РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Кафедра инфокоммуникаций «Основы работы с DuckDB»

Отчет по лабораторной работе по дисциплине «Программирование на Python»

Выполнил студент группы ИВТ-б-о-2	21-1	
<u>Толубаев Рамиль Ахметович</u>		
«20» <u>мая</u> 20 <u>23</u> г.		
Подпись студента		
Работа защищена « »	_20_	_г.
Проверил Воронкин Р.А		

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

Порядок выполнения работы:

1. Решить задачи:

Задание 1. Выполните команды. Что вернула команда .schema?

```
D create table customer(name string);
D select * from customer;

name
varchar
0 rows

D .schema customer
CREATE TABLE customer("name" VARCHAR);;
```

Рисунок 1 - Результат выполнения задания 1

Команда .schema customer вывела запрос CREATE, соответствующий таблине customer.

Задание 2. С помощью команды .help найдите команду, которая отвечает за вывод времени выполнения запроса.

```
D .timer on
D select count(*) from read_csv_auto('./city.csv');

count_star()
    int64

1117

Run Time (s): real 0.078 user 0.046875 sys 0.000000
```

Рисунок 5 - Результат выполнения задания 2

Задание 3. Загрузить файл city.csv. Выполните запрос. Какое число он вернул?

```
D select max(length(city)) from read_csv_auto('./city.csv');

max(length(city))
    int64

25
```

Рисунок 6 - Результат выполнения задания 3

Задание 4. загрузите файл city.csv, но без использования опции --csv.

```
D create table city as select * from read_csv_auto('./city.csv');
D
```

Рисунок 7 - Результат выполнения задания 4

Задание 5. Напишите 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.

```
D select timezone, count(city) as city_count from read_csv_auto('./city.csv')
 where federal_district in ('Приволжский', 'Сибирский')
 group by timezone
 order by timezone ASC;
             city_count
  timezone
               int64
  varchar
 UTC+3
                    101
 UTC+4
                     41
 UTC+5
                     58
 UTC+6
                      6
  UTC+7
                     86
  UTC+8
                      22
```

Рисунок 8 - Результат выполнения задания 5

Задание 6. Напишите запрос, который посчитает количество городов в каждом часовом поясе. Отсортируйте по количеству городов по убыванию.

```
create table city as select * from read_csv_auto('./city.csv');
select
timezone, count(city) as city_count from city
group by timezone
order by city_count DESC;
timezone
           city_count
varchar
             int64
UTC+3
                   656
UTC+5
                   173
UTC+7
                    86
UTC+4
                    66
UTC+9
                    31
UTC+8
                    28
UTC+10
                    22
UTC+2
                    22
UTC+11
                    17
UTC+12
                     6
UTC+6
                     6
11 rows
            2 columns
```

Рисунок 9 - Результат выполнения задания 6

2. Выполните индивидуальное задание. Каждый запрос к базе данных сохраните в файл с расширением sql.

Загрузите в SQLite выбранный Вами датасет в формате CSV. Сформируйте более пяти запросов к таблицам БД. Выгрузите результат выполнения запросов в форматы CSV и JSON.

Запрос 1.

```
create table covid as select * from read_csv_auto('./covid.csv');
SELECT Country, CasesLast7d
FROM covid
WHERE CasesLast7d BETWEEN 2500 AND 3000;
           CasesLast7d
Country
varchar
              int64
Colombia
                  2512
copy(
SELECT Country, CasesLast7d
FROM covid
WHERE CasesLast7d BETWEEN 2500 AND 3000) to 'req1.json';
SELECT Country, CasesLast7d
FROM covid
WHERE CasesLast7d BETWEEN 2500 AND 3000) to 'req1.csv' (header, delimiter ',');
```

Рисунок 10 - Результат выполнения запроса 1 индивидуального задания Запрос 2.

Рисунок 11 - Результат выполнения запроса 2 индивидуального задания

Запрос 3.

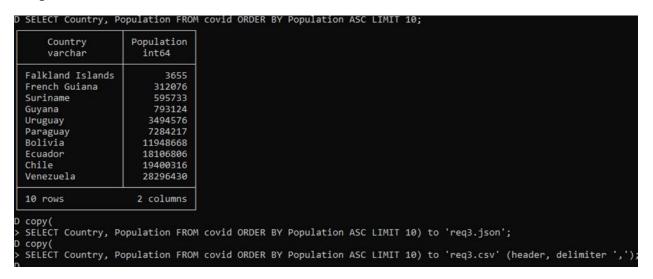


Рисунок 12 - Результат выполнения запроса 3 индивидуального задания Запрос 4.



Рисунок 13 - Результат выполнения запроса 4 индивидуального задания

Запрос 5.

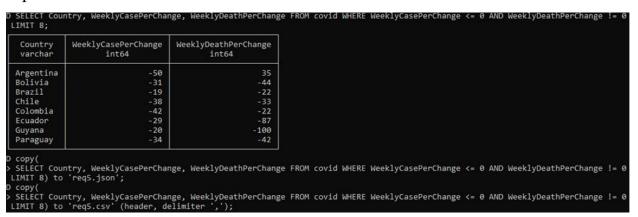


Рисунок 14 - Результат выполнения запроса 5 индивидуального задания Запрос 6

int64	CasesPreceding7d int64	Population int64								
48	4312	51821865								
119	4901	33766958								
16	1260	28296430								
181	4629	18106806								
133	2312	11948668								
111	1216	7284217								
50	50	793124								
	119 16 181 133 111	119 4901 16 1260 181 4629 133 2312 111 1216	119 4901 33766958 16 1260 28296430 181 4629 18106806 133 2312 11948668 111 1216 7284217	119 4901 33766958 16 1260 28296430 181 4629 18106806 133 2312 11948668 111 1216 7284217	119 4901 33766958 16 1260 28296430 181 4629 18106806 133 2312 11948668 111 1216 7284217	119 4991 33766958 16 1260 28296430 181 4629 18106806 133 2312 11948668 111 1216 7284217	119 4901 33766958 16 1260 28296430 181 4629 18106806 133 2312 11948668 111 1216 7284217	119 4991 33766958 16 1260 28296430 181 4629 18106806 133 2312 11948668 111 1216 7284217	119 4901 33766958 16 1260 28296430 181 4629 18106806 133 2312 11948668 111 1216 7284217	119 4901 33766958 16 1260 28296430 181 4629 18106806 133 2312 11948668 111 1216 7284217

Рисунок 15 - Результат выполнения запроса 6 индивидуального задания

Вывод: были исследованы базовые возможности системы управления базами данных DuckDB.