

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»
Вариант №2

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Основы работы с SQLite3

Цель: Исследовать базовые возможности системы управления базами данных SQLite3.

Ход работы

7. Решите задачу: выполните [в песочнице](#) команды:

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

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

Рисунок 1. Результат выполнения запроса

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 строчка:

```
Run Time: real xxx user xxx sys xxx
```

Например:

```
sqlite> .SOMETHING on  
sqlite> select count(*) from city;  
1117  
Run Time: real 0.000 user 0.000106 sys 0.000069
```

```
sqlite> .timer on  
sqlite> select count(*) from city;  
Run Time: real 0.001 user 0.000050 sys 0.000151  
Parse error: no such table: city
```

Рисунок 2. Результат выполнения запроса

9. Решите задачу: загрузите файл city.csv [в песочнице](#):

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

Затем выполните такой запрос:

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

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

Рисунок 3. Результат выполнения запроса

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.

Вам поможет команда `.help import`. Всего должно получиться две команды:

```
do_something
.import city.csv city
```

Какая команда должна быть вместо `do_something`?

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

Рисунок 3. Запросы

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:

timezone	city_count
UTC+3	xxx
UTC+4	xx
UTC+5	xx
UTC+6	x
UTC+7	xx
UTC+8	xx

Укажите в ответе значение `city_count` для `timezone = UTC+5`.

```
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 5. Результат выполнения запроса

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.
- Например:

нижний Новгород, Москва, Владивосток

Чтобы посчитать расстояние между двумя городами, используйте формулу из школьного курса геометрии:

$$distance^2 = (lat_1 - lat_2)^2 + (lon_1 - lon_2)^2 \quad (1)$$

Где (lat_1, lon_1) — координаты первого города, а (lat_2, lon_2) — координаты второго.

```
sqlite> select city,
...> sqrt(power(geo_lat - sam_lat, 2) + power(geo_lon - sam_lon, 2)) as distance from city
...> cross join (select geo_lat as sam_lat, geo_lon as sam_lon from city where city = 'Самара')
...> as samara
...> where city != 'Самара' order by distance limit 3;
Новокуйбышевск|0.18569700863441
Чапаевск|0.358068603404667
Кинель|0.528066220190501
```

Рисунок 6. Результат выполнения запроса

13. Решите задачу: напишите [в песочнице](#) запрос, который посчитает количество городов в каждом часовом поясе. Отсортируйте по количеству городов по убыванию. Получится примерно так:

timezone	city_count
UTC+3	xxx
UTC+5	xxx
UTC+7	xxx
UTC+4	xxx
...	

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

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

```
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
```

Рисунок 7. Результат выполнения запроса

Индивидуальное задание

Загрузите в SQLite выбранный Вами датасет в формате CSV (датасет можно найти на сайте [Kaggle](https://www.kaggle.com/)). Сформируйте более пяти запросов к таблицам БД. Выгрузите результат выполнения запросов в форматы CSV и JSON.

Табличный вид		Форма											
				Отфильтровать...									
				Всего загружено строк: 200									
	Invoice_ID	Branch	City	Customer_	Gender	Product_line	Unit_price	Quantity	Tax	Total			
1	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715			
2	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.82	80.22			
3	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255			
4	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.288	489.048			
5	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785			
6	699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165			
7	355-53-5943	A	Yangon	Member	Female	Electronic accessories	68.84	6	20.652	433.692			
8	315-22-5665	C	Naypyitaw	Normal	Female	Home and lifestyle	73.56	10	36.78	772.38			
9	665-32-9167	A	Yangon	Member	Female	Health and beauty	36.26	2	3.626	76.146			
10	692-92-5582	B	Mandalay	Member	Female	Food and beverages	54.84	3	8.226	172.746			
11	351-62-0822	B	Mandalay	Member	Female	Fashion accessories	14.48	4	2.896	60.816			
12	529-56-3974	B	Mandalay	Member	Male	Electronic accessories	25.51	4	5.102	107.142			
13	365-64-0515	A	Yangon	Normal	Female	Electronic accessories	46.95	5	11.7375	246.4875			
14	252-56-2699	A	Yangon	Normal	Male	Food and beverages	43.19	10	21.595	453.495			
15	829-34-3910	A	Yangon	Normal	Female	Health and beauty	71.38	10	35.69	749.49			
16	299-46-1805	B	Mandalay	Member	Female	Sports and travel	93.72	6	28.116	590.436			
17	656-95-9349	A	Yangon	Member	Female	Health and beauty	68.93	7	24.1255	506.6355			
18	765-26-6951	A	Yangon	Normal	Male	Sports and travel	72.61	6	21.783	457.443			
19	329-62-1586	A	Yangon	Normal	Male	Food and beverages	54.67	3	8.2005	172.2105			
20	319-50-3348	B	Mandalay	Normal	Female	Home and lifestyle	40.3	2	4.03	84.63			
21	300-71-4605	C	Naypyitaw	Member	Male	Electronic accessories	86.04	5	21.51	451.71			
22	371-85-5789	B	Mandalay	Normal	Male	Health and beauty	87.98	3	13.197	277.137			

Рисунок 8. Загруженные данные

Вывод различающихся значений столбца City:

Запрос	
1	select distinct city from supermarket_sales;
2	
Табличный вид	
Форма	
City	
1	Yangon
2	Naypyitaw
3	Mandalay

Рисунок 9. Запрос и результат его выполнения

Сумма общего дохода для каждого города:

```
3 select City, sum(Total) as Total_Income
4 from supermarket_sales
5 group by City;
```

Табличный вид		Форма
		1
	City	Total_Income
1	Mandalay	20744.702999999998
2	Naypyitaw	27934.746000000003
3	Yangon	24575.207999999995

Рисунок 10. Запрос и результат его выполнения

Выбор всех строк таблицы, где значение Total превышает 800:

```
7 select * from supermarket_sales where Total > 800;
```

Табличный вид

Форма

↺

☑

✕

⏪

⏩

1

⏴

⏵

🖨

Всего загружено строк: 12

	Invoice_ID	Branch	City	Customer_	Gender	Product_line	Unit_price	Quantity	Tax	Total
1	228-96-1411	C	Naypyitaw	Member	Female	Food and beverages	98.7	8	39.48	829.08
2	574-22-5561	C	Naypyitaw	Member	Female	Fashion accessories	82.63	10	41.315	867.615
3	232-11-3025	A	Yangon	Normal	Male	Sports and travel	78.77	10	39.385	827.085
4	393-65-2792	C	Naypyitaw	Normal	Male	Food and beverages	89.48	10	44.74	939.54
5	829-49-1914	C	Naypyitaw	Member	Female	Food and beverages	78.31	10	39.155	822.255
6	766-85-7061	B	Mandalay	Normal	Male	Health and beauty	87.87	10	43.935	922.635
7	228-96-1411	C	Naypyitaw	Member	Female	Food and beverages	98.7	8	39.48	829.08
8	574-22-5561	C	Naypyitaw	Member	Female	Fashion accessories	82.63	10	41.315	867.615
9	232-11-3025	A	Yangon	Normal	Male	Sports and travel	78.77	10	39.385	827.085
10	393-65-2792	C	Naypyitaw	Normal	Male	Food and beverages	89.48	10	44.74	939.54
11	829-49-1914	C	Naypyitaw	Member	Female	Food and beverages	78.31	10	39.155	822.255
12	766-85-7061	B	Mandalay	Normal	Male	Health and beauty	87.87	10	43.935	922.635

Рисунок 11. Запрос и результат его выполнения

Средняя цена продукта по каждой линии продукта:

```

9 select Product_line, avg(Unit_price) as Avg_Unit_Price
10 from supermarket_sales
11 group by Product_line;

```










Табличный вид		Форма							
									Всего загруже
	Product_line	Avg_Unit_Price							
1	Electronic accessories	55.00625000000001							
2	Fashion accessories	54.25249999999999							
3	Food and beverages	65.28882352941177							
4	Health and beauty	59.88454545454545							
5	Home and lifestyle	54.15066666666667							
6	Sports and travel	62.14722222222222							

Рисунок 12. Запрос и результат его выполнения

Количество продаж и сумма налога для каждого филиала:

```

13 select Branch, count(Invoice_ID) as Total_Sales, sum(Tax) as Total_Tax
14 from supermarket_sales
15 group by Branch;

```

Табличный вид		Форма
		Всего загружено строк: 3
Branch	Total_Sales	Total_Tax
1 A	70	1170.6279999999997
2 B	60	988.223
3 C	70	1330.2259999999997

Рисунок 13. Запрос и результат его выполнения

Общее количество проданных товаров по типу клиента:






```

17 select Customer_type, sum(Quantity) as Total_Quantity
18 from supermarket_sales
19 group by Customer_type;




```

Табличный вид

Форма



1



Всего загруз

	Customer_type	Total_Quantity
1	Member	572
2	Normal	592

Рисунок 14. Запрос и результат его выполнения

Контрольные вопросы:

1. Реляционные базы данных и системы управления базами данных (СУБД) используются для хранения, управления и обработки структурированных данных в виде таблиц, связанных друг с другом.

2. Язык SQL (Structured Query Language) используется для работы с реляционными базами данных. Он позволяет выполнять операции создания, чтения, обновления и удаления данных, а также управлять структурой базы данных.

3. Язык SQL состоит из подмножества команд, таких как SELECT (для выборки данных), INSERT (для вставки данных), UPDATE (для обновления данных), DELETE (для удаления данных), CREATE (для создания объектов), ALTER (для изменения структуры объектов), и других.

4. СУБД SQLite - это встраиваемая база данных, которая работает локально на устройстве без необходимости сервера. Клиент-серверные СУБД, например MySQL или PostgreSQL, требуют наличия сервера для обработки запросов к базе данных.

5. Установка SQLite в Windows и Linux обычно осуществляется путем загрузки исполняемого файла или установочного пакета с официального сайта SQLite и последующего выполнения инструкций по установке.

6. Для создания базы данных SQLite используется команда `CREATE DATABASE <database_name>;`.

7. Для выяснения текущей базы данных в SQLite можно использовать команду `.database` в командной строке SQLite.

8. Для создания таблицы в SQLite используется команда `CREATE TABLE <table_name> (<column1_name> <data_type>, <column2_name> <data_type>, ...);`, а для удаления - команда `DROP TABLE <table_name>;`.

9. Первичный ключ в таблице обычно является уникальным столбцом, который однозначно идентифицирует каждую запись в таблице.

10. Для создания автоинкрементного первичного ключа в SQLite используется ключевое слово `AUTOINCREMENT` при определении столбца.

11. Инструкция NOT NULL указывает, что значение в столбце не может быть NULL, а DEFAULT позволяет задать значение по умолчанию для столбца.

12. Внешние ключи используются для установления связей между таблицами. Для создания внешнего ключа в SQLite используется команда
FOREIGN KEY (<column_name>) REFERENCES
<other_table>(<other_column>)

13. Для вставки строки в таблицу базы данных SQLite используется команда INSERT INTO <table_name> (<column1>, <column2>, ...) VALUES (<value1>, <value2>, ...);

14. Для выборки данных из таблицы SQLite используется команда SELECT <column1>, <column2>, ... FROM <table_name>;

15. Для ограничения выборки данных с помощью условия WHERE используется команда SELECT <columns> FROM <table_name> WHERE <condition>;

16. Для упорядочивания выбранных данных используется команда SELECT <columns> FROM <table_name> ORDER BY <column> [ASC|DESC];

17. Для обновления записей в таблице SQLite используется команда UPDATE <table_name> SET <column1>=<value1>, <column2>=<value2> WHERE <condition>;

18. Для удаления записей из таблицы SQLite используется команда DELETE FROM <table_name> WHERE <condition>;

19. Для группировки данных из выборки из таблицы SQLite используется команда SELECT <columns>, <aggregation_function>(<column>) FROM <table_name> GROUP BY <column>;

20. Для получения значения агрегатной функции в выборке из таблицы SQLite используется команда SELECT <aggregation_function>(<column>) FROM <table_name>;

21. Для объединения нескольких таблиц в операторе SELECT в SQLite используются различные типы JOIN (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN).

22. Подзапросы и шаблоны используются для выполнения сложных запросов, включая вложенные запросы, а также повторяющиеся фрагменты запросов.

23. Представления VIEW в SQLite позволяют создавать виртуальные таблицы на основе запросов, что упрощает доступ к данным.

24. Для импорта данных в SQLite можно использовать инструменты импорта CSV, SQL-скрипты или специализированные программы для конвертации данных.

25. Команда .schema в SQLite используется для отображения схемы базы данных, включая определения таблиц, индексы и другие объекты.

26. Группировка и сортировка данных в запросах SQLite выполняется с помощью команд GROUP BY и ORDER BY соответственно.

27. Табличные выражения в SQLite позволяют создавать временные таблицы для выполнения сложных запросов.

28. Для экспорта данных из SQLite в форматы CSV и JSON можно использовать инструменты командной строки или сторонние программы.

29. Другие форматы для экспорта данных могут включать XML, Excel, SQL-скрипты и другие текстовые форматы.

Вывод: В ходе выполнения лабораторной работы были исследованы базовые возможности системы управления базами данных SQLite3.

