

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

Институт цифрового развития

Кафедра инфо коммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.20**  
**Дисциплины «Программирование на Python»**

Выполнил:

Волошин Алексей Вадимович  
1 курс, группа ИТС-6-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., канд. техн. наук, доцент кафедры  
инфокоммуникаций

---

(подпись)

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

Ставрополь, 2023 г.

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

7 Создадим таблицу и отобразим её схему(рис. 1)

```
Last login: Wed Feb 16 07:47:11 2022 from 127.0.0.1
SQLite version 3.34.0 2020-12-01 16:14:00
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 1 - Схема

8 Если ее включить команду timer on, в результатах запроса добавится нужная строчка(рис. 2).

```
Last login: Wed Feb 16 13:04:59 2022 from 127.0.0.1
SQLite version 3.34.0 2020-12-01 16:14:00
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .timer on
sqlite> select count(*) from city;
Run Time: real 0.000 user 0.000066 sys 0.000033
Error: no such table: city
sqlite> █
```

Рисунок 2 – Результат запроса

9 Выполним запрос и получим ответ (рис. 3).

```
Last login: Wed Feb 16 13:11:45 2022 from 127.0.0.1
SQLite version 3.34.0 2020-12-01 16:14:00
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 max(length(city)) from city;
25
sqlite> █
```

Рисунок 3 – Ответ на запрос

10 Загрузите файл city.csv в песочнице с помощью команды .import , но без использования опции --csv . Сделаем .mode (рис. 4).

```
Last login: Wed Feb 16 13:28:18 2022 from 127.0.0.1
SQLite version 3.34.0 2020-12-01 16:14:00
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 4 – Результат выполнения

11 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;
```

И получим ответ, изображенный на рисунке 5.

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

Рисунок 5 – Результат выполнения

12. 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.

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;
```

distance	city
0.00105299999999886	Заречный
0.0094843000000004	Каменка
0.01199310000000051	Елизovo

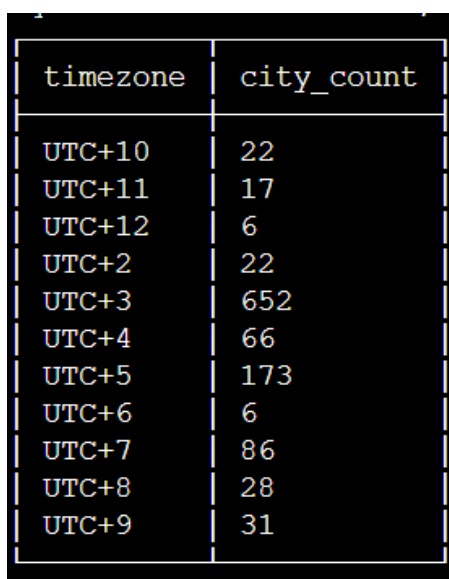
Рисунок 6 – Результат выполнения

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

Выполним следующий запрос:

```
SELECT timezone, count(Distinct city) AS city_count FROM city GROUP BY timezone ORDER BY city_count DESC;
```

Получим результат, изображенный на рисунке 7.



timezone	city_count
UTC+10	22
UTC+11	17
UTC+12	6
UTC+2	22
UTC+3	652
UTC+4	66
UTC+5	173
UTC+6	6
UTC+7	86
UTC+8	28
UTC+9	31

Рисунок 7 – Результат выполнения

Выполним тоже самое в формате CSV (рис. 8) и с другим разделителем (рис. 7).

```
timezone|city_count
UTC+10|22
UTC+11|17
UTC+12|6
UTC+2|22
UTC+3|652
UTC+4|66
UTC+5|173
UTC+6|6
UTC+7|86
UTC+8|28
UTC+9|31
```

Рисунок 8 – Разделитель pipe

```
timezone,city_count
UTC+10,22
UTC+11,17
UTC+12,6
UTC+2,22
UTC+3,652
UTC+4,66
UTC+5,173
UTC+6,6
UTC+7,86
UTC+8,28
UTC+9,31
```

Рисунок 9 – Формат CSV с заголовками

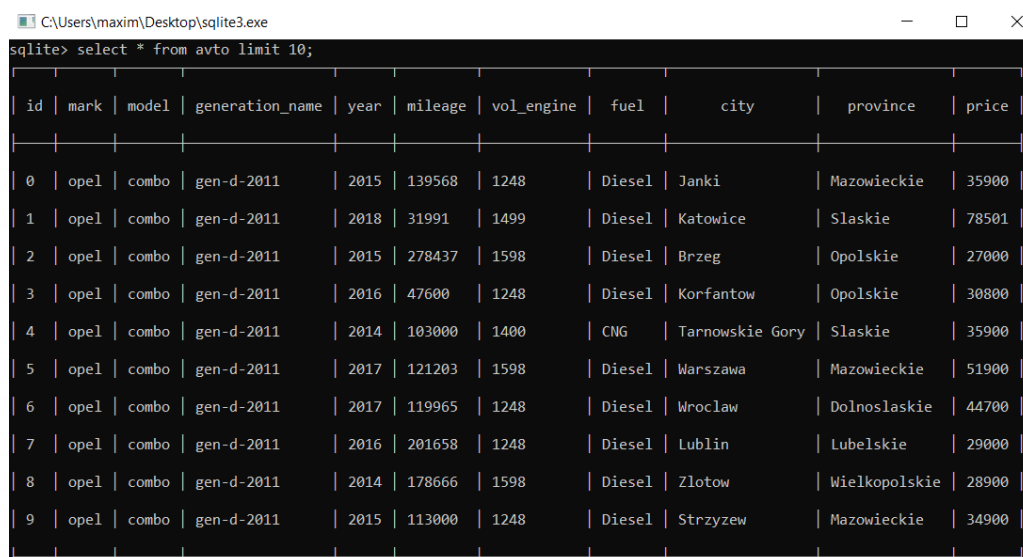
### Индивидуальное задание

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.

Загрузим выбранную таблицу – легковые машины проданные в Польше за последние 10 лет (рис. 9).

БД имеет следующие поля:

1. Id – уникальный номер, в данном случае просто число на 1 больше предыдущего;
2. Mark – марка машины;
3. Model – машины;
4. Generation\_name – поколение машины (год выпуска модели);
5. Year – год продажи машины;
6. Milleage – пробег;
7. Vol\_engine – объем двигателя;
8. Fuel – тип топлива;
9. City – город продажи;
10. Province – область продажи;
11. Price – цена машины в долларах.



```
sqlite> select * from avto limit 10;
```

id	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	Slaskie	78501
2	opel	combo	gen-d-2011	2015	278437	1598	Diesel	Brzeg	Opolskie	27000
3	opel	combo	gen-d-2011	2016	47600	1248	Diesel	Korfantow	Opolskie	30800
4	opel	combo	gen-d-2011	2014	103000	1400	CNG	Tarnowskie Gory	Slaskie	35900
5	opel	combo	gen-d-2011	2017	121203	1598	Diesel	Warszawa	Mazowieckie	51900
6	opel	combo	gen-d-2011	2017	119965	1248	Diesel	Wroclaw	Dolnoslaskie	44700
7	opel	combo	gen-d-2011	2016	201658	1248	Diesel	Lublin	Lubelskie	29000
8	opel	combo	gen-d-2011	2014	178666	1598	Diesel	Zlotow	Wielkopolskie	28900
9	opel	combo	gen-d-2011	2015	113000	1248	Diesel	Strzyzew	Mazowieckie	34900

Рисунок 10 – База данных с заголовками

1 запрос: выбрать машины, проданные в 2015 году.

Запрос: select model, mark from avto where year = 2015;(рис. 10).

```
sqlite> select model,mark from avto where year = 2015;
```

model	mark
combo	opel
combo	opel
combo	opel
combo	opel
combo	opel
combo	opel
combo	opel
combo	opel
combo	opel
combo	opel
combo	opel
combo	opel
combo	opel

Рисунок 11 – Результат выполнения

2 запрос: вычислить средний пробег машин на дизельном топливе.

Запрос: `select avg(mileage) as AverageMileage from avto where fuel = "Diesel";`(рис. 11).

```
sqlite> select avg(mileage) as AverageMileage from avto where fuel = "Diesel";
[{"AverageMileage":175834.4372060401074}]
```

Рисунок 12 – Результат выполнения

3 запрос: выбор и подсчет моделей машин марки Opel, сгруппированный по модели.

Запрос: `select count(model)as count,model from avto where mark = 'opel' Group by 2;`(рис. 12).



```

sqlite> select count(distinct model)as count,model from avto where mark = 'opel' Group by
sqlite> select count(model)as count,model from avto where mark = 'opel' Group by 2;
count,model
96,adam
84,agila
10,ampera
160,antara
3331,astra
224,combo
1549,corse
224,crossland-x
18,frontera
128,grandland-x
1664,insignia
27,karl
896,meriva
896,mokka
20,omega
96,signum
64,tigra
827,vectra
320,vivaro
1280,zafira

```

Рисунок 13 – Результат выполнения запроса

4 запрос: определить максимальную и минимальную стоимость автомобилей марки Audi.

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

```

[{"Average mileage":175854.4572660401074}]
sqlite> select min(price), max(price) from avto where mark = 'audi';
[{"min(price)": "10000", "max(price)": "99999"}]

```

Рисунок 14 – Результат выполнения работы

5 запрос: выбрать и посчитать количество, выпущенных моделей машин на бензине с 2016 по 2018 год, отсортировав по марке.

Запрос: select count(model), mark from avto where year Between 2016 and 2018 and fuel = 'Gasoline' Group by 2;(рис. 14)

count(model)	mark
55	alfa-romeo
640	audi
669	bmw
40	chevrolet
150	citroen
356	fiat
841	ford
193	honda
444	hyundai
353	kia
562	mazda
784	mercedes-benz
162	mini
141	mitsubishi
422	nissan
1005	opel
338	peugeot
690	renault
361	seat
1054	skoda
817	toyota
1090	volkswagen
410	volvo

qlite>

Рисунок 15 – Результат

6 запрос: посчитать количество и среднюю стоимость машины в каждом городе, учитывая машины проданные за последние три года.

Запрос: `select city, Round(avg(price),1), count(*) as count from avto where year between date('now', '-3 years') and date('now') Group by 1 Order by Count desc;`(рис. 15).

city	Round(avg(price),1)	count
Warszawa	195728.7	1802
Lodz	193555.9	1286
Chorzow	183215.8	975
Gdansk	228220.2	764
Wroclaw	211555.2	654
Krakow	190225.1	651
Poznan	220130.7	571
Katowice	189777.0	562
Gdynia	194783.4	408
Torun	205942.5	356
Bydgoszcz	190675.0	323
Szczecin	173739.9	307
Sosnowiec	157982.5	297
Czestochowa	248051.9	283
Opole	201712.9	275
Kalisz	221199.4	259

Рисунок 16 – Результат выполнения

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

### 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. Такой конкретный путь не может быть использован в другой системе: