-flow

3. Создала виртуальное окружение Anaconda с именем репозитория.

```

@Ichizuchi →/workspaces/LR_2.20 (develop) $ conda create -n myenv python=3.10
Channels:
  - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/conda/envs/myenv

added / updated specs:
  - python=3.10

The following packages will be downloaded:


```

package	build	
_libgcc_mutex-0.1	main	3 KB
_openmp_mutex-5.1	1_gnu	21 KB
bzip2-1.0.8	h5eee18b_5	262 KB
ca-certificates-2024.3.11	h06a4308_0	127 KB
ld_impl_linux-64-2.38	h1181459_1	654 KB
libffi-3.4.4	h6a678d5_0	142 KB
libgcc-ng-11.2.0	h1234567_1	5.3 MB
libgomp-11.2.0	h1234567_1	474 KB
libstdcxx-ng-11.2.0	h1234567_1	4.7 MB

Рисунок 3 – Создание виртуального окружения

Создадим таблицу и отобразим её схему

```

Last login: Tue Apr 23 11:45:04 2024 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> create table customer (name);
sqlite> select *
...> from customer;
sqlite> .schema customer
CREATE TABLE customer (name);
sqlite>

```

Рисунок 4 – Схема

Если ее включить команду timer on, в результатах запроса добавится  
нужная строчка

```
sqlite> .timer on
sqlite> select count(*) from city;
Run Time: real 0.001 user 0.000043 sys 0.000043
Parse error: no such table: city
sqlite> █
```

Рисунок 5 – Результат запроса

Выполним запрос и получим ответ

```
sqlite> .import --csv city.csv city
sqlite> select max(length(city)) from city;
25
Run Time: real 0.001 user 0.000545 sys 0.000363
sqlite>
```

Рисунок 6 – Ответ на запрос

Загрузите файл city.csv в песочнице с помощью команды. import, но без использования опции --csv. Сделаем. Mode

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

Рисунок 7 – Результат выполнения

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.

Выполним следующий 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;
UTC+3,202
UTC+4,82
UTC+5,116
UTC+6,12
UTC+7,172
UTC+8,44
Run Time: real 0.003 user 0.002411 sys 0.000000

```

Рисунок 8 – Результат выполнения

Решите задачу: напишите в песочнице запрос, который найдет три ближайших к Самаре города, не считая саму Самару. Если не получится, не расстраивайтесь — задача действительно непростая. Вернитесь к ней, когда пройдете все модули курса — и увидите, как все изменилось.

```

sqlite> SELECT city_name,
...>      SQRT(POW(latitude - samara_latitude, 2) + POW(longitude - samara_longitude, 2)) AS distance
...> FROM city
...> CROSS JOIN (SELECT latitude AS samara_latitude, longitude AS samara_longitude
FROM city WHERE city_name = 'Самара') AS samara
...> WHERE city_name != 'Самара'
...> ORDER BY distance
...> LIMIT 3;
Город3|0.0
Город5|0.0406647267296882
Город2|0.0888689484578318

```

Рисунок 9 – Результат выполнения

Напишите в песочнице запрос, который посчитает количество городов в каждом часовом поясе. Отсортируйте по количеству городов по убыванию.

```

sqlite> SELECT timezone, COUNT(*) AS city_count
...> FROM city
...> GROUP BY timezone
...> ORDER BY city_count DESC;
UTC+7|2
UTC+3|2
UTC+5|1
UTC+10|1
sqlite>

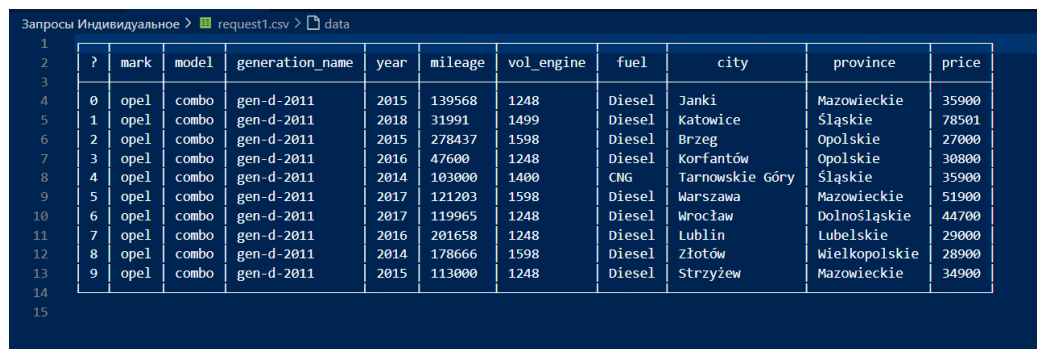
```

Рисунок 10 – Результат выполнения

## Выполнение индивидуального задания

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.

Запрос: `select * from cars limit 10;`

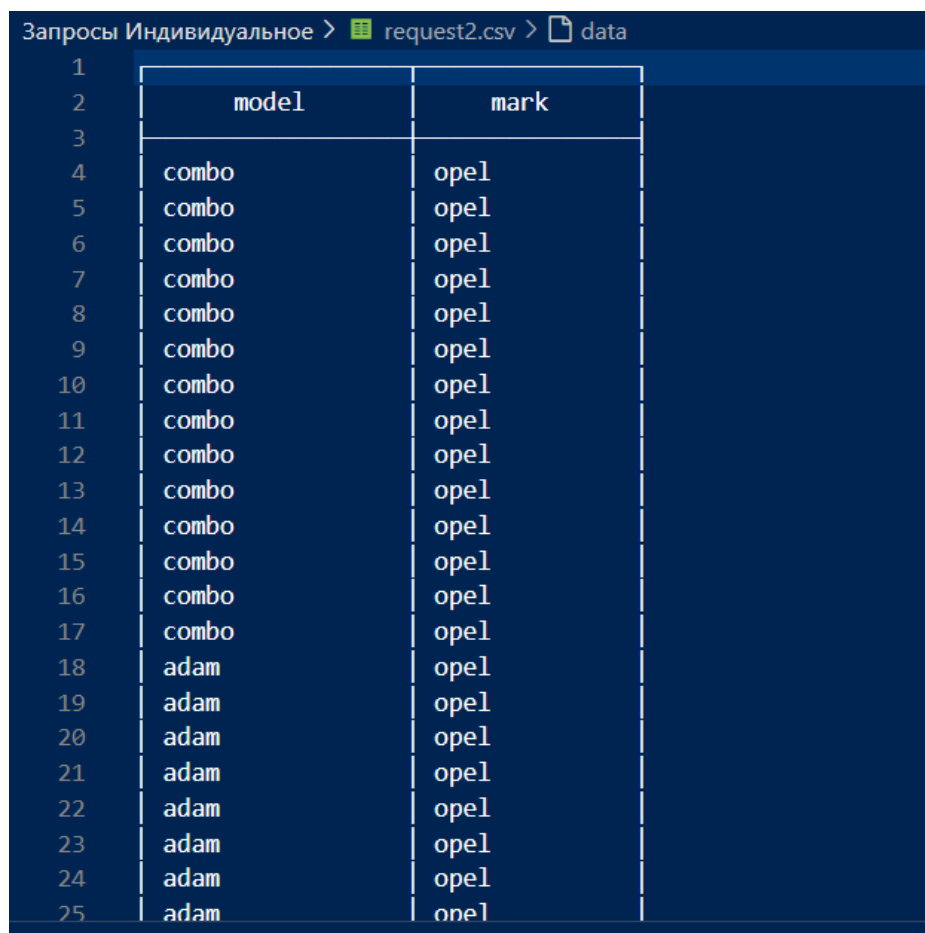


Запросы Индивидуальное > request1.csv > data

?	mark	model	generation_name	year	mileage	vol_engine	fuel	city	province	price
0	opel	combo	gen-d-2011	2015	139568	1248	Diesel	Janki	Mazowieckie	35900
1	opel	combo	gen-d-2011	2018	31991	1499	Diesel	Katowice	śląskie	78501
2	opel	combo	gen-d-2011	2015	278437	1598	Diesel	Brzeg	Opolskie	27000
3	opel	combo	gen-d-2011	2016	47600	1248	Diesel	Korfantów	Opolskie	30800
4	opel	combo	gen-d-2011	2014	103000	1400	CNG	Tarnowskie Góry	śląskie	35900
5	opel	combo	gen-d-2011	2017	121203	1598	Diesel	Warszawa	Mazowieckie	51900
6	opel	combo	gen-d-2011	2017	119965	1248	Diesel	Wrocław	Dolnośląskie	44700
7	opel	combo	gen-d-2011	2016	201658	1248	Diesel	Lublin	Lubelskie	29000
8	opel	combo	gen-d-2011	2014	178666	1598	Diesel	Złotów	Wielkopolskie	28900
9	opel	combo	gen-d-2011	2015	113000	1248	Diesel	Strzyżew	Mazowieckie	34900

Рисунок 11. Первый запрос

Запрос: `select model, mark from cars where year = 2015;`

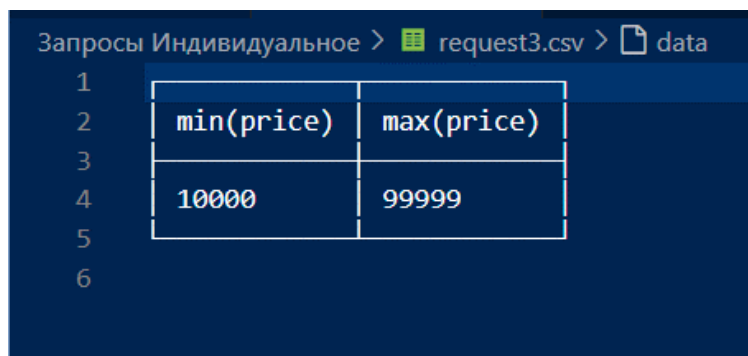


Запросы Индивидуальное > request2.csv > data

	model	mark
1		
2		
3		
4	combo	opel
5	combo	opel
6	combo	opel
7	combo	opel
8	combo	opel
9	combo	opel
10	combo	opel
11	combo	opel
12	combo	opel
13	combo	opel
14	combo	opel
15	combo	opel
16	combo	opel
17	combo	opel
18	adam	opel
19	adam	opel
20	adam	opel
21	adam	opel
22	adam	opel
23	adam	opel
24	adam	opel
25	adam	opel

Рисунок 12. Второй запрос

Запрос: `select min(price), max(price) from cars where mark = 'audi';`

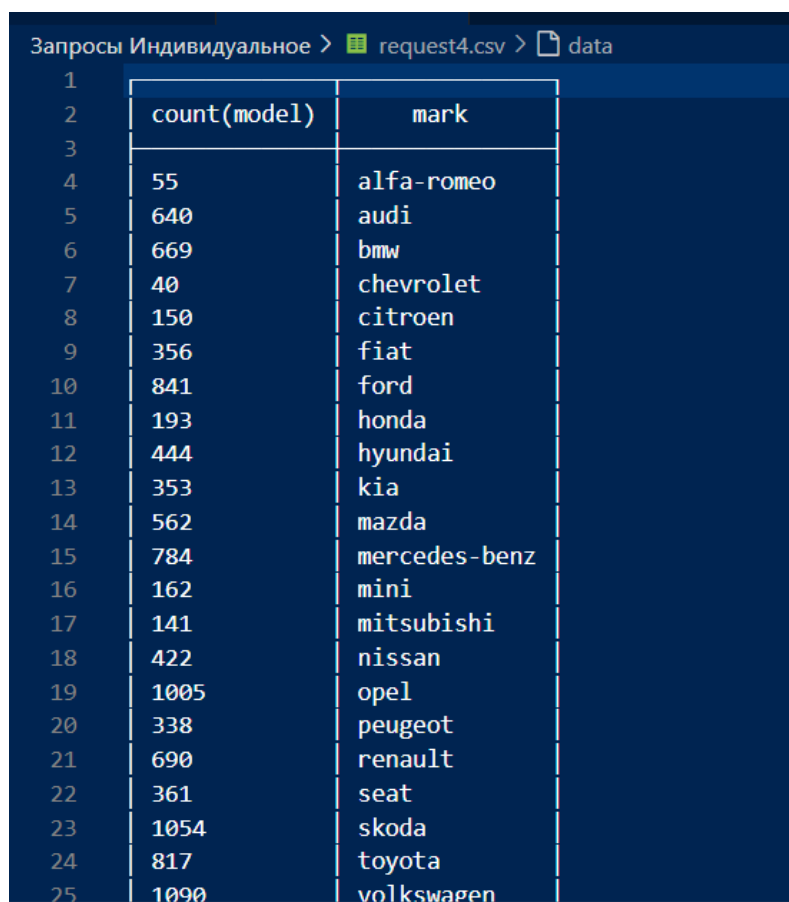


The screenshot shows a database interface with a query window titled 'Запросы Индивидуальное > request3.csv > data'. The query result is displayed in a table with two columns: 'min(price)' and 'max(price)'. The values are 10000 and 99999 respectively. The interface includes a line editor on the left with line numbers 1 through 6.

	min(price)	max(price)
1		
2	min(price)	max(price)
3		
4	10000	99999
5		
6		

Рисунок 13. Третий запрос

Запрос: `select count(model), mark from cars where year Between 2016 and 2018 and fuel = 'Gasoline' Group by 2;`



The screenshot shows a database interface with a query window titled 'Запросы Индивидуальное > request4.csv > data'. The query result is displayed in a table with two columns: 'count(model)' and 'mark'. The results are listed in 25 rows, showing the count of models for each car mark. The interface includes a line editor on the left with line numbers 1 through 25.

	count(model)	mark
1		
2	count(model)	mark
3		
4	55	alfa-romeo
5	640	audi
6	669	bmw
7	40	chevrolet
8	150	citroen
9	356	fiat
10	841	ford
11	193	honda
12	444	hyundai
13	353	kia
14	562	mazda
15	784	mercedes-benz
16	162	mini
17	141	mitsubishi
18	422	nissan
19	1005	opel
20	338	peugeot
21	690	renault
22	361	seat
23	1054	skoda
24	817	toyota
25	1090	volkswagen

Рисунок 14. Четвертый запрос

Запрос: `select city, Round(avg(price),1), count(*) as count from cars where year between date('now', '-3 years') and date('now') Group by 1 Order by Count desc;`

Запросы Индивидуальное > request5.csv > data

	city	Round(avg(price),1)	count
1			
2			
3			
4	Łódź	191325.4	324
5	Chorzów	196981.1	241
6	Warszawa	230206.4	240
7	Gdańsk	252558.8	174
8	Poznań	158291.3	96
9	Wrocław	294324.2	91
10	Płock	329171.4	73
11	Katowice	145725.2	64
12	Wrocław	128267.7	60
13	Kraków	179220.2	55
14	Częstochowa	199160.9	44
15	Rybnik	185251.1	42
16	Gliwice	240298.2	35
17	Nadarzyn	169322.4	34
18	Toruń	126283.6	30
19	Sosnowiec	157783.7	30
20	Zielona Góra	177225.2	26
21	Opole	156378.8	22
22	Gorzów Wielkopolski	132687.5	22
23	Olsztyn	211149.9	21
24	Wolica	316630.0	20
25	Bielsko-Biała	152402.1	20

Рисунок 15. Пятый запрос

### Ответы на контрольные вопросы

1. До Python 3.4 работа с путями файловой системы осуществлялась либо с помощью методов строк: `>>> path.rsplit('\\', maxsplit=1)[0]`, либо с помощью модуля `os.path` : `os.path.isfile(os.path.join(os.path.expanduser('~'), 'realpython.txt'))`

2. Что регламентирует PEP 428? - Модуль `pathlib` был введен в Python 3.4 (PEP 428) для решения этих проблем. Он объединяет необходимые функции в одном месте и делает его доступным через методы и свойства простого в использовании объекта `Path`

3. Все, что вам действительно нужно знать, это класс `pathlib.Path` . Есть несколько разных способов создания пути. Прежде всего, существуют classmethods наподобие `cwd()` (текущий рабочий каталог) и `.home()` (домашний каталог вашего пользователя).

4. Свойство `PurePath.parents` представляет собой неизменяемую последовательность, обеспечивающую доступ к логическим предкам пути.

5. Традиционно для чтения или записи файла в Python использовалась



встроенная функция `open()` . Это все еще верно, поскольку функция `open()` может напрямую использовать объекты `Path` .

6. Для простого чтения и записи файлов в библиотеке `pathlib` есть несколько удобных методов:

- `.read_text()` : открыть путь в текстовом режиме и вернуть содержимое в виде строки.

- `.read_bytes()` : открыть путь в двоичном/байтовом режиме и вернуть содержимое в виде

- строки байтов.

- `.write_text()` : открыть путь и записать в него строковые данные.

- `.write_bytes()` : открыть путь в двоичном/байтовом режиме и записать в него данные.

7. Различные части пути удобно доступны как свойства.

Основные примеры включают в себя:

- `.name` : имя файла без какого-либо каталога

- `.parent` : каталог, содержащий файл, или родительский каталог, если путь является

- каталогом

- `.stem` : имя файла без суффикса

- `.suffix` : расширение файла

- `.anchor` : часть пути перед каталогами

8. Через `pathlib` вы также получаете доступ к базовым операциям на уровне файловой системы, таким как перемещение, обновление и даже удаление файлов. По большей части эти методы не выдают предупреждение и не ждут подтверждения, прежде чем информация или файлы будут

потеряны. Будьте осторожны при использовании этих методов. Чтобы переместить файл, используйте `.replace()`. Обратите внимание, что если место назначения уже существует, `.replace()` перезапишет его. К сожалению, `pathlib` явно не поддерживает безопасное перемещение файлов.

9. Есть несколько разных способов перечислить много файлов. Самым простым является метод `.iterdir()`, который перебирает все файлы в данном каталоге. Комбинируется `.iterdir()` с классом `collection.Counter` для подсчета количества файлов каждого типа в текущем каталоге.

10. Определяется функция `tree()`, которая будет печатать визуальное дерево, представляющее иерархию файлов, с корнем в данном каталоге. Здесь мы также хотим перечислить подкаталоги, поэтому мы используем метод `.rglob()`.

11. Последний пример покажет, как создать уникальное нумерованное имя файла на основе шаблона. Сначала укажите шаблон для имени файла с местом для счетчика. Затем проверьте существование пути к файлу, созданного путем соединения каталога и имени файла (со значением счетчика).

12. Ранее мы отмечали, что когда мы создавали экземпляр `pathlib.Path`, возвращался либо объект `WindowsPath`, либо `PosixPath`. Тип объекта будет зависеть от операционной системы, которую вы используете. Эта функция позволяет довольно легко писать кроссплатформенный код. Можно явно запросить `WindowsPath` или `PosixPath`, но вы будете ограничивать свой код только этой системой без каких-либо преимуществ.

**Вывод:** в результате выполнения работы были приобретены навыки по работе с базовыми возможностями системы управления базами данных SQLite3.