Задание №2

Давайте обратимся к базе данных ClickHouse для извлечения данных и переноса их в DataFrame. Вот пример кода:

```
from clickhouse driver import Client
import pandas as pd
import json

client = Client(host='localhost', port=9000, user='default', password='pivanet')

query_2 = 'SELECT parameters, timestamp, user, communication_number, communication_id, script_id, script_name, mrf, client_mrf, script_owner, communication_execute(query_2)

df_all = pd.DataFrame(result_2, columns=['parameters', 'timestamp', 'user', 'communication_number', 'communication_id', 'script_id', 'script_name', 'script_id', 'script_name', 'communication_id', 'script_id', 'script_id', 'script_name', 'communication_id', 'script_id', 'script_name', 'communication_id', 'script_id', 'script_name', 'communication_id', 'script_id', 'script_id', 'script_name', 'communication_id', 'script_id', 'script_id
```

Список колонок:

Функция extract_fields конвертирует JSON-строку в словарь Python и извлекает специфические поля. С помощью пр.vectorize создается векторизованная версия функции. Это позволяет применить функцию ко всему массиву данных одновременно, ускоряя выполнение. Эта векторизованная функция применяется к столбцу parameters DataFrame. Результат, массив словарей, преобразуется в DataFrame. Столбец parameters удаляется из исходного DataFrame, а полученный DataFrame добавляется в него. Затем выводятся первые строки обновленного DataFrame для проверки.

Вот сравнение по скорости применения .apply с методом np.vectorize.

apply

pandas apply: 0.010205268859863281

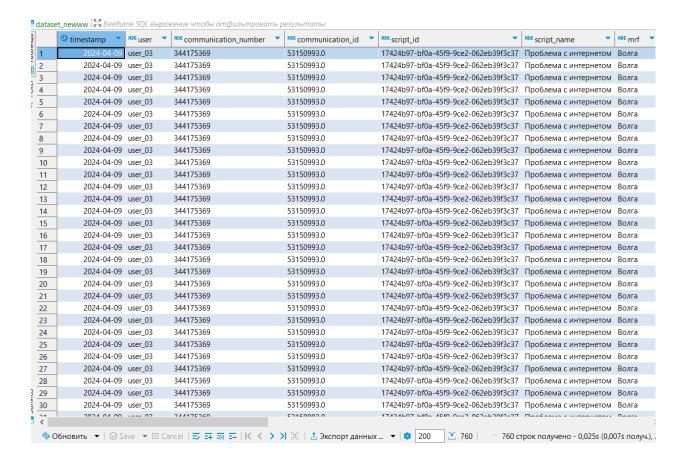
Задание №2

Vectorize time: 0.008959531784057617

Таким образом, np.vectorize работает быстрее

Мы получили итоговый датафрейм со всеми необходимыми колонками.

Затем мы создали таблицу dataset_neww с нашими колонками и загрузили в неё наш датафрейм.



Задание №2

Задание №2