

Лабораторная работа №9

Цель задания: попрактиковаться с генерацией данных для очереди сообщений и сохранением данных в таблицах в GreenPlum.

а) Используйте Kafka сервер, который доступен на сервере “vm-strmng-s-1.test.local” и по порту 9092. Напишите Python скрипт (используя библиотеку Python **kafka**), который будет генерировать поток данных (ровно одну тысячу сообщений) в формате json и складывать данные в ваш topic (используйте шаблон именования “lab10_фамилия”) очереди сообщения Kafka.

Протокол данных json модели:

```
{
  client: "любое имя клиента",
  opened: "дата открытия заявки в тех поддержку",
  priority: 0 | 1 | 2
}
```

Где **priority** принимает одно значение из 3 возможных

- 0 - низкий приоритет заявки клиента
- 1 - средний приоритет заявки клиента
- 2 - высокий приоритет заявки клиента

Добейтесь в коде Python, чтобы ваш генератор создавал заявки в следующих пропорциях по приоритетам (значение для поля **priority**)

- низкий приоритет - 75 % всех сгенерированных данных
- средний приоритет - 20 % всех сгенерированных данных
- высокий приоритет - 5 % всех сгенерированных данных

Файл dz_9_put.py:

```
"""Генерация потока данных"""

import json
import random
from datetime import datetime, timedelta
from kafka import KafkaProducer

try:
    from faker import Faker
except ImportError:

    class Faker:
        """Заменяет Faker при его недоступности"""

        def __init__(self, *args, **kwargs):
            pass

        @staticmethod
        def last_name():
            return f'user{random.randint(1, 1000):03}'

random.seed(13)
priority_choices = [0] * 15 + [1] * 4 + [2]
```

```

random.shuffle(priority_choices)

producer = KafkaProducer(bootstrap_servers='vm-strmng-s-1.test.local:9092')

fake = Faker()
open_date = datetime.now() - timedelta(hours=10)
num_requests = 1000
for idx in range(num_requests):
    open_date += timedelta(seconds=random.randint(0, 60))

    request = {"client": fake.last_name(),
               "opened": open_date.strftime("%d.%m.%Y %H:%M"),
               "priority": priority_choices[idx % 20]}

    producer.send("lab10_pavlov", json.dumps(request).encode('utf-8'))

producer.close()

```

b) Создайте 3 физических таблицы в GreenPlum (формат: columnar, append only), используя шаблон именования

- “lab10_фамилия_0” (таблица для хранения заявок с низким приоритетом)
- “lab10_фамилия_1” (таблица для хранения заявок со средним приоритетом)
- “lab10_фамилия_2” (таблица для хранения заявок с высоким приоритетом)

Выберите ключ дистрибуции, какой по вашему мнению необходим для предотвращения SKEW аномалии (самый простой - DISTRIBUTED RANDOMLY).

```

CREATE TABLE lab10_pavlov_0 (
    client TEXT,
    opened TIMESTAMP DEFAULT '2000-01-01 00:00:00'
) WITH (
    APPENDONLY = TRUE,
    ORIENTATION = COLUMN
)
DISTRIBUTED RANDOMLY;

CREATE TABLE lab10_pavlov_1 (
    client TEXT,
    opened TIMESTAMP DEFAULT '2000-01-01 00:00:00'
) WITH (
    APPENDONLY = TRUE,
    ORIENTATION = COLUMN
)
DISTRIBUTED RANDOMLY;

CREATE TABLE lab10_pavlov_2 (
    client TEXT,
    opened TIMESTAMP DEFAULT '2000-01-01 00:00:00'
) WITH (
    APPENDONLY = TRUE,
    ORIENTATION = COLUMN
)
DISTRIBUTED RANDOMLY;

```

с) Создайте consumer на языке Python, который будет вычитывать данные из вашей очереди сообщений с шаблоном имени "lab10_фамилия" и раскидать данные по таблицам в зависимости от значения поля **priority** в json данных.

Файл dz_9_get.py:

```
"""Чтение данных из очереди сообщений"""

import json
import pg8000
from datetime import datetime
from configparser import ConfigParser
from kafka import KafkaConsumer

config = ConfigParser()
config.read('greenplum.ini')
base_name = 'GreenPlum'
dbname = config.get(base_name, 'databases')
user = config.get(base_name, 'user_name')
pswd = config.get(base_name, 'user_pass')
host = config.get(base_name, 'host_urls')
port = config.get(base_name, 'host_port')

try:
    conn = pg8000.connect(database=dbname, user=user, password=pswd, host=host,
port=port)
    cursor = conn.cursor()

    consumer = KafkaConsumer(
        "lab10_pavlov",
        bootstrap_servers='vm-strmng-s-1.test.local:9092',
        group_id='my-group',
        # enable_auto_commit=True,
        value_deserializer=lambda x: json.loads(x.decode('utf-8'))
    )

    for message in consumer:
        data = message.value
        client = data.get('client', 'none_client')
        opened = data.get('opened', '01.01.2000 00:00')
        opened = datetime.strptime(opened, '%d.%m.%Y %H:%M').strftime('%Y-%m-%d %H:%M')

        table_name = f"lab10_pavlov_{data.get('priority', 0)}"

        query = "INSERT INTO {} (client, opened) VALUES (%s, %s)".format(table_name)
        cursor.execute(query, (client, opened))
        conn.commit()

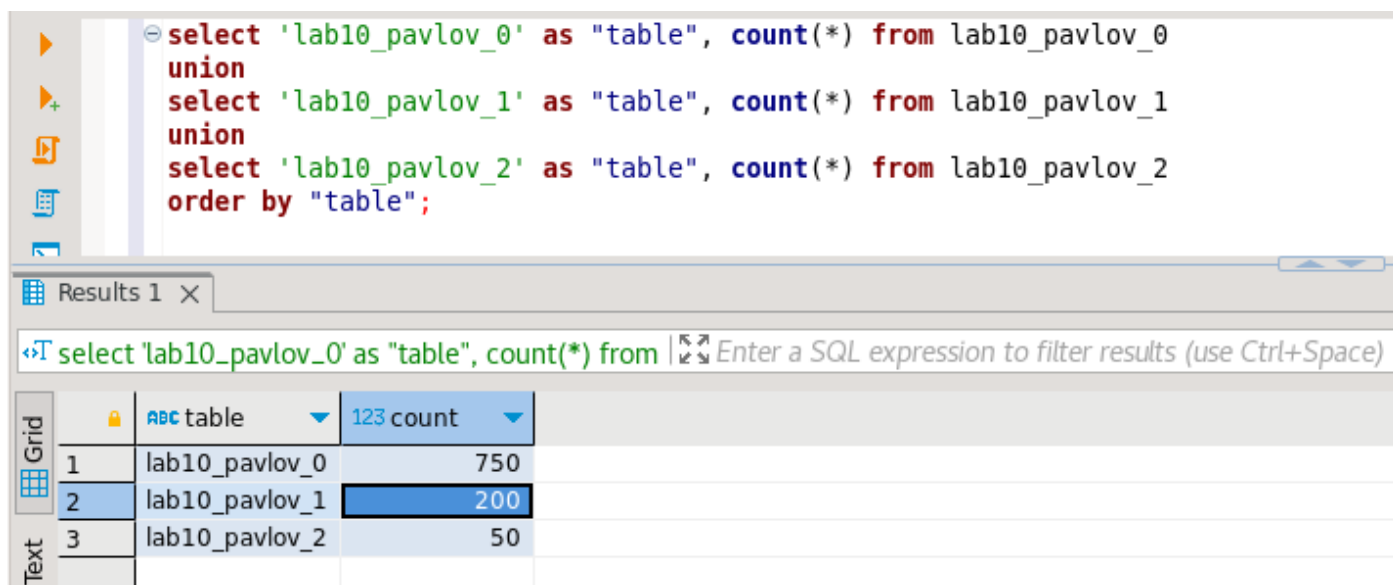
        print(query, (client, opened))

finally:
    consumer.close()
    conn.close()
```

Файл greenplum.ini:

```
[GreenPlum]
host_urls = 192.168.77.21
host_port = 5432
databases = postgres
user_name = user
user_pass = password
```

Результат:



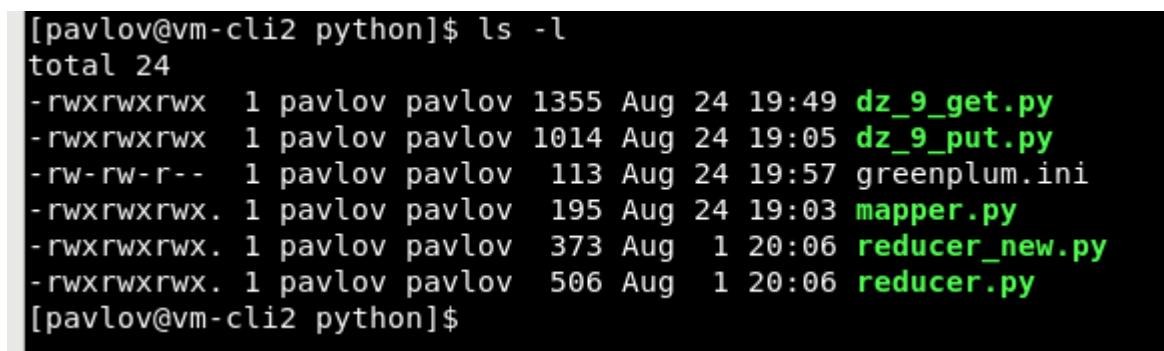
The screenshot shows a SQL query editor with the following query:

```
select 'lab10_pavlov_0' as "table", count(*) from lab10_pavlov_0
union
select 'lab10_pavlov_1' as "table", count(*) from lab10_pavlov_1
union
select 'lab10_pavlov_2' as "table", count(*) from lab10_pavlov_2
order by "table";
```

Below the query, the results are displayed in a table grid. The grid has two columns: 'table' and 'count'. The results are as follows:

	table	count
1	lab10_pavlov_0	750
2	lab10_pavlov_1	200
3	lab10_pavlov_2	50

Файлы на стенде в домашнем каталоге:



```
[pavlov@vm-cli2 python]$ ls -l
total 24
-rwxrwxrwx  1 pavlov pavlov 1355 Aug 24 19:49 dz_9_get.py
-rwxrwxrwx  1 pavlov pavlov 1014 Aug 24 19:05 dz_9_put.py
-rw-rw-r--  1 pavlov pavlov  113 Aug 24 19:57 greenplum.ini
-rwxrwxrwx. 1 pavlov pavlov  195 Aug 24 19:03 mapper.py
-rwxrwxrwx. 1 pavlov pavlov  373 Aug  1 20:06 reducer_new.py
-rwxrwxrwx. 1 pavlov pavlov  506 Aug  1 20:06 reducer.py
[pavlov@vm-cli2 python]$
```

В репозитории: https://github.com/saspav/data_engineer