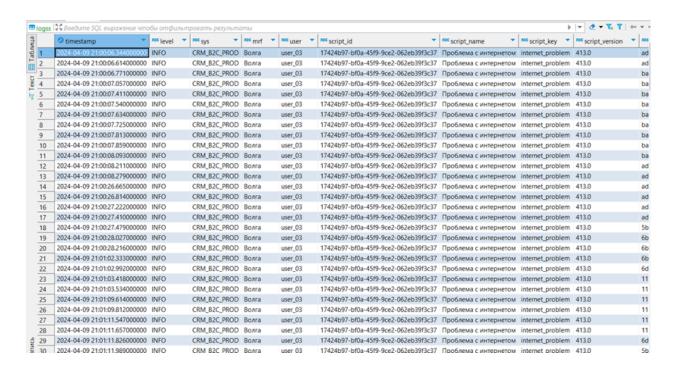
## **№**2

Вначале у нас есть данные, которые были загружены в ClickHouse на первом этапе.



С помощью Python подключимся к базе данных и извлечем parameters и script\_id из Clickhouse.

```
from clickhouse_driver import Client
import pandas as pd
import json

client = Client(host='localhost', port=9000, user='default', pas

query = 'SELECT parameters, script_id from db_test.logss'
result = client.execute(query)

df = pd.DataFrame(result, columns=['parameters', 'script_id'])
```

```
client = Client(host='localhost', port=9000, user='default', password='pivanet')
query = 'SELECT parameters, script_id from db_test.logss'
result = client.execute(query)

df = pd.DataFrame(result, columns=['parameters', 'script_id'])
```

Мы получили следующий датафрейм.

script_id	parameters	
17424b97-bf0a-45f9-9ce2-062eb39f3c37	0	0
17424b97-bf0a-45f9-9ce2-062eb39f3c37		1
17424b97-bf0a-45f9-9ce2-062eb39f3c37	{"Type_equipment": ""}	2
17424b97-bf0a-45f9-9ce2-062eb39f3c37		3
17424b97-bf0a-45f9-9ce2-062eb39f3c37	{"TYPE_EQUIPMENT": "оборудованием"}	4
a67dfe4b-a531-4366-9d7c-cfa348256b20	{"SERVICE_STATUS_LOCAL": "Включена"}	755
a67dfe4b-a531-4366-9d7c-cfa348256b20		756
a67dfe4b-a531-4366-9d7c-cfa348256b20	{"SERVICE_ID": "160502", "ACCOUNT_NUMBER": "ac	757
a67dfe4b-a531-4366-9d7c-cfa348256b20		758
a67dfe4b-a531-4366-9d7c-cfa348256b20	0	759

Согласно тексту задания, нам необходимо создать таблицу с группировкой по script\_id. Эта таблица должна содержать данные, подходящие для обработки специалистами по ВІ. Таблица должна быть одна.

Можно заметить, что для script\_id присутствуют одинаковые строки. Давайте посмотрим, какие уникальные значения содержатся в этом столбце:

```
df['script_id'].value_counts()
```

```
script_id

7e3cfde7-53a7-40a9-b814-c373df9d8d04 501

a67dfe4b-a531-4366-9d7c-cfa348256b20 160

17424b97-bf0a-45f9-9ce2-062eb39f3c37 99

Name: count, dtype: int64
```

Всего есть три script\_id. Сделаем группировку.

```
def merge_dicts(dicts):
    result_dict = {}
    for dictionary in dicts:
        if isinstance(dictionary, dict): # Убедимся, что элемен for key, value in dictionary.items():
            result_dict[key] = value # Обновление значения return result_dict
import pandas as pd

# Группировка по script_id и агрегация словарей в parameters aggregated_df = df.groupby('script_id')['parameters'].agg(merge_aggregated_df
```

После группировки датафрейм выглядит следующим образом:

	script_id	parameters
0	17424b97-bf0a-45f9-9ce2-062eb39f3c37	{'TYPE_EQUIPMENT': 'оборудованием', 'CATALOG':
1	7e3cfde7-53a7-40a9-b814-c373df9d8d04	{'GET_CLIENT_TIMEZONE': '+03:00', 'EMPLOYEE_MR
2	a67dfe4b-a531-4366-9d7c-cfa348256b20	{'COMMUNICATION_NUMBER': '000000344206174', 'C

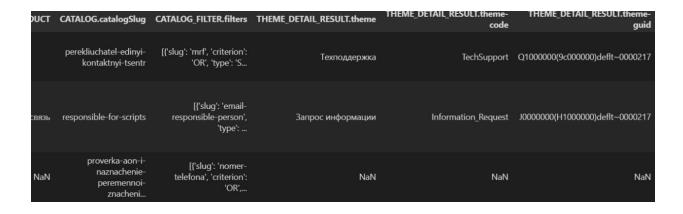
То есть у нас есть script\_id и параметры. Параметры представляют из себя словари со значениями. Нужно превратить колонку параметров в несколько

колонок. Сделать таблицу шире. По факту задача похожа на парсинг JSONфайлов.

Попробуем раскрыть словари исходного датафрейма на один уровень. Вот результат:



Я заметил, что после раскрытия словарей на один уровень остаются еще словари. А также списки. В списках обычно содержатся словари. Нужно написать функцию, которая будет превращаться ключи словарей в новые столбцы. Вот как выглядят словари в значениях:



Напишем функцию, которая проходится по колонкам датафрейма. Если значением ячейки является лист, и внутри листа лежит словарь, то мы достаем словарь Если находим словарь,

Напишем функцию, которая проходится по колонкам датафрейма. Если значением ячейки является лист, и внутри листа лежит словарь, то мы достаём этот словарь. Если находим словарь, мы далее анализируем его содержимое, чтобы определить, можно ли его развернуть в отдельные

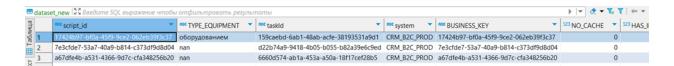
столбцы для улучшения структуры данных. Этот подход позволяет преобразовывать вложенные структуры данных в более плоский и анализируемый формат.

Применяя разработанные функции, мы последовательно идентифицируем столбцы, содержащие словари (find\_dict\_columns) и списки (find\_list\_columns). Это первый шаг в процессе структурирования данных. Для столбцов, содержащих словари, используется функция expand\_dict\_columns, которая преобразует каждый словарь в отдельные столбцы. Это позволяет более детально анализировать информацию, которая ранее была скрыта внутри словарей. Функцию можно посмотреть в коде.

После раскрытия мы получаем итоговый датафрейм, в котором 491 колонка. Он не содержит списки и словари в качестве значений.

```
Final result structure:
script id
                                    object
                                    object
TYPE EQUIPMENT
$$$taskId
                                    object
$$$system
                                    object
BUSINESS KEY
                                    object
INTERACTION TOPICS.result.guid
                                    object
CASE TYPES.type1.0
                                   float64
CASE TYPES.type1.title
                                    object
CASE TYPES.type1.id
                                    object
CASE TYPES.type1.guid
                                    object
Length: 491, dtype: object
```

В именах столбцов присутствует недопустимый символ \$. Сделаем предобработку. После этого будем делать вставку в нашу таблицу ClickHouse. После вставки посмотрим данные через DBeaver:



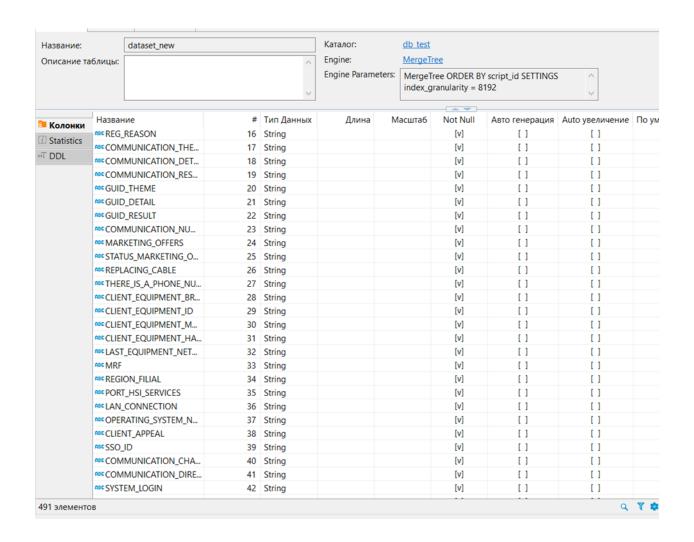


Таблица создана и пригодна к использованию для специалистов по BI. Перейдем к следующей задачке, где запустим скрипт обработки через Airflow.