به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

تحلیل داده‌های حجیم

پروژه پایانی

نام و نام خانوادگی	سبحان راستی
شماره دانشجویی	810198160
تاریخ ارسال گزارش	1400/04/29

فهرست گزارش سوالات

گام سوم: cassandra

بنده در این سوال سه جدول(البته استفاده از جدول درست نیست) ایجاد کردم یکی از آن‌ها بنابر سوالات داده شده طراحی شده است. در هر سه جدول کلید اول را کانال قرار داده‌ایم البته ما در این داده‌ها همه را از سهامیاب گرفته‌ایم پس فقط یک کانال داریم.

در جدول اولی چون میخواهیم به سوالات زمان ارسال پاسخ دهیم، پس sendTime را کلید بعدی قرار داده‌ایم:

```
session.execute("""
    CREATE TABLE IF NOT EXISTS fullData (
        canal text,
        sendTime timestamp,
        id text ,
        senderName text ,
        content text ,
        senderUsername text ,
        PRIMARY KEY (canal, sendTime, id) );
    """)
```

در جدول دوم هشتگ را کلید دوم گذاشتیم و بعد زمان ارسال را به این دلیل که بتوانیم به راحتی برای هر هشتگ در یک بازه زمانی درخواست شده داده‌های مربوطه را برگردانیم.

```
session.execute("""
    CREATE TABLE IF NOT EXISTS  hashtag (
        canal text,
        hashtag text,
        sendTime timestamp,
        id text ,
        senderName text ,
        content text ,
        senderUsername text ,
        PRIMARY KEY (canal, hashtag, sendTime, id) );
    """)
```

در جدول سوم کلمات کلیدی را کلید دوم گذاشتیم و بعد زمان ارسال را به این دلیل که بتوانیم به راحتی برای هر کلمه کلیدی در یک بازه زمانی درخواست شده داده‌های مربوطه را برگردانیم.

```
session.execute("""
    CREATE TABLE IF NOT EXISTS  keyWord (
        canal text,
        keyword text,
        sendTime timestamp,
        id text ,
        senderName text ,
        content text ,
        senderUsername text ,
        PRIMARY KEY (canal, keyword, sendTime, id) );
    """)
```

همانطور که میبینید بنده از پایتون برای اجرای کویری‌های کاساندرا استفاده کردم و این کار را با استفاده از کتابخانه cassandra انجام داده‌ام.

در داده‌ای که وارد میشود ابتدا آن را به کاساندرا اضافه میکنیم:

```
insert_fullData = session.prepare("""
        INSERT INTO fullData (
                            canal,
                            sendTime,
                            id,
                            senderName,
                            content,
                            senderUsername
                            )
        VALUES (?, ?, ?, ?, ?, ?)
        IF NOT EXISTS;
        """)
insert_hashtag = session.prepare("""
        INSERT INTO hashtag (
                            canal,
                            hashtag,
                            sendTime,
                            id,
                            senderName,
                            content,
                            senderUsername
                            )
        VALUES (?, ?, ?, ?, ?, ?, ?)
        IF NOT EXISTS;
        """)
insert_keyWord = session.prepare("""
        INSERT INTO keyWord (
                            canal,
                            keyword,
                            sendTime,
                            id,
                            senderName,
                            content,
                            senderUsername
                            )
        VALUES (?, ?, ?, ?, ?, ?, ?)
        IF NOT EXISTS;
        """)
```

برای استفاده از داده‌های وارد شده بنده شش تابع نوشته‌ام که سه عدد از آن‌ها دو بازه زمانی کار میکنند که هر کدام از این سه با یک جدول کار میکند. سه تابع بعدی سوالات خواسته شده را پشتیبانی میکند به این صورت که دو ورودی دارد که یکی تعداد (ساعت، روز، هفته، ماه، سال) و دومی اگر نیاز بود هشتگ یا کلمه کلیدی مربوطه میباشد.

ابتدا کدهایی را مشاهده میفرمایید که بازه زمانی را میگیرند:

```
def all_post(timestamp1 , timestamp2, canal = 'sahamyab' ):
    cluster = Cluster(['127.0.0.1'])
    session = cluster.connect()
    session.set_keyspace(KEYSPACE)
    code = session.prepare("""
            SELECT * FROM fullData
            WHERE canal = ? AND sendTime >= ? AND sendTime <= ?;
            """)
    result = session.execute(code, [canal, timestamp1, timestamp2])
    return result

def all_post_by_hashtag(hashtag, timestamp1 , timestamp2, canal = 'sahamyab'):
    cluster = Cluster(['127.0.0.1'])
    session = cluster.connect()
    session.set_keyspace(KEYSPACE)
    code = session.prepare("""
            SELECT * FROM hashtag
            WHERE canal = ? AND hashtag = ? AND sendTime >= ? AND sendTime <= ?;
            """)
    result = session.execute(code, [canal, hashtag, timestamp1, timestamp2])
    return result

def all_post_by_keyWord(keyword, timestamp1 , timestamp2, canal = 'sahamyab' ):
    cluster = Cluster(['127.0.0.1'])
    session = cluster.connect()
    session.set_keyspace(KEYSPACE)
    code = session.prepare("""
            SELECT * FROM keyWord
            WHERE canal = ? AND keyWord = ? AND sendTime >= ? AND sendTime <= ?;
            """)
    result = session.execute(code, [canal,keyword,timestamp1, timestamp2])
    return result
```

در ادامه کدهای داده‌هایی را شماهده میکنید که تعداد میگیرد و بنا بر تعداد برای ساعت، ماه و ..... را محاسبه میند و بر میگرداند:

```python
def get_post(n):
    timestamp2 = datetime.today()
    timestamp1 = timestamp2 - timedelta(hours=n)
    n_hour = all_post(timestamp1 , timestamp2)

    timestamp1 = timestamp2 - timedelta(days=n)
    n_day = all_post(timestamp1 , timestamp2)

    timestamp1 = timestamp2 - timedelta(weeks=n)
    n_week = all_post(timestamp1 , timestamp2)

    m = timestamp2.month - int(n % 12)
    y = timestamp2.year - int(n / 12)
    if m <= 0:
        m = 12 + m
        y -= 1
    timestamp1 = datetime.strptime(f'{y}-{m}-{timestamp2.day}', '%Y-%m-%d')
    n_month = all_post(timestamp1 , timestamp2)

    timestamp1 = datetime.strptime(f'{timestamp2.year-n}-{timestamp2.month}-{timestamp2.day}', '%Y-%m-%d')
    n_year = all_post(timestamp1 , timestamp2)

    return n_hour, n_day, n_week, n_month, n_year
```

```python
def get_post_by_hashtag(n, hashtag):
    timestamp2 = datetime.today()
    timestamp1 = timestamp2 - timedelta(hours=n)
    n_hour = all_post_by_hashtag(hashtag, timestamp1 , timestamp2)

    timestamp1 = timestamp2 - timedelta(days=n)
    n_day = all_post_by_hashtag(hashtag, timestamp1 , timestamp2)

    timestamp1 = timestamp2 - timedelta(weeks=n)
    n_week = all_post_by_hashtag(hashtag, timestamp1 , timestamp2)

    m = timestamp2.month - int(n % 12)
    y = timestamp2.year - int(n / 12)
    if m <= 0:
        m = 12 + m
        y -= 1
    timestamp1 = datetime.strptime(f'{y}-{m}-{timestamp2.day}', '%Y-%m-%d')
    n_month = all_post_by_hashtag(hashtag, timestamp1 , timestamp2)

    timestamp1 = datetime.strptime(f'{timestamp2.year-n}-{timestamp2.month}-{timestamp2.day}', '%Y-%m-%d')
    n_year = all_post_by_hashtag(hashtag, timestamp1 , timestamp2)

    return n_hour, n_day, n_week, n_month, n_year
```

```python
def get_post_by_keyWord(n, keyword):
    timestamp2 = datetime.today()
    timestamp1 = timestamp2 - timedelta(hours=n)
    n_hour = all_post_by_keyWord(keyword, timestamp1 , timestamp2)

    timestamp1 = timestamp2 - timedelta(days=n)
    n_day = all_post_by_keyWord(keyword, timestamp1 , timestamp2)

    timestamp1 = timestamp2 - timedelta(weeks=n)
    n_week = all_post_by_keyWord(keyword, timestamp1 , timestamp2)

    m = timestamp2.month - int(n % 12)
    y = timestamp2.year - int(n / 12)
    if m <= 0:
        m = 12 + m
        y -= 1
    timestamp1 = datetime.strptime(f'{y}-{m}-{timestamp2.day}', '%Y-%m-%d')
    n_month = all_post_by_keyWord(keyword, timestamp1 , timestamp2)

    timestamp1 = datetime.strptime(f'{timestamp2.year-n}-{timestamp2.month}-{timestamp2.day}', '%Y-%m-%d')
    n_year = all_post_by_keyWord(keyword, timestamp1 , timestamp2)

    return n_hour, n_day, n_week, n_month, n_year
```

در کدها پیشبینی بزرگ شدن عدد ماه یعنی بزرگتر از 12 یا کوچکتر از 1 شده است.

گام پنجم:clickhouse

مقدمه

در این بخش چالش زیادی برای راه اندازی Clickhouse داشتیم، چرا که برای کار با این دیتاست نیاز به سیستم عامل لینوکس بود. البته ابزارهای دیگری در ویندوز این امکان را فراهم میکرد که خود آن ها نیز چالش های خاص خود را داشت. به هر صورت پس از نصب و راه اندازی Clickhouse در حالیکه سرور آن در حال اجراست به وسیله ی کد پایتون داده ها را در آن وارد کرده و جداول را ساختیم.

توضیحات کد پایتون و ارتباط با Clickhouse

در این بخش همانطور که در تصاویر زیر مشخص است پس از اتصال به سرور Clickhouse داده های پردازش شده و ریخته شده در فایل توسط کافکا به صورت آنلاین و مداوم را خوانده و پس از انتخاب بخش های مورد نیاز آن، حاصل را به Clickhouse اضافه مینماییم.

```python
import json
from clickhouse_driver import Client
from datetime import datetime

# client = Client('localhost')
client = Client('localhost',
                password='*****')

client.execute('DROP TABLE IF EXISTS default.sahamyab')
client.execute("""CREATE TABLE IF NOT EXISTS default.sahamyab
                (
                id String,
                sendTime DateTime,
                sendTimePersian String,
                hashtags Array(Nullable(String)),
                key_words Array(Nullable(String)),
                senderUsername String,
                senderName String,
                content String
                )
                 ENGINE = MergeTree PARTITION BY toYYYYMMDD(sendTime) ORDER BY toYYYYMMDD(sendTime)
                """)


def insert(jData):
    data = [[
        jData['id'],
        datetime.strptime(jData['sendTime'], '%Y-%m-%dT%H:%M:%SZ'),
        jData['sendTimePersian'],
        jData['_meta']['hashtags'],
        jData['_meta']['key_words'],
        jData['senderUsername'],
        jData['senderName'],
        jData['content']
    ]]
```

بخش اول کد اضافه کردن داده به **Clickhouse**

```
                                '('
                                'id,'
                                'sendTime,'
                                'sendTimePersian,'
                                'hashtags,'
                                'key_words,'
                                'senderUsername,'
                                'senderName,'
                                'content'
                                ')'
                                ' VALUES ', data)
    print(jData['id'])
    return True

def insert_jsons(data):
    for i in data:
        insert(i)

import json
# use the below code to read the data
f2 = open('preprocessed_data.json', encoding="utf8")
jdata = json.load(f2)
f2.close()
insert_jsons(jdata)
```

بخش دوم کد اضافه کردن داده به **Clickhouse**

داده ها در محیط **Clickhouse**

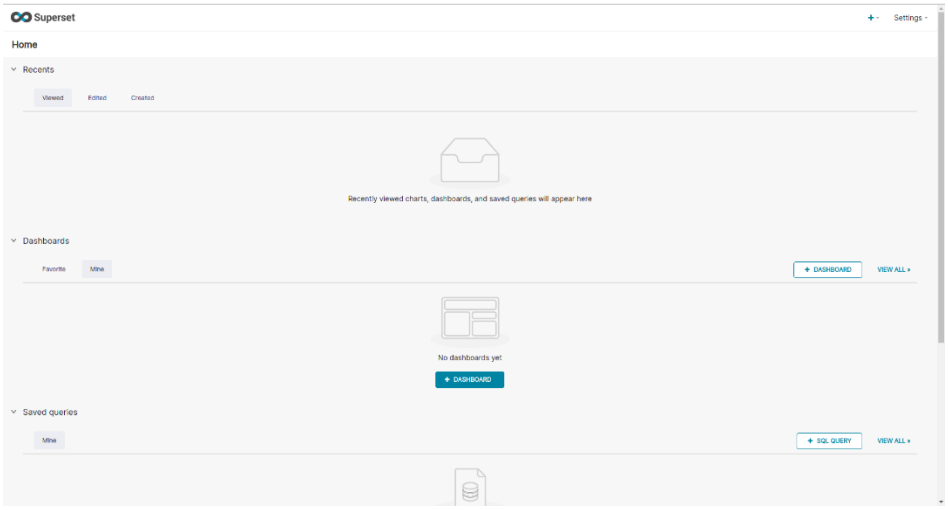
گام ششم:apache superset

مقدمه

در این بخش پس از اضافه شدن داده ها به Clickhouse و نصب و راه اندازی Apache Superset، Superset را به Clickhouse متصل کردیم. این اتصال در صورت داشتن رمز برای Clickhouse در ویندوز برای ما چالش برانگیز بود، چرا که نمیتوانستیم اتصال را برقرار کرده و از طرف دیگر نتوانستیم رمز را حذف نماییم. در نهایت در سیستم عامل لینوکس این مهم میسر شد.

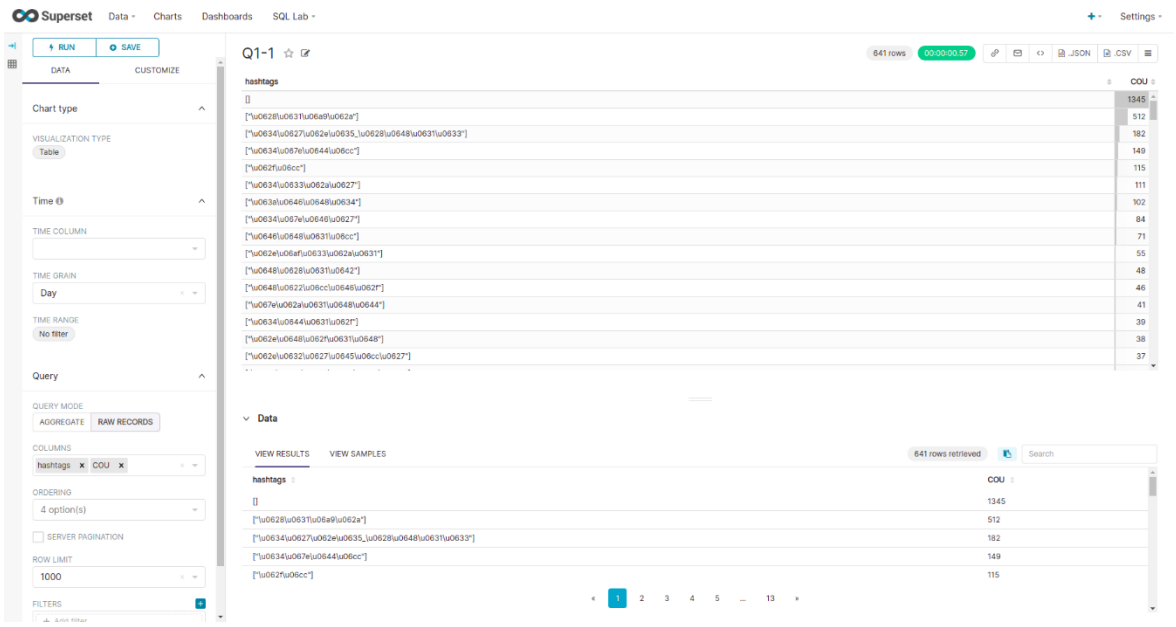
نمودارها و موارد خواسته شده

با توجه به موارد خواسته شده در هر بخش کوئری مربوطه را داده و داده هارا از Clickhouse دریافت کرده سپس بعد از explore کردن داده ها و ذخیره ی آنها با انتخاب پارامتر ها و نمودارهای مناسب نتیجه را بدست آوردیم برای مواردی نیز به وسیله ی رابط کاربری superset و محیط گرافیکی آن و بدون نیاز به کوئری نمودار های خواسته شده را بدست آوردیم. همانطور که در دستور کار خواسته شده است **سه داشبورد** از موارد خواسته شده تهیه کردیم که آنها را در ادامه می بینیم.

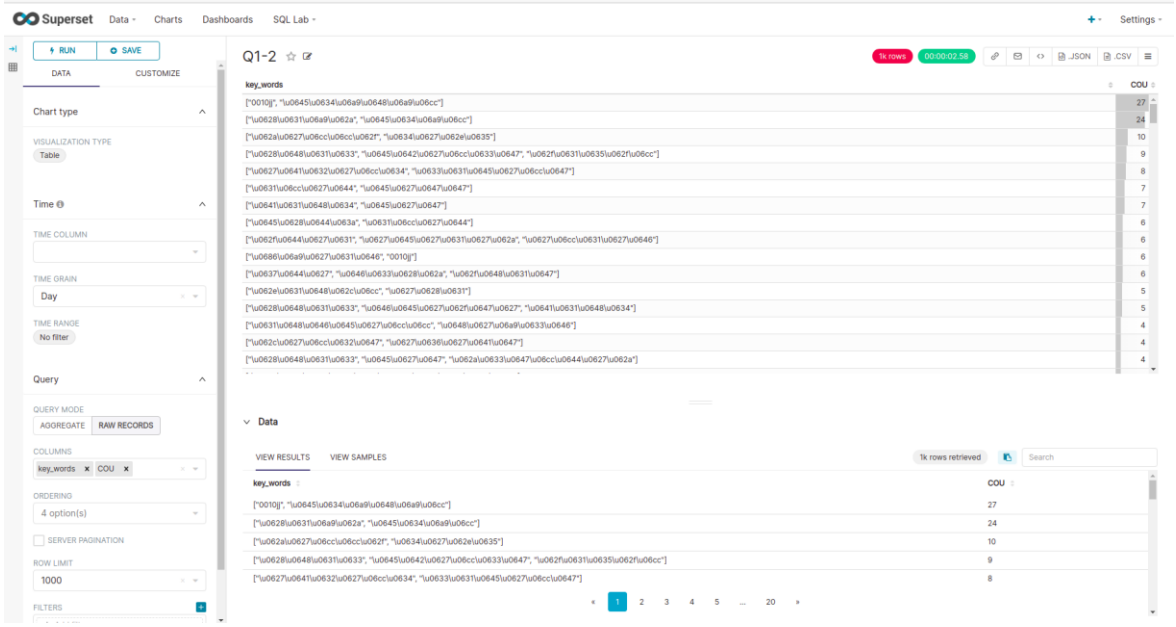


محیط **superset**

در تصاویر 31 و 32 جداول بدست آمده پس از در خواست های انجام شده را می بینید. در تصویر 31 جدول و نمودار نزولی برای بیشترین تکرار در بین هشتگ ها و همین مورد را در تصویر 32 برای کلمات کلیدی می بینید.

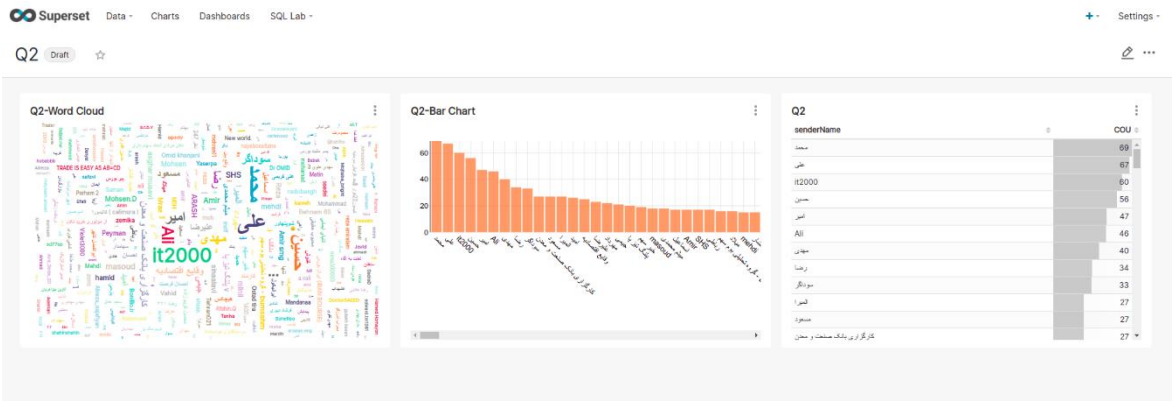


هشتگ های دریافتی به همراه تعداد و نمودار آنها به صورت اسکی در محیط **superset**



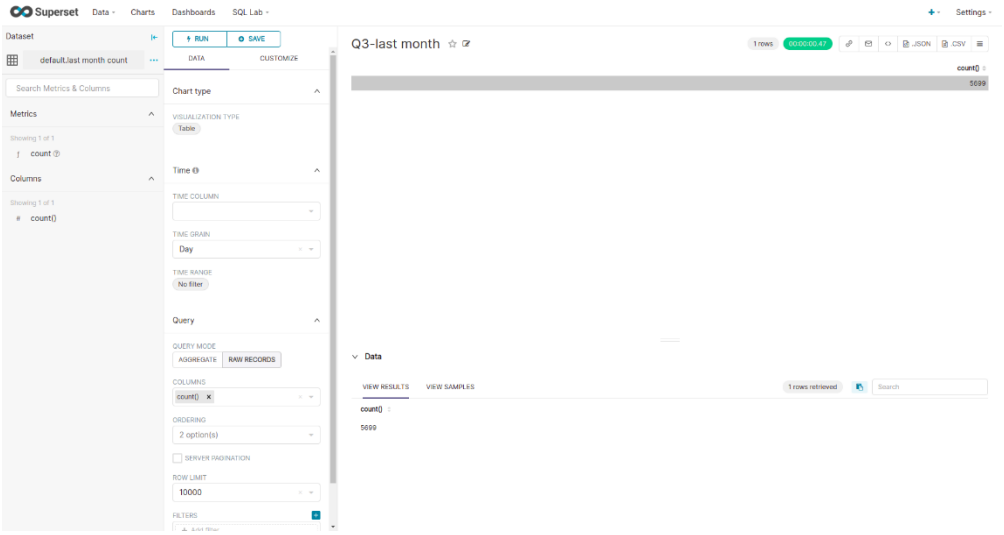
کلمات کلیدی به همراه تعداد و نمودار آنها به صورت اسکی در محیط **superset**

در این بخش تعداد توییت های ارسال شده توسط هر کاربر را به دست آورده و به صورت نزولی مرتب کردیم و نمودار های مربوطه را کشیدیم. در ابر کلمات که یک ابزار کارآمد برای تحلیل داده های متنی است مشاهده میکنیم که کاربرانی با نام علی، محمد، سوداگر،it2000 از فعال ترین کاربرها در این حوزه بوده اند.

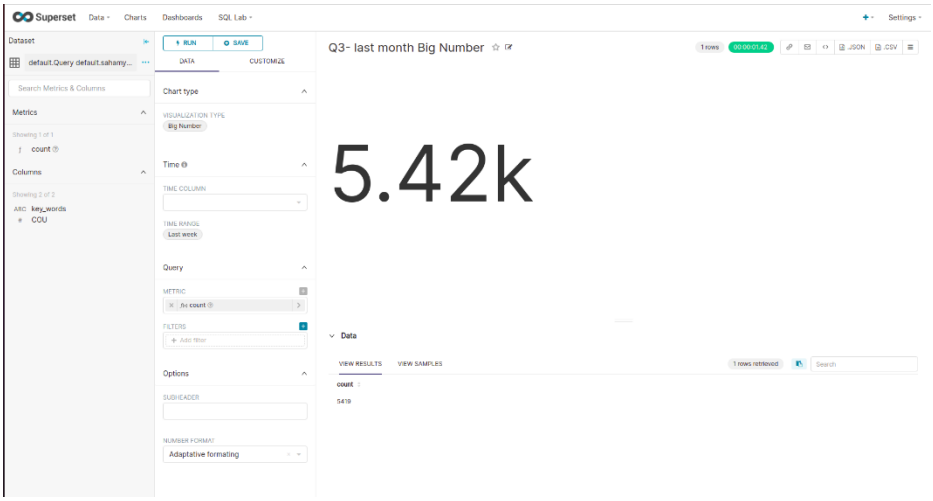


تعداد توییت های کاربران به صورت نزولی در قالب جدول نمودار و ابرکلمات در محیط **superset**

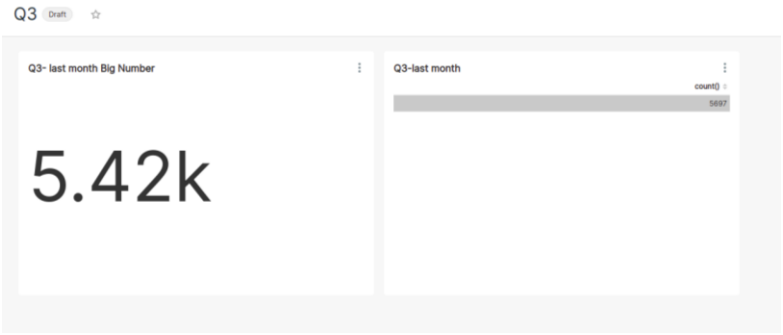
در ادامه تعداد توییت ها را در زمان ها مختلف بررسی کردیم که در تصویر 33 تعداد به دست آمده برای ماه اخیر را میبینیم و در تصاویر بعدی فرم این داده را به صورت Big Number کشیدیم.



تعداد توییت ها در ماه اخیر در محیط **superset**



Big Number تعداد توییت ها در ماه اخیر در محیط **superset**



Big Number و تعداد توییت ها در ماه اخیر در محیط **superset**