



به نام خدا



دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

## کلان داده و تحلیل داده‌های حجمی

پروژه پایانی

فاطمه ایمانی پور

رومینا اوجی

مهندی کیخا

سروش زاده‌خواست

## فهرست گزارش سوالات

4.....	مقدمه
5.....	گام اول – دریافت اطلاعات و preprocess
5.....	راه اندازی کافکا
5.....	جمع آوری داده
9.....	پیش پردازش داده ها
12.....	گام دوم-persistence
12.....	نصب و راه اندازی:
14.....	استفاده از persian analyzer
18.....	داسبورد طراحی شده:
19.....	ابر کلمات یک کانال یا خبرگزاری خاص در یک بازه زمانی
22.....	متن ده پست اخیری که دریافت شده است
24.....	تعداد پستهای ارسال شده به ازای چند تا از کلمات کلیدی خاص در یک بازه زمانی
28.....	ده هشتگ بیشتر استفاده شده در یک کانال با تعداد تکرار هر هشتگ و در یک بازه زمانی
31.....	یک نمودار به انتخاب خودتان (نمودار دایره‌ای تعداد پستهای هر کانال در 24 ساعت اخیر).
33.....	کوئری تمام پستهای حاوی یک کلمه خاص از یک کانال خاص در یک بازه زمانی:
38.....	گام سوم - Channel/Hashtag History (Cassandra)
39.....	کاساندرا
39.....	نصب و راه اندازی
42.....	Kafka
43.....	ایجاد KEYSPACE
44.....	طراحی جداول
44.....	جدول posts
46.....	جدول channel

47	جدول keyword
48	ذخیره اطلاعات در جداول
49	سوالات
54	گام چهارم - Statistics
71	گام پنجم- clickhouse
71	ClickHouse
74	Superset
75	گزارشها مرتبط با هشتگها/کلمات کلیدی
81	گزارشها مربوط به کانالها/کاربران (در صورت استفاده از توئیتها)
84	گزارش‌های عمومی سامانه مانند آمارکلی دریافت اطلاعات در یک روز و یک ساعت گذشته
89	گزارش‌های مرتبط با یک کanal خاص / یک هشتگ خاص
96	ساخت یک مدل پیش‌بینی کننده با اسپارک- بخش امتیازی
96	پیش‌پردازش و دریافت داده‌ها
99	پیش‌بینی زمان ارسال پست بعدی:
101	پیش‌بینی هشتگ‌های یک پست:

## مقدمه

در این پروژه پیام رسان تلگرام به عنوان منبع داده انتخاب شده است. 500 کanal خبر رسانی فارسی تلگرام انتخاب شده تا با کمک کتابخانه<sup>1</sup> telethon این کanal ها خوش<sup>2</sup> شوند.

ساختار ارتباطی بخش های مختلف بر مبنای apache kafka تعبیه شده که در ادامه جزئیات آن نیز ذکر میشود.

کد مربوط به جمع آوری اطلاعات و نیز پیش پردازش داده ها روی یک سرور به شکل مداوم در حال اجراست و بعد از انجام مرحله پیش پردازش داده ها وارد تاپیک مشخصی از کافکا میشوند تا در ادامه آنالیز های مختلف روی آنها انجام شود .

اطلاعات سرور فوق در اختیار اعضای گروه قرار گرفته تا هر یک از آنان بتوانند با دریافت اطلاعات از کافکا برروی تاپیک تعریف شده در گام اول(مرحله پیش پردازش) ؛ مراحل مختلف آتی را روی سیستم های خود انجام دهند. در زیر گزارش چگونگی انجام و خروجی های هر گام آورده شده است.

آدرس github پروژه : <https://github.com/rominaoji/big-data-final-project> است. کد های هر بخش به تفکیک در پوشه ای هم نام با آن بخش در این آدرس قرار داده شده است

نویسندهای گزارش:

گام اول : فاطمه ایمانی پور

گام دوم : رومینا اوجی

گام سوم : سروش زاده خواست

گام چهارم: مهدی کیخا

گام پنجم : فاطمه ایمانی پور

بخش امتیازی : رومینا اوجی

<sup>1</sup> <https://tl.telethon.dev/>  
<sup>2</sup> crawl

## گام اول - دریافت اطلاعات و preprocess

### راه اندازی کافکا

پس از نصب کافکا و انجام تنظیمات اولیه لازم است تا تاپیک های لازم را برای انتقال پیام بین ماژول های مختلف در کانال های مناسب ایجاد کنیم.

به دو تاپیک rawData جهت انتقال داده خام از ماژول خرش به ماژول پیش پردازش و نیز تاپیک preprocessedData جهت انتقال پیام های پیش شده از ماژول پیش پردازش به سایر ماژول ها نیاز داریم.

با استفاده از دستور زیر میتوانیم یک topic جدید تولید کنیم :

```
[root@main-worker-kuber kafka_2.13-2.7.0]# ./bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic newTopic
Created topic newTopic.
```

و میتوانیم لیست تمام تاپیک های تعریف شده روی kafka را به کمک دیتور زیر بگیریم :

```
[root@main-worker-kuber kafka_2.13-2.7.0]# ./bin/kafka-topics.sh --zookeeper localhost:2181 --list
TutorialTopic
__consumer_offsets
newTopic
preProcessedData
rawData
test
```

### جمع آوری داده

همانظور که در بخش مقدمه گفته شد در این پروژه برای جمع آوری داده ها از کتابخانه telethon استفاده شده است این کتابخانه پایتونی به جهت تعامل با API های تلگرام تهییه شده و استفاده از امکاناتی نظیر دریافت اطلاعات حساب های کاربری، اطلاعات کانال ها و گروه ها و .... (در صورت وجود مجوز های لازم) به عنوان یک حساب کاربری یا یک بات فراهم میسازد.

برای این کار ابتدا لازم است تا اطلاعات لازم برای استفاده از API های تلگرام مانند app\_key را دریافت کنیم و با این اطلاعات و البته اطلاعات مربوط به حساب کاربری مورد نظرمان بتوانیم به اصطلاح یک client تلگرامی ایجاد کنیم.

قطعه کد زیر مربوط به این مرحله است.

```

-- 
17 from telethon import TelegramClient
18 from telethon.errors import SessionPasswordNeededError
19 import os
20 import asyncio
21
22 config = configparser.ConfigParser()
23 config.read("config.ini")
24
25 # Setting configuration values
26 api_id = config['Telegram']['api_id']
27 api_hash = config['Telegram']['api_hash']
28 api_hash = str(api_hash)
29 phone = config['Telegram']['phone']
30 username = config['Telegram']['username']
31 print(username)
32
33
34
35 #Create the client and connect
36 async def create_client():
37     client = TelegramClient(username, api_id, api_hash)
38     await client.start()
39     await client.connect()
40     print("Client Created")
41     # Ensure you're authorized
42     if not await client.is_user_authorized():
43         client.send_code_request(phone)
44         try:
45             await client.sign_in(phone, input('Enter the code: '))
46         except SessionPasswordNeededError:
47             await client.sign_in(password=input('Password: '))
48     return client
49

```

نکته : فایل config.ini حاوی اطلاعات API و نیز حساب کاربری است که قصد داریم تا آن را رصد کنیم.

بعد از تعریف و ایجاد یک client آنگاه با تعریف یک listener پیام های ارسال شده کانال های حساب کاربری مورد نظر را در هر لحظه دریافت میکنیم :

```

116 from telethon import events
117 @events.register( events.NewMessage(func=lambda e: e.is_channel and not e.is_group))
118 async def my_event_handler(event):
119
120     print("=====new message=====")
121     data = json.loads(json.dumps(event.message.to_dict(),default=date_format))
122

```

نکته حائز اهمیت آن است که در این بخش API مذکور تنها ID کانالی که پیام را ارسال کرده در اختیار ما قرار میدهد . حال برای آنکه بدانیم این ID مربوط به کدام کانال است باید با استفاده از API دیگری با ارسال ID کانال اطلاعاتی نظیر , title , username ... کانال را دریافت کنیم . در این بخش یک دیتابیس Mongo در نظر گرفته شده که بعد از دریافت اطلاعات مربوط به هر کانال این اطلاعات را در دیتابیس ذخیره میکنیم تا در دفعات بعدی بتوانیم اطلاعات کانال مورد نظر را به شکل محلی استخراج کنیم و نیازی به فراخوانی مجدد API جهت دریافت اطلاعات کانال نباشد . به این شکل با توجه به اینکه 500 کانال داریم نهایتا سرویس مورد نظر تنها 500 بار فراخوانی خواهد شد.

این مرحله به شکل کلی از 4 زیر بخش زیر تشکیل شده است:

(1) فراخوانی متدها get\_channel\_name و name با استفاده از ID کانال

:

```
118 from telethon import events  
119 @events.register(events.NewMessage(func=lambda e: e.is_channel and not e.is_group))  
120 async def my_event_handler(event):  
121  
122     print("=====new message=====")  
123     data = json.loads(json.dumps(event.message.to_dict(), default=date_format))  
124     info=await get_channel_name(data['peer_id']['channel_id'])  
125     data['chennel_name']=info[0]  
126     data['chennel_username']=info[1]  
127
```

: get\_channel\_name (2) تعریف متدها

در این متدها ابتدا چک میشود که اطلاعاتی از کانال مورد نظر به شکل محلی ذخیره شده یا خیر . اگر پاسخ به این سوال آری باشد آنگاه اطلاعات موجود در دیتابیس در مورد این کانال برگردانده میشود در غیر این صورت اطلاعات کانال از طریق API تلگرام دریافت میشود.

```
91 | async def get_channel_name(channel_id):  
92 |     res=find_channel_info(channel_id)  
93 |     print(res)  
94 |     if res is None:  
95 |  
96 |         print('going to get channel {0} info'.format(channel_id))  
97 |         ch_info=await fetch_channel_info(channel_id)  
98 |  
99 |         channel_name=ch_info.to_dict()['chats'][0]['title']  
100 |         channel_username=ch_info.to_dict()['chats'][0]['username']  
101 |  
102 |         print('going to save channel {0} info'.format(channel_id))  
103 |         save_channel_info(channel_id,channel_name,channel_username)  
104 |  
105 |     else:  
106 |         channel_name=res['name']  
107 |         channel_username=res['username']  
108 |  
109 |     return [channel_name,channel_username]  
110 |  
111 |
```

(3) تعریف متدها fetch\_channel\_info: جهت دریافت اطلاعات کانال از API تلگرام:

```
84     async def fetch_channel_info(channel_id):
85         result = await client(functions.channels.GetFullChannelRequest(channel=channel_id))
86         return(result)
87 
```

4) تعریف روش های مناسب برای ذخیره و بازیابی اطلاعات هر کanal در دیتابیس Mongo

```
66     from pymongo import MongoClient
67     mongoClient = MongoClient()
68     db=mongoClient.channelInfo
69
70
71     def save_channel_info(channelId,channelName,channelUsername):
72         channelInfo = {
73             '_id' : channelId,
74             'name' : channelName,
75             'username': channelUsername
76         }
77         db.channelInfo.insert_one(channelInfo)
78
79
80     def find_channel_info(channelId):
81         return db.channelInfo.find_one({"_id":channelId})
82 
```

نمونه ای از خروجی این مرحله :

```
=====new message=====
{'_id': 1000180159, 'name': '24پرس | Bourse24', 'username': 'bourse24ir'}
('': 'Message', 'id': 199983, 'peer_id': ('': 'PeerChannel', 'channel_id': 1000180159), 'date': None, 'message': '#تغییر #افزایش معرف کشورمای دنبی تا سال 2025 به بین از 13 میلیون تن می رسد\nبن در سال 2025 به بین از 13 میلیون تن می رسد', 'out': False, 'mentioned': False, 'media_unread': False, 'silent': False, 'post': True, 'from_scheduled': False, 'legacy': False, 'edit_hide': False, 'pinned': False, 'from_id': None, 'fwd_from': None, 'via_bot_id': None, 'reply_to': ('': 'MessageReplyHeader', 'reply_to_msg_id': 199982, 'reply_to_peer_id': None, 'reply_to_top_id': 199971), 'media': ('': 'MessageMediaPhoto', 'photo': ('': 'Photo', 'id': 6033032534130013606, 'access_hash': -2031337325634492198, 'file_reference': None, 'date': None, 'sizes': [{}: 'PhotoStrippedSize', 'type': 'i', 'bytes': None], {}: 'PhotoSize', 'type': 'm', 'w': 320, 'h': 180, 'size': 19512}, {}: 'PhotoSize', 'type': 'x', 'w': 800, 'h': 450, 'size': 66866}, {}: 'PhotoSizeProgressive', 'type': 'y', 'w': 1280, 'h': 720, 'size': [7481, 10069, 70534, 75451, 80393]]}, 'dc_id': 4, 'has_stickers': False, 'video_sizes': [], 'ttl_seconds': None}, 'reply_markup': None, 'entities': [{}: 'MessageEntityHashtag', 'offset': 5, 'length': 5}, {}: 'MessageEntityMention', 'offset': 95, 'length': 11}], 'views': 1, 'forwards': 0, 'replies': None, 'edit_date': None, 'post_author': None, 'grouped_id': None, 'restriction_reason': [], 'ttl_period': None, 'chennel_name': '24پرس | Bourse24', 'chennel_username': 'bourse24ir'}
=====new message=====
{'_id': 1000180159, 'name': '24پرس | Bourse24', 'username': 'bourse24ir'}
('': 'Message', 'id': 199984, 'peer_id': ('': 'PeerChannel', 'channel_id': 1000180159), 'date': None, 'message': '#تغییر #افزایش معرف کشورمای دنبی تا سال 2025 به بین از 13 میلیون تن می رسد', 'out': False, 'mentioned': False, 'media_unread': False, 'silent': False, 'post': True, 'from_scheduled': False, 'legacy': False, 'edit_hide': False, 'pinned': False, 'from_id': None, 'fwd_from': None, 'via_bot_id': None, 'reply_to': ('': 'MessageReplyHeader', 'reply_to_msg_id': 199983, 'reply_to_peer_id': None, 'reply_to_top_id': 199971), 'media': ('': 'MessageMediaPhoto', 'photo': ('': 'Photo', 'id': 6033032534130013607, 'access_hash': -9198270833847020472, 'file_reference': None, 'date': None, 'sizes': [{}: 'PhotoStrippedSize', 'type': 'i', 'bytes': None], {}: 'PhotoSize', 'type': 'm', 'w': 320, 'h': 180, 'size': 15607}, {}: 'PhotoSize', 'type': 'x', 'w': 800, 'h': 450, 'size': 59633}, {}: 'PhotoSizeProgressive', 'type': 'y', 'w': 1280, 'h': 720, 'size': [8129, 10636, 67856, 72276, 76071]}], 'dc_id': 4, 'has_stickers': False, 'video_sizes': [], 'ttl_seconds': None}, 'reply_markup': None, 'entities': [{}: 'MessageEntityHashtag', 'offset': 5, 'length': 5}, {}: 'MessageEntityMention', 'offset': 43, 'length': 111], 'views': 1, 'forwards': 0, 'replies': None, 'edit_date': None, 'post_author': None, 'grouped_id': None, 'restriction_reason': [], 'ttl_period': None, 'chennel_name': '24پرس | Bourse24', 'chennel_username': 'bourse24ir'} 
```

سپس اطلاعات جمع آوری شده در این مرحله را در کanal rawData کافکا منتشر میکنیم .

```

105 from telethon import events
106 @events.register( events.NewMessage(func=lambda e: e.is_channel and not e.is_group))
107 async def my_event_handler(event):
108
109     print("=====new message=====")
110     data = json.loads(json.dumps(event.message.to_dict(),default=date_format))
111     info=await get_channel_name(data['peer_id'][ 'channel_id'])
112     data[ 'channel_name']=info[0]
113     data[ 'chennel_username']=info[1]
114     json_data = json.dumps(data)
115     print(data)
116     producer = KafkaProducer(bootstrap_servers='localhost:9092')
117     future= producer.send('test5', json_data.encode('utf-8'))
118     result = future.get(timeout=60)
119
120
121

```

کد مربوط به این بخش در فایل crawlTelegram.py در پوشه step1 آورده شده است.

## پیش پردازش داده ها

در این مرحله داده ها را از تاپیک rawData کافکا دریافت میکنیم و پیش پردازش های لازم را روی آنها انجام میدهیم.

```

154 from kafka import KafkaConsumer
155 consumer=KafkaConsumer(bootstrap_servers='localhost:9092')
156 consumer.subscribe('rawData')
157 for msg in consumer:
158     message=asyncio.run(preprocess(msg.value.decode('utf-8')))

```

پیش پردازش های انجام شده :

افزودن یک ID یکتا به هر پیام با استفاده از متده UUID و نیز اضافه کردن زمان ارسال پیام در فرمت timestamp . اطلاعاتی همچون متن پیام ، ID و name و username کانال ارسال کننده پیام به شکل مستقیم از داده خام اولیه استخراج شده و در یک چارچوب جدید کنار هم قرار میگیرند .

```

125 async def preprocess(message):
126     print("=====new message=====")
127     d = json.loads(message)
128     if(len(d[ 'message'])>2):
129
130         ts = time.time()
131         message={}
132         message[ 'id']=str(uuid.uuid4())
133         message[ 'send_time']=ts
134         message[ 'context']=d[ 'message']
135         message[ 'sender_id']=d[ 'peer_id'][ 'channel_id']
136         message[ 'sender_name']=d[ 'chennel_name']
137         message[ 'sender_username']=d[ 'chennel_username']
138

```

برای استخراج hashtags و email ها از متن پیام به شکل زیر از دو regex استفاده میکنیم تا تمام های احتمالی در متن را استخراج کنند.

```
--  
88 def extract_hashtags(message):  
89     return re.findall(r"#(\w+)", message)  
90  
91  
92 def extract_emails(message):  
93     return re.findall(r'[\w\.-]+@[ \w\.-]+', message)  
94  
95
```

برای استخراج keyword ها از متن از BERT استفاده میکنیم.

یک مدل bi-directional transformer که این امکان رو به ما میدهد تا کلمات رو به بردار های معنی دار تبدیل کنیم.

کلمات کلیدی کلماتی خواهند بود که بردار آنها بیشترین شباهت cos را به بردار متن مورد نظر داشته باشد.

چرا که به این شکل آنها میتوانند نماینده خوبی از متن باشند.

برای انجام این کار ابتدا لازم است تا لیستی از کلمات کاندید ایجاد کنیم

برای این کار ابتدا لازم است متن را به شکل زیر پیش ویرایش کنیم

```
52 def pre_process_message(message):  
53     normalizer = Normalizer()  
54     norm_doc=normalizer.normalize(message)  
55     doc_tokens=word_tokenize(norm_doc)  
56     tagger = POSTagger(model='resource/postagger.model')  
57     pos=tagger.tag(word_tokenize(message))  
58     verbs=[]  
59     for (key, val) in pos:  
60         if(val=='V'):  
61             verbs.append(key)  
62     stemmer = Stemmer()  
63     pre_doc= [word for word in doc_tokens if ( stemmer.stem(word) not in stop_words and word not in stop_words) and (stemmer.stem(word) not in verbs and word not in verbs) and not is_number(word)]  
64     pre_doc=' '.join(pre_doc)  
65  
66     return pre_doc
```

1. Normalize متن کردن

2. Tokenize متن کردن

3. اعمال POS tagger روی متن و یافتن فعل های متن

حال در هر متن tokenize شده تنها token هایی را نگه میداریم که :

1. خود و آنها stemm stop word نباشند (لیست stop word ها را به شکل ثابت از

چند مرجع پیش از این استخراج و در فایل stop\_words.txt ذخیره کرده ایم)

در اینجا stemm کلمات را هم در نظر میگیریم چرا که ممکن است شکل ظاهری کلمه stop word نباشد اما ریشه آن stop word باشد.

2. فعل نباشد.

3. عدد نباشد.

سپس به شکل زیر کلمات کلیدی را استخراج میکنیم.

```

69 def extract_keyword(message, keyword_count):
70     processed_message=pre_process_message(message)
71     keywords=[]
72     if(len(processed_message)<1):
73         return keywords
74     n_gram_range = (1, 1)
75     # Extract candidate words/phrases
76     count = CountVectorizer(ngram_range=n_gram_range).fit([processed_message])
77     candidates = count.get_feature_names()
78     model = SentenceTransformer('distilbert-base-nli-mean-tokens')
79     doc_embedding = model.encode([processed_message])
80     candidate_embeddings = model.encode(candidates)
81     top_n = keyword_count
82     distances = cosine_similarity(doc_embedding, candidate_embeddings)
83     keywords = [candidates[index] for index in distances.argsort()[0][-top_n:]]
84     keywords= check_static_keyword(processed_message,keywords)
85     return keywords
86

```

در اینجا بعد از بدست آوردن شباهت کسینوسی هر یک از کلمات کاندید با متن میتوان انتخاب نهایی keyword ها را به چند شکل انجام داد :

1. کلماتی که شباهت آن ها با متن از یک حد مشخصی بیشتر باشد.

2. N کلمه ای که بیشترین شباهت را با متن داشته باشند .

که در اینجا روش دوم را انتخاب کردیم و n را برابر با 4 در نظر گرفتیم .

همچنین لازم است تا در صورت وجود کلمات کلیدی که در صورت سوال آورده شده در متن آن ها را نیز حتما تحت هر شرایطی به عنوان کلمه کلیدی متن بیاوریم که برای این کار از روش زیر استفاده میکنیم :

```

43
44 static_keywords=[['انفایل', 'دلان', 'اطلاع', 'کورس', 'دانشگاه', 'کرونا', 'بیرونی', 'اتصال', 'تحريم', 'دولت', 'کوید', 'کوید19', 'کوید19', 'روحانی', 'حسن روحانی']]
45
46
47 def check_static_keyword(processed_message,keywords):
48     for word in processed_message.split():
49         if word in static_keywords and word not in keywords:
50             keywords.append(word)
51     return keywords

```

نمونه ای از خروجی این مرحله :

```
new message
('id': 'b5e9b360-301d-4f7d-8491-e6004c180d24', 'context': 'نیز میتوانید تبریت فیادین شهرداری تهران\n' +
    'کلی امروز و فرد سایخ برداشتی داری داشتند. این روز در ۱۵ جایگاه عرضه بهداشتی دام در تهران \u25aa' +
    'تفصیلی از آنها نماید. این روز در ۱۶ میکر تاسیت و این روز در ۲۰۰۰ نفر از این ایستگاهها بازدید کردند.\n' +
    'بجز اینها، ۱۹ نفر از ایستگاههای تهران (جهانشنبه ۳۰ نیز) از ساعت ۸ تا ۱۰ مراقبه کردند.\n' +
    'send_time': 1626783151.6113122}, 'sender_id': 1105855986, 'sender_name': '@AkhbarFourzi', 'keywords': []}
new message
('id': 'dd78d080-5ceb-4257-8663-c562c9a2eb45', 'context': 'رسیم پیش راه مازندران! خوش بود و خود را به مازندران راه گیرد \u25aa' +
    'مساکن ۲۲۹ دستگاه خودروی غیربروگی که قصد رزرو به مازندران داشتند، اما شمار اعمال قانونی به میان ایستگاه خودرو فکر نداشتند. این دستگاه خودرو نیز اعمال قانون خوردند. نیزم \u25aa' +
    'دانشمندانه شدند. این دستگاه خودرو نیز اعمال قانون خوردند. نیزم \u25aa' +
    'send_time': 1626783156.2924619, 'sender_id': 1102274913, 'sender_name': 'Akhbar_Dagh', 'keywords': []}
[], 'emails': [], 'hashtags': []}
[{"text": "رسیم پیش راه مازندران! خوش بود و خود را به مازندران راه گیرد \u25aa"}, {"text": "مساکن ۲۲۹ دستگاه خودروی غیربروگی که قصد رزرو به مازندران داشتند، اما شمار اعمال قانونی به میان ایستگاه خودرو فکر نداشتند. این دستگاه خودرو نیز اعمال قانون خوردند. نیزم \u25aa"}, {"text": "دانشمندانه شدند. این دستگاه خودرو نیز اعمال قانون خوردند. نیزم \u25aa"}, {"text": "send_time": 1626783156.2924619, "sender_id": 1102274913, "sender_name": "Akhbar_Dagh", "keywords": []}]}]
```

در اخر پیام پیش پردازش شده بر روی تاپیک preprocessedData جهت آنالیز های بعدی منتشر میشود.

```
157 if not message is None:
158     producer = KafkaProducer(bootstrap_servers='localhost:9092')
159     future= producer.send('preProcessedData', message.encode('utf-8'))
160     result = future.get(timeout=60)
161
```

کد مربوط به این بخش در آدرس `github` پروژه در فolder `step1-crawl-preprocess` در فایل `crawlTelegram.py` در موجود است.

## گام دوم-persistence

### نصب و راه اندازی:

برای نصب و راه اندازی elasticSearch از داکر (bitnami) استفاده شده است.

آن به شرح زیر می باشد:

```

version: "2"
services:
  elasticsearch:
    image: docker.io/bitnami/elasticsearch:7
    volumes:
      - ./data/elastic:/bitnami/elasticsearch/data
    ports:
      - "9200:9200"
  kibana:
    image: docker.io/bitnami/kibana:7
    ports:
      - "5601:5601"
    volumes:
      - "kibana_data:/bitnami/kibana"
    depends_on:
      - elasticsearch
volumes:
  elasticsearch_data:
    driver: local
  kibana_data:
    driver: local

```

برای استفاده از آن چون روی WSL کار می‌کنیم و بر روی آن ubunto 20.04 را نصب کرده‌ایم به و باید هر بار دسترسی‌ها ایجاد شود bash فایل زیر نوشته شده است که آن را با دستور sh init.sh اجرا می‌کنیم

```

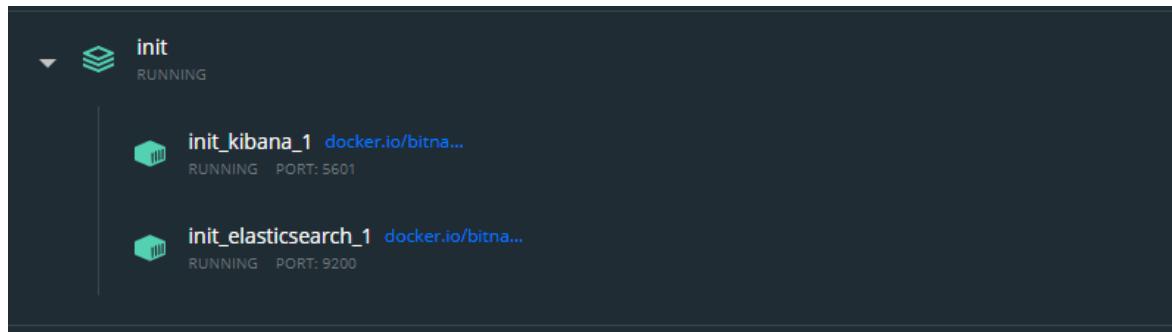
mkdir -p data/elastic
mkdir -p data/cassandra

sudo chmod -R 777 data

sudo sysctl -w vm.max_map_count=262144

```

سپس با -d docker-compose up این دو سرویس را فعال می‌کنیم.



## استفاده از persian analyzer

آنالایزر برای داده‌های متنی استفاده می‌شود که به منظور تحلیل متن در زمان indexing و search مورد استفاده قرار می‌گیرد.

برای ایجاد آن از دستور زیر استفاده شده است و آن را بر روی محتوای پست‌ها یعنی context اعمال می‌کیم که بخش اول برای تعریف این آنالایزر بر روی index ای به نام telegram-data می‌باشد و بخش دوم که mapping می‌باشد آنالایزرهای مختلف که برای آن طراحی شده است (rebuilt\_persian) بر روی فیلد context اعمال می‌شود.

```

PUT /telegram-data
{
  "settings": {
    "analysis": {
      "char_filter": {
        "zero_width_spaces": {
          "type": "mapping",
          "mappings": [ "\u200C=>\u0020"]
        }
      },
      "filter": {
        "persian_stop": {
          "type": "stop",
          "stopwords": "_persian_"
        }
      },
      "analyzer": {
        "rebuilt_persian": {
          "tokenizer": "standard",
          "char_filter": [ "zero_width_spaces" ],
          "filter": [
            "lowercase",
            "decimal_digit",
            "arabic_normalization",
            "persian_normalization",
            "persian_stop"
          ]
        }
      }
    },
    "mappings": {
      "properties": {
        "context": {
          "type": "text",
          "analyzer": "rebuilt_persian"
        }
      }
    }
  }
}

```

سپس به کافکا متصل می‌شویم و داده‌ها را دریافت می‌کنیم. کافکا در یک سرور فیزیکی (VPS) به آدرس 185.236.37.254:9091 قرار گرفته است که به صورت Kafka producer عمل می‌کند و داده‌ها را از کانال‌های مختلف تلگرام دریافت می‌کند و ما نیز به عنوان consumer به آن متصل می‌شویم و داده‌ها

را به صورت بلادرنگ در هر لحظه دریافت می‌کنیم همچنین لازم است برای آن topic ای در نظر گرفته شود که ما به topic ای به نام preProcessedData متصل می‌شویم.

پس از دریافت داده‌ها به هر کدام از آن‌ها فیلد زمان را اضافه می‌کنیم و سپس در elasticsearch ذخیره می‌کنیم. (توجه کنید که زمان نمایش دهنده در کیبانا UTC می‌باشد که 4 ساعت و 30 دقیقه با ایران اختلاف زمانی دارد و همچنین حتی تغییر زمان کیبانا به زمان ایران باز هم این زمان را به صورت UTC نمایش می‌دهد به همین دلیل 4 ساعت و 30 دقیقه از زمان UTC کم شده تا زمان به وقت ایران باشد.)

ابتدا کتابخانه‌ها را import می‌کنیم

```
from kafka import KafkaConsumer
from datetime import datetime, timedelta
from elasticsearch import Elasticsearch
import json
```

سپس تنظیمات مربوطه برای اتصال به کافکا به عنوان consumer را نوشته و به topic مورد نظر متصل می‌شویم و index مربوطه به elasticsearch را مشخص می‌کنیم

```
TOPIC_NAME = "preProcessedData"
INDEX_NAME = "telegram-data"
BOOTSTRAP_SERVERS = "185.236.37.254:9092"

def __init__(self):
    self._es = Elasticsearch()
    self._consumer = KafkaConsumer(DataTransport.TOPIC_NAME, bootstrap_servers=DataTransport.BOOTSTRAP_SERVERS)
```

حال داده‌ها را به صورت لحظه‌ای می‌خوانیم و آن‌ها را به صورت UTF8 انکد می‌کنیم تا بتوان به زبان فارسی خوانده شود و هر پیام را به صورت json می‌خوانیم و فیلد زمان را به آن اضافه می‌کنیم و در elasticsearch ذخیره می‌کنیم:

```

def start_loop(self):
    for msg in self._consumer:
        try:
            msg_value = msg.value.decode('utf-8')
            msg_value = json.loads(msg_value)
            # because of UTC datetime
            timeSubtract = timedelta(minutes=270)
            msg_value["timestamp"] = datetime.now() - timeSubtract
            self._es.index(index=DataTransport.INDEX_NAME, body=msg_value)
            self._es.indices.refresh(index=DataTransport.INDEX_NAME)
            print(msg_value)

```

در زمان نگارش این گزارش در حدود شصت و نه هزار داده جمعآوری شده است:

## Index Management

[Index Management docs](#)

[Indices](#)   [Data Streams](#)   [Index Templates](#)   [Component Templates](#)

Update your Elasticsearch indices individually or in bulk. [Learn more.](#)

Include rollup indices

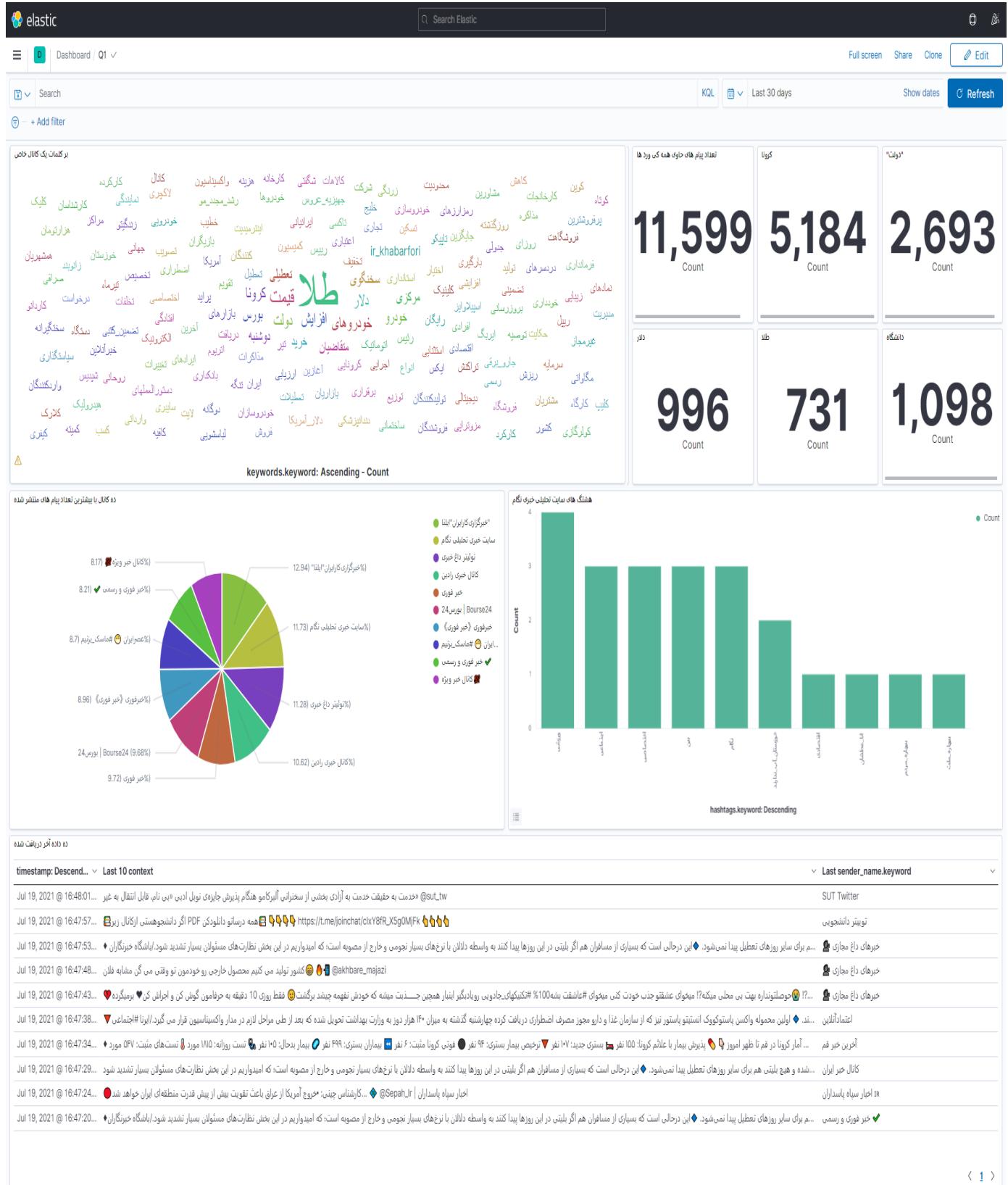
Include hidden indices

[Reload indices](#)

<input type="checkbox"/>	Name	Health	Status	Primaries	Replicas	Docs count	Storage size	Data stream
<input type="checkbox"/>	telegram-data	● yellow	open	1	1	69429	62mb	
<input type="checkbox"/>	my-index-000001	● yellow	open	1	1	2	7.7kb	

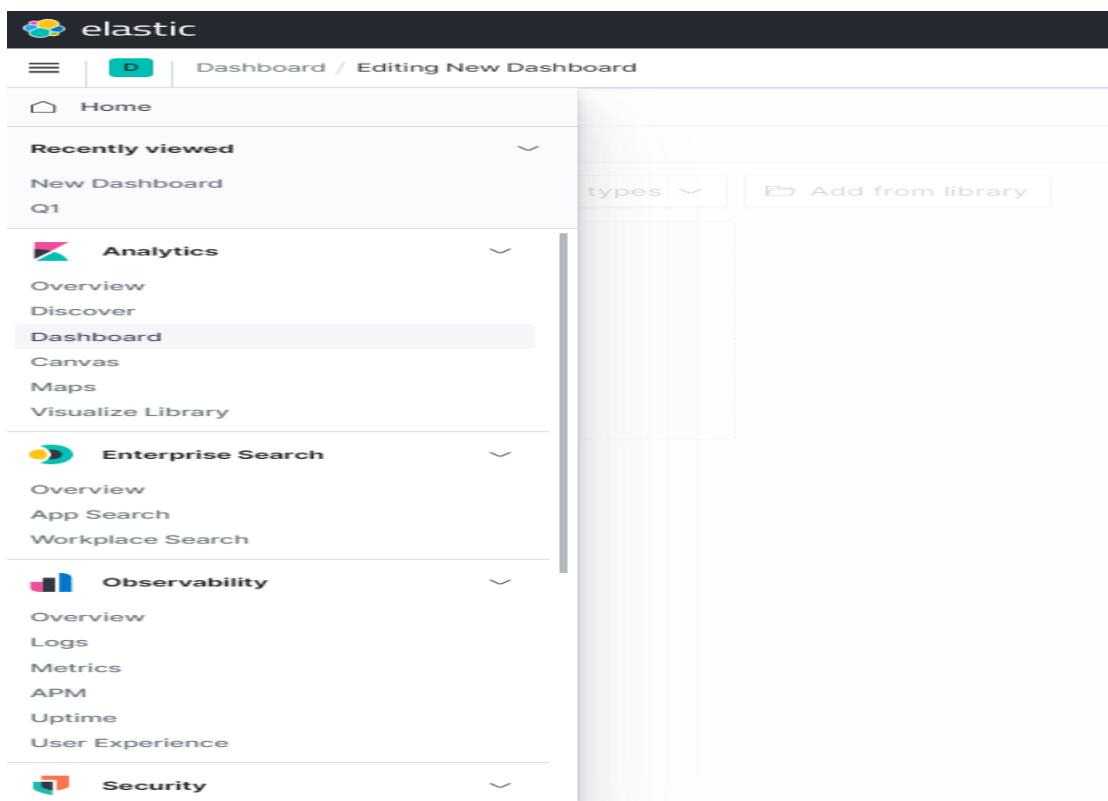
## داشبورد طراحی شده:

نمایی کلی از داشبورد طراحی شده به شرح زیر می‌باشد:



برای ایجاد داشبورد به صورت زیر عمل می‌کنیم:

از منوی سمت چپ گزینه Dashboard را انتخاب می‌کنیم:

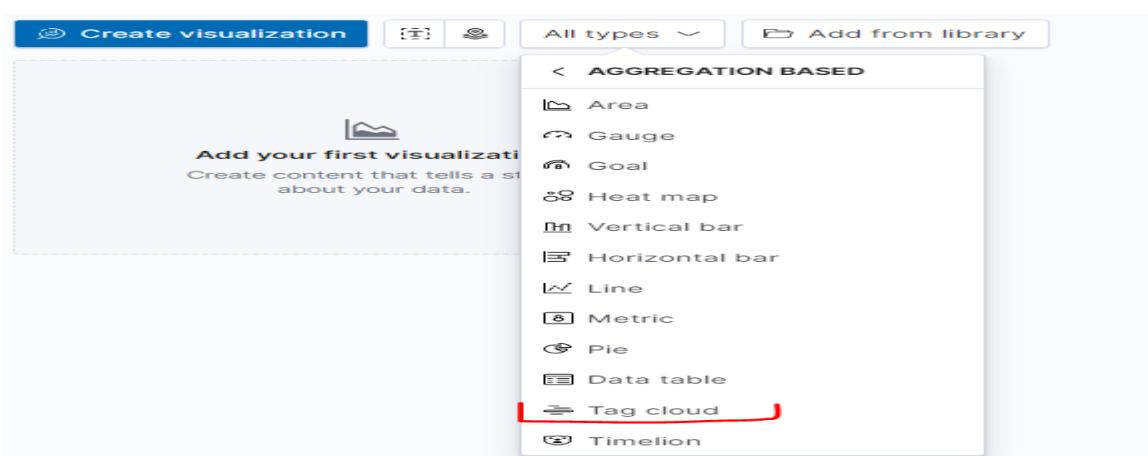


و نمودارهای مربوطه را به آن اضافه می‌کنیم و سپس ذخیره می‌کنیم.

در هر بخش نحوه ایجاد نمودار مربوطه توضیح داده شده است.

## ابر کلمات یک کانال یا خبرگزاری خاص در یک بازه زمانی

برای ایجاد ابر کلمات در دکمه all types بخش AGGREGATION BASED را انتخاب می‌کنیم و Tag cloud را انتخاب می‌کنیم.



سپس index مورد نظر را انتخاب می‌کنیم که در اینجا telegram-data می‌باشد.

## New Tag cloud / Choose a source

< [Select a different visualization](#)

Search...  
telegram-data\*

Sort ▾

Types ▾

برای metric Tag size از معیار count استفاده می‌کنیم و می‌خواهیم که فیلد کلمات کلیدی را برحسب تعداد به صورت نزولی در ابر کلمات نمایش دهد و با توجه به تعداد زیاد کلمات از 500 کلمه استفاده می‌کنیم.

telegram-data\*

Data Options

**Metrics**

Tag size

Aggregation Count

Custom label

**Buckets**

Tags

Aggregation Terms

Field keywords.keyword

Order by Metric: Count

Order Ascending Size 500

Group other values in separate bucket

Show missing values

Custom label

X Discard ▶ Update

برای مشخص کردن نام کانال در قسمت سمت چپ بالا (filter) مشخص می‌کنیم که نام کانال برابر با اخبار فوری، طلا، خودرو باشد و همچنین بازه زمانی بین زمان حال و یک روز قبل از زمان حال باشد.

sender\_name.keyword: اخبارفوری | ارز,طلا,خودرو × timestamp: now-1d to now × + Add filter

**EDIT FILTER**

**Field**: sender\_name.keyword    **Operator**: is

**Value**: اخبارفوری | ارز,طلا,خودرو

Create custom label?

**Cancel** **Save**

sender\_name.keyword: اخبارفوری | ارز,طلا,خودرو × timestamp: now-1d to now × + Add filter

**EDIT FILTER**

**Field**: timestamp    **Operator**: is between

now-1d → now

Create custom label?

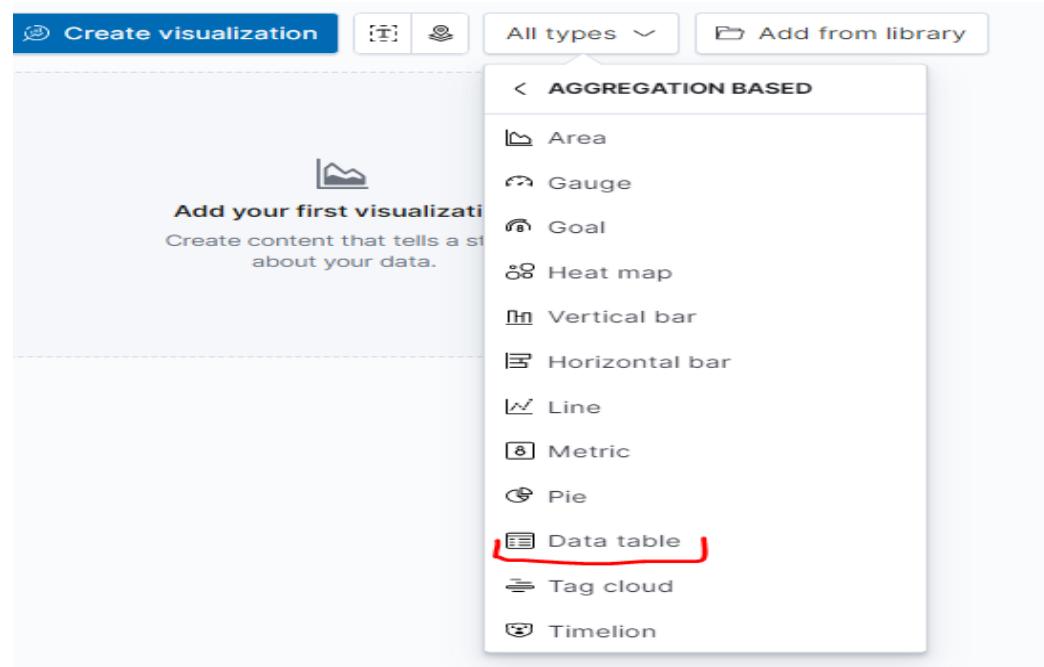
**Cancel** **Save**

ابر کلمات به دست آمده به شرح زیر می‌باشد:

کارکرد کسب کارشناسان  
کارخانه همشهریان فروش هزینه شرایط  
لوسالانی غیرمجاز سنتورالمهای جهیزیه عروس  
وارداتی صرافی رمزارزهای تولیدکنگان تسکین بازاریان پادکاری  
دریس‌های تکنیرات بازار اقتصادی اجرایی اصلاحیه  
گزیده مشاورین سرمایه خطیب خلفات اعتدالی وکیلیه بازارهای بورس  
کیفت مدیریت روزگنشده حکیم ناکس استانداری دولت رییل سخنگوی دلار  
هدیولیک رشد-مجد-مو تطبیقات امضراری کلینیک اومالیک خودروهای افزایش کرونا  
کفیری مدربیت رسمی جایگین ایرانیان آذربین در وقت آخرین خوب تطبیق کرونایی  
کارکرد فرماداری خودروها نتگ بارگیری افرادی آمریکا مذاکرات خودروی اندک  
مشتریان نزدیکی خودروسان نضمن-کنی ایتمدیت ابردهای ایتمدیت نولید  
برداخت لایت سایبری خودروسان خودداری در خواست اینمه  
کلال مگارانی لیاستریو شیپس زیانی سختگرانه کافیه کلومتر  
keywords.keyword: Ascending -

## متن ده پست اخیری که دریافت شده است

برای نمایش متن ده پست اخیر از بخش ALL types نمودارهای از نوع aggregation based را انتخاب می‌کنیم و نمودار Data table را انتخاب می‌کنیم.



از استفاده می‌کنیم که TOP HIT مربوط به متون هر پست را نمایش دهد و در بخش concatenate معیار را با سایز 10 ست می‌کنیم تا آخرین 10 داده را نمایش دهد.

A screenshot of the 'Metrics' configuration panel for the 'Top Hit' aggregation. The panel includes fields for 'Metric' (set to 'Top Hit'), 'Field' (set to 'context'), 'Aggregate with' (set to 'Concatenate'), 'Size' (set to 10), 'Sort on' (set to 'timestamp'), 'Order' (set to 'Descending'), and a 'Custom label' field (which is empty). There is also a 'Top Hit help' link and a 'Top Hit help' button.

همچنین برای هر متن مشخص می‌کنیم که مربوط به چه کانالی می‌باشد.

The screenshot shows the 'Top Hit' aggregation configuration. It includes fields for 'Field' (sender\_name.keyword), 'Aggregate with' (Concatenate), 'Sort on' (timestamp), and 'Order' (Descending). There is also a 'Custom label' field and an 'Advanced' link.

Metric

Aggregation

Top Hit

Field

sender\_name.keyword

Aggregate with

Concatenate

Sort on

timestamp

Order

Descending

Custom label

Size

1

Top Hit help

Advanced

و همچنین مشخص می‌کنیم در چه زمانی ارسال شده است.

### Buckets

The screenshot shows the 'Terms' aggregation configuration for bucketing data. It uses 'timestamp' as the field, 'Alphabetical' as the order, and 'Descending' as the sort order. The size is set to 10. There are options to 'Group other values in separate bucket' and 'Show missing values'. A 'Custom label' field and an 'Advanced' link are also present.

Split rows

Aggregation

Terms

Field

timestamp

Order by

Alphabetical

Order

Descending

Size

10

Group other values in separate bucket

Show missing values

Custom label

Advanced

Add

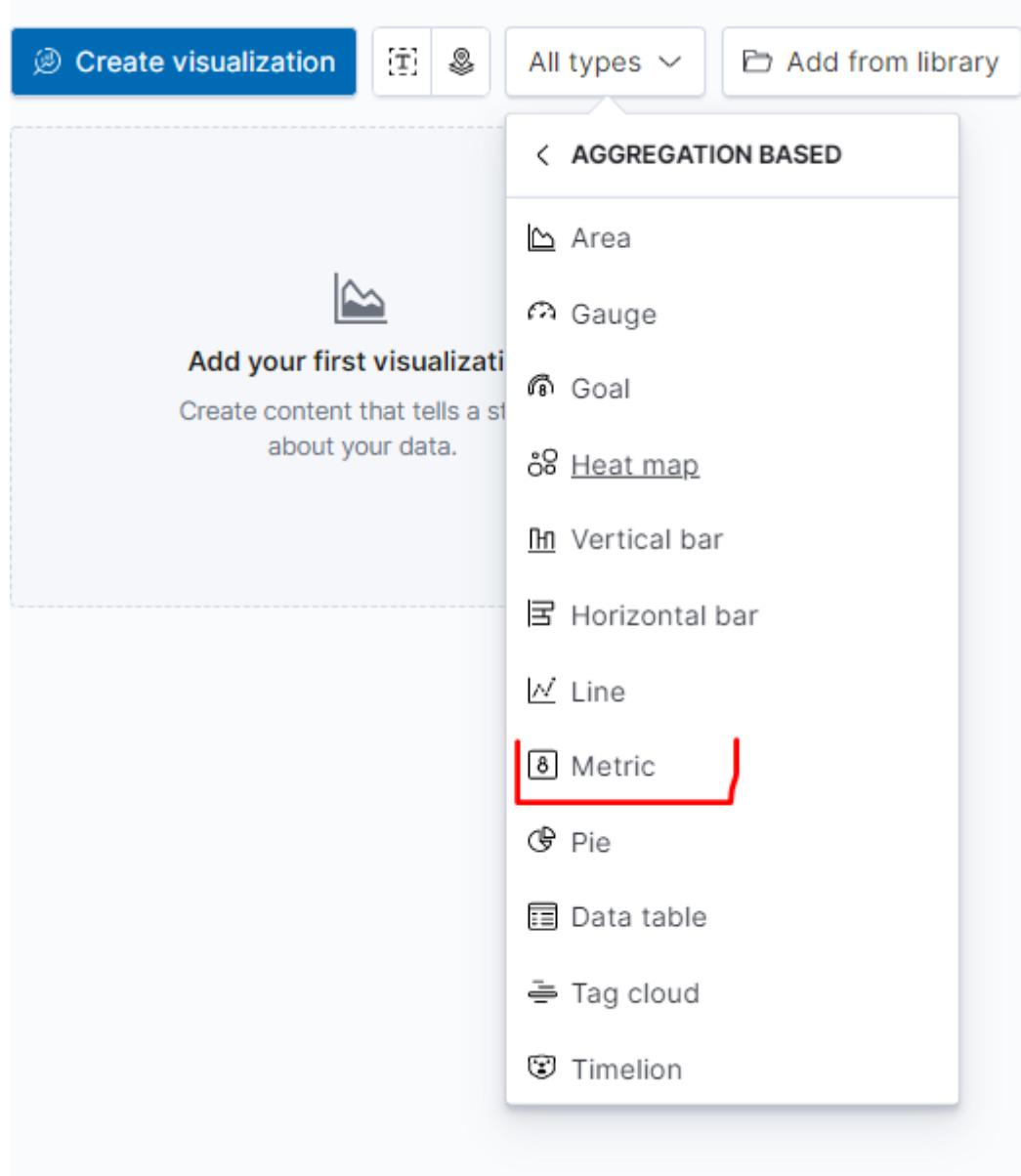
## متن ده پست اخیر دریافت شده به همراه نام کانال و زمان ارسال آن در زیر نمایش داده شده است

۶۰ نامه اخر دریافت شده

timestamp: Descend... ▾	Last 10 context	Last sender_name.keyword
Jul 19, 2021 @ 17:33:39...	# ارزشمندترین کاربردهای گلدن بوی ۲۰۲۱ بر اساس فهمت در تنسیفرمارکت 🔴	Catalan_ir
Jul 19, 2021 @ 17:33:35...	با لگاه خبری انتخاب... می خواهد بود؟ ظاهر آنون ۳۱ ابل در اوخر سال جاری به صورت خواهد بود؟ هچ کس به طور قطعی نموداند اما تعداد زیادی از خبرهای تایید نشده و شایعات در مورد سخت افزار آینده ابل اند روشنی را در مورد انتظارات ما ایجاد کرده است. / انتخاب جزئیات در 🔴	
Jul 19, 2021 @ 17:33:31...	ویداش   Weed_Lash ... موقعی که دارم عیب ریگان رو میگم و ازت بیرون که از شکست و خطاو دیگران خوشحال میشم، دیگنا همون شفقت از مخز فعال میشه که موقع مصرف کوکانن با آمریش جنسی فعال میشه) حالا فیلمید جراحتیت کردن اینقدر حلال بیند؟ 🔴 خواص برت ...	
Jul 19, 2021 @ 17:33:27...	عصرپاریان 📺 #ماسک_بزنیم ... ان لیست ۵۰ هزار شماره تماس فقرز داشته‌اند طبق گزارش "سی ان بی سی" ایک از شخصیت‌های مهم که گوشی او نویسنده این نیوز افشار جاسوسی هد شده است. تازدج جمال خاشقچی مقول که نویسندگان حکومت سعودی در سال ۲۰۱۸ به قتل رسیده بوده است. / ازنا ...	
Jul 19, 2021 @ 17:33:21...	پایگاه خبری عصر سیزور ... ریگار من شود 🔴 آن آرجون با حضور ۱۶۰ داوطلب در ۳۰ جوړه افغانستان ریگار من شود که این تعداد ۱۷۴ نفر خام میباشدند. کارت ورود به جلسه دولطوان، از روز سعینه ۲۹ تیرماه ۱۴۰۰ جهت رویت و چاپ، بر روی سامانه مژک آرمون جهاد دانشگاهی به شناسی 🔴	
Jul 19, 2021 @ 17:33:16...	کانال خبر فوری ساری 📺 میلیون یول زیباناز یوگا! مگه یوگا و اس آرشنی اعصاب نیوادن که خودش اعصاب و روان ادم رو یهه میزیند! 🔴 کانال خبر فوری ساری 🔴	
Jul 19, 2021 @ 17:33:11...	پایگاه خبری عجب شیر نیوز ... ایق پاشایی مدیر کل آموزش و پرورش استان، علیزاده نماینده مراغه و عجب شیر، صدقی رئیس سازمان نوسازی استان، فرماندار امام جمعه و مدیر آموزش و پرورش از بروزه های خبرساز شهرستان عجب شیر بازدید کردند 🔴 عجب شیر نیوز 🟢 سریع 🔴 مستقل 🔴 مردمی 🔴	
Jul 19, 2021 @ 17:33:06...	پایگاه خبری عجب شیر نیوز ... ایق پاشایی مدیر کل آموزش و پرورش استان، علیزاده نماینده مراغه و عجب شیر، صدقی رئیس سازمان نوسازی استان، فرماندار امام جمعه و مدیر آموزش و پرورش از بروزه های خبرساز شهرستان عجب شیر بازدید کردند 🔴 عجب شیر نیوز 🟢 سریع 🔴 مستقل 🔴 مردمی 🔴	
Jul 19, 2021 @ 17:33:01...	رشد ۸ هزار واحد شاخص کل 🔴 شاخص کل بازار بورس تهران ۸ هزار ۴۰ واحد رشد کرد و به عدد بیک میلیون ۳۱ هزار واحد رسید 🔴	NewsNews   اخبار تازه
Jul 19, 2021 @ 17:32:57...	گروه نمایندگی افتخار 🏆 افساط اینا ۵ ساله 🔴 پیش پرداخت متنوع ۴۰-۴۰-۳۰-۳۰٪ جهت اطلاعات پیشتر با ام تماش بگیرید آرaten 🔴 ۰۲۱۸۶۰۳۱۸۶۵ 🔴 ۰۲۱۵۵۹۲۷۹۰۴ 🔴 ۰۲۱۸۶۰۳۱۸۶۲ 🔴 ۰۲۱۸۸۴۸۶۰۹۰ 🔴 ۰۲۱۸۸۴۸۷۰۹۰	NewsNews   اخبار تازه ...

## تعداد پست‌های ارسال شده به ازای چند تا از کلمات کلیدی خاص در یک بازه زمانی

برای نمایش تعداد پست‌ها با ویژگی‌های خواسته شده مانند قبل عمل کرده ولی اینبار نمودار Metric را انتخاب می‌کنیم:



در بخش **Metric** مشخص می کنیم که معیار ما Count می باشد:

## Metrics

### Metric

#### Aggregation

Count help ↗

Count

#### Custom label

[+ Add](#)

در بخش فیلتر مشخص می‌کنیم کلمه کلیدی چه کلمه‌ای باشد در اینجا کلمه کرونا انتخاب شده است به همین ترتیب سایر کلمات را مشخص می‌کنیم:

EDIT FILTER

[Edit as Query DSL](#)

Field	keywords.keyword	Operator	is
Value	کرونا		
<input type="checkbox"/> <a href="#">Create custom label?</a>			
		<a href="#">Cancel</a>	<a href="#">Save</a>

برای مشخص کردن بازه زمانی نیز از فیلتر استفاده می‌کنیم و تعداد این کلمات کلیدی در یک هفته اخیر را نمایش می‌دهیم:

timestamp: now-1w to now

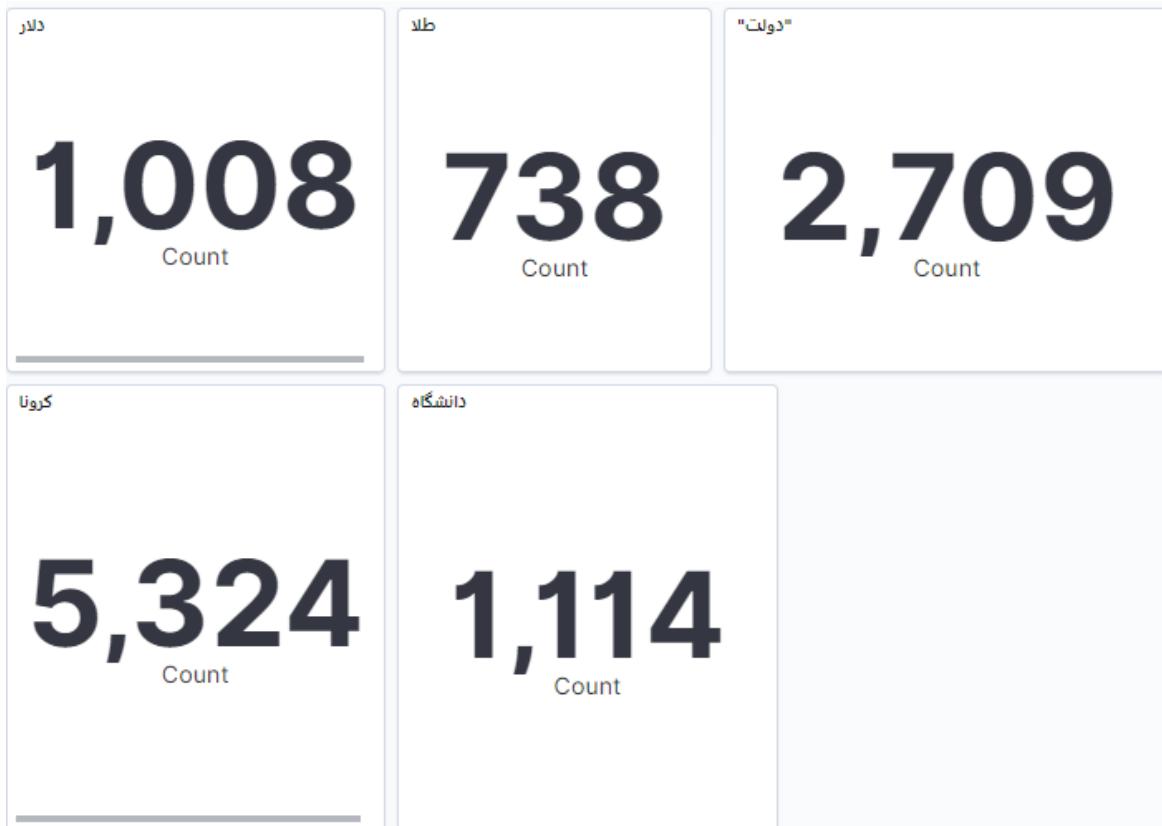
+ Add filter

EDIT FILTER

[Edit as Query DSL](#)

Field	timestamp	Operator	is between
	now-1w	→	now
<input type="checkbox"/> <a href="#">Create custom label?</a>			
		<a href="#">Cancel</a>	<a href="#">Save</a>

خروجی به دست آمده برای برخی از این کلمات کلیدی به شرح زیر می‌باشد:



در صورتی که بخواهیم تعداد تمامی این کلمات کلیدی را پیدا کنیم از فیلتر زیر استفاده می‌کنیم:

keywords.keyword: is one of + Add filter

[Edit as Query DSL](#)

Field	Operator
keywords.keyword	is one of

Values

- boras ×
- اقتصاد ×
- تحریم ×
- دولت ×
- روحانی ×
- انتخابات ×
- دلار ×
- طلا ×
- کرونا ×
- کویند19 ×
- تورم ×
- دانشگاه ×
- حسن روحانی ×

Create custom label?

[Cancel](#) [Save](#)

خروجی آن به شرح زیر می‌باشد:



ده هشتگ بیشتر استفاده شده در یک کانال با تعداد تکرار هر هشتگ و در یک بازه زمانی

برای این بخش از نمودار ستونی استفاده می‌کنیم که vertical bar نام دارد:

The screenshot shows the 'Create visualization' interface with the following elements:

- Search:** A search bar at the top.
- + Add filter:** A button to add filters.
- Create visualization:** A blue button.
- All types:** A dropdown menu.
- Add from library:** A button to add visualizations from a library.
- AGGREGATION BASED:** A category header.
- Vertical bar:** An option highlighted with a red box.
- Horizontal bar:**
- Line:**
- Metric:**
- Pie:**
- Data table:**
- Tag cloud:**
- Timelion:**

برای محور y ها معیار مورد نظر را Count در نظر می‌گیریم:

### Metrics

Y-axis

Aggregation

Count help ↗

Count

Custom label

+ Add

و برای محور y ها مشخص می‌کنیم 10 هشتگ را بر حسب معیار count به صورت نزولی نمایش دهد.

### Buckets

X-axis

⟳ ⚡

Aggregation

Terms help ↗

Terms

Field

hashtags.keyword

Order by

Metric: Count

Order

Size

Descending

10

Group other values in separate bucket

Show missing values

Custom label

> Advanced

در بخش فیلتر بازه زمانی را مشخص می‌کنیم که از سه روز قبل از زمان حال تا زمان حال باشد:

**EDIT FILTER**

[Edit as Query DSL](#)

Field	Operator	
timestamp	is between	
now-3d	→	now

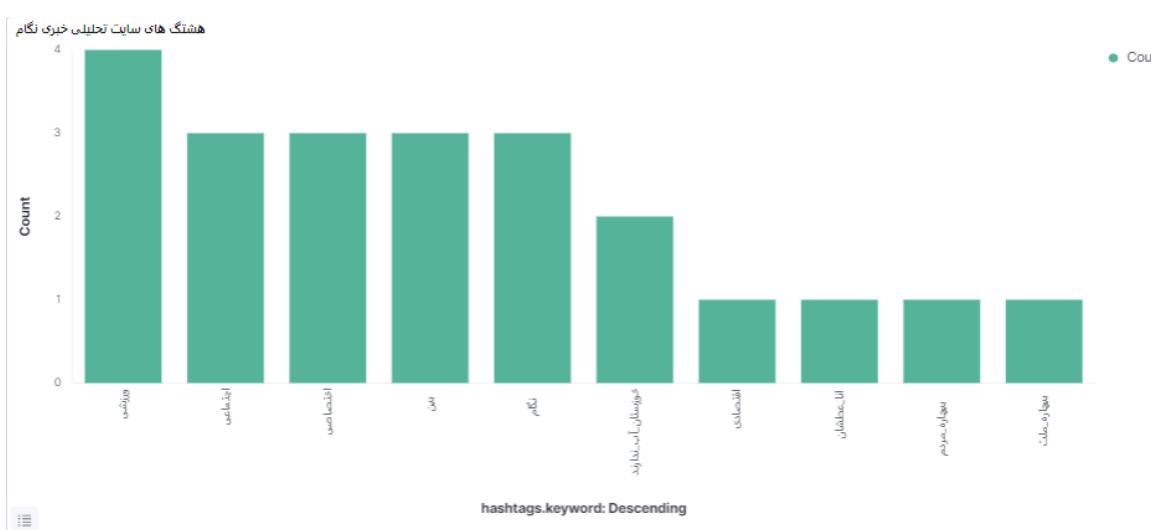
Create custom label?

[Cancel](#) [Save](#)

همچنین در بخش فیلتر مشخص می‌کنیم فرستنده سایت خبری تحلیلی نگام باشد:

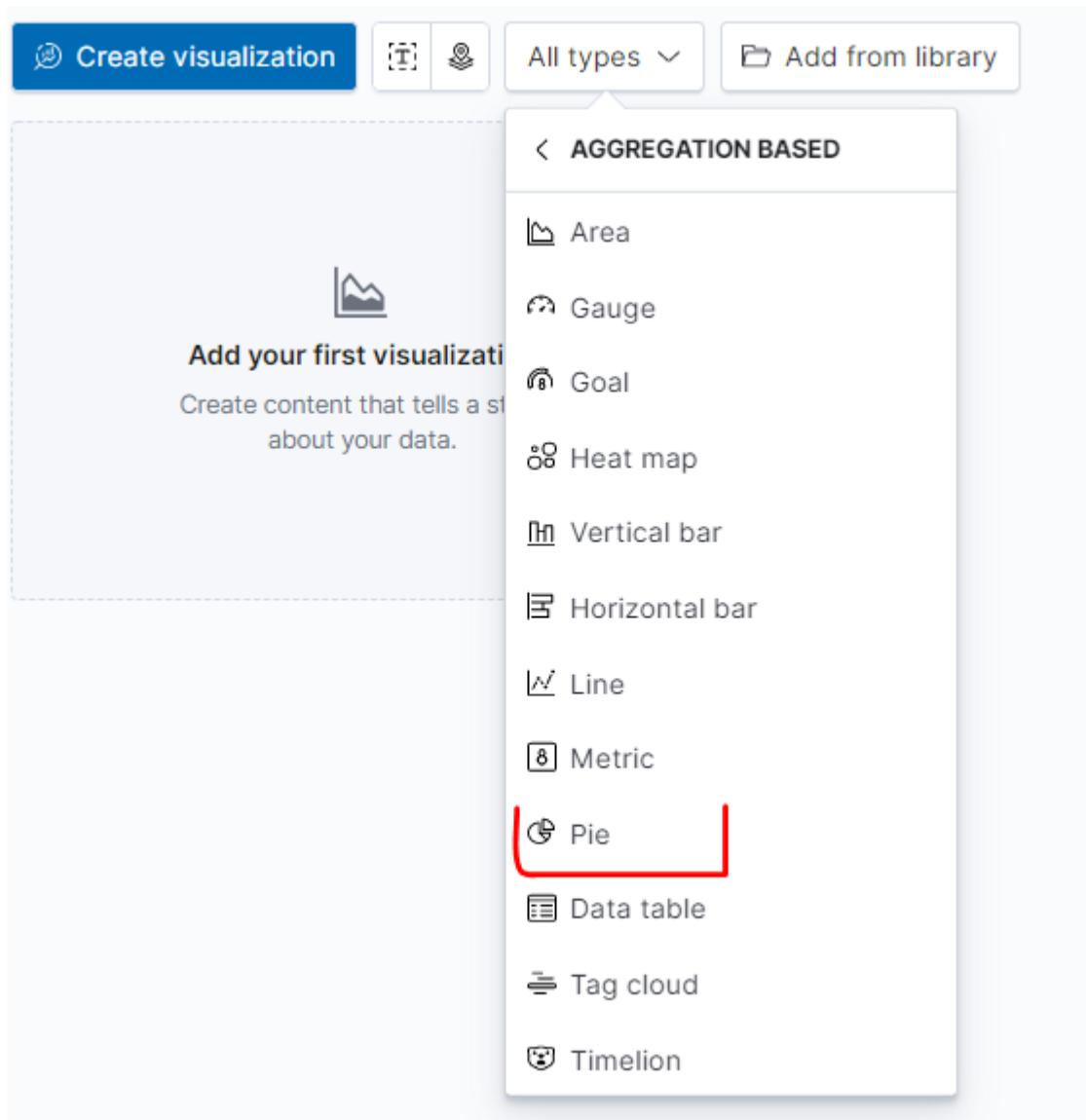


نمودار به دست آمده به شرح زیر می‌باشد که هر ستون نام یک هشتگ را مشخص می‌کند و محور y تعداد آن را مشخص می‌کند:



یک نمودار به انتخاب خودتان (نمودار دایره‌ای تعداد پست‌های هر کانال در 24 ساعت اخیر)

برای این بخش نمودار pie را انتخاب می‌کنیم:



معیار مورد نظر را count انتخاب می‌کنیم:

### Metrics

slice size

Aggregation Count [Count help](#)

Custom label

در این بخش مشخص می‌کنیم بر روی فیلد SenderName که نام کانال را مشخص می‌کند 10 کانال با تعداد بیشتر پست را به صورت نمودار دایره ای نمایش دهد.

### Buckets

split slices [Terms help](#)

Aggregation Terms

Field sender\_name.keyword

Order by Metric: Count

Order Descending Size 10

Group other values in separate bucket

Show missing values

Custom label

> Advanced

همچنین با فیلتر مشخص می‌کنیم داده‌های 24 ساعت اخیر انتخاب شود:

**EDIT FILTER**

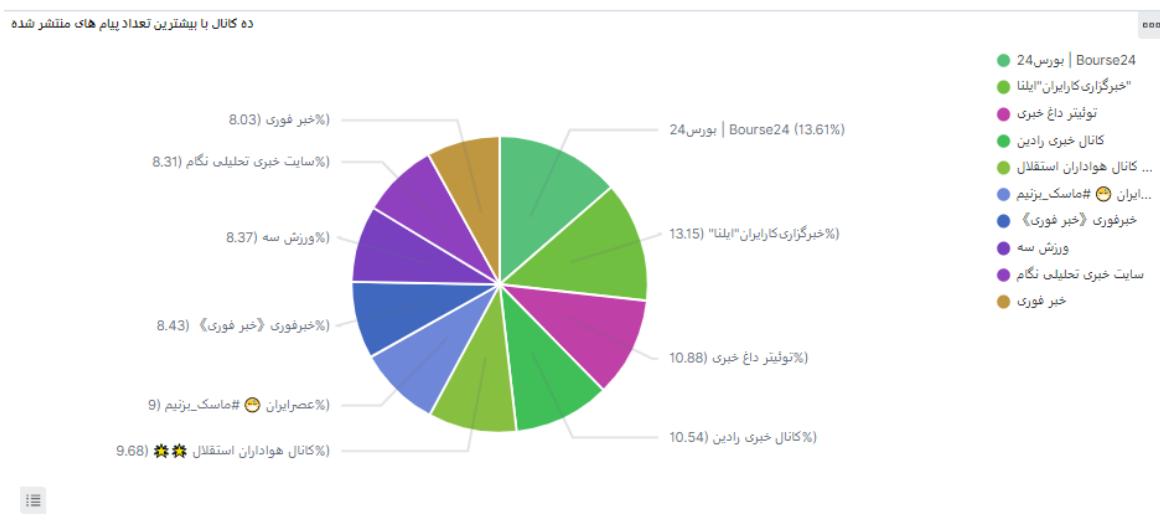
[Edit as Query DSL](#)

Field	Operator	
timestamp	is between	
now-24h	→	now

Create custom label?

[Cancel](#) [Save](#)

نمودار به دست آمده به شرح زیر می‌باشد:



کوئری تمام پست‌های حاوی یک کلمه خاص از یک کanal خاص در یک بازه زمانی:

کوئری مورد نظر به صورت زیر می‌باشد:

```
GET /telegram-data/_search
{
  "query": {
    "bool": {
      "must": [
        {"match": {
          "sender_username": "ilnair"
        }},
        {"match": {
          "context": "کرونا"
        }},
        {
          "range": {
            "timestamp": {
              "gte": "now-1d",
              "lte": "now"
            }
          }
        }
      ]
    }
  }
}
```

طبق این کوئری پیام‌های کانال ilnair انتخاب می‌شود که دارای واژه کرونا باشد و بازه زمانی آنها یک روز گذشته باشد.

بخشی از خروجی به شرح زیر می‌باشد:

```

2 {
3     "took" : 3,
4     "timed_out" : false,
5     "_shards" : {
6         "total" : 1,
7         "successful" : 1,
8         "skipped" : 0,
9         "failed" : 0
10    },
11    "hits" : {
12        "total" : {
13            "value" : 18,
14            "relation" : "eq"
15        },
16        "max_score" : 8.846619,
17        "hits" : [
18            {
19                "_index" : "telegram-data",
20                "_type" : "_doc",
21                "_id" : "rORcvXoBkCjSm8okB3N8",
22                "_score" : 8.846619,
23                "_source" : {
24                    "id" : "12650eb6-ee3a-4f99-af2c-411573298de9",
25                    "context" : "بستری ۱۱۹۸ کودک زیر ۱۰ سال از ابتدای شیوع کرونا"
26                }
27            }
28        ]
29    }
30 }
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

رئیس دانشگاه علوم پزشکی بابل گفت: از زمان شیوع کرونا تاکنون در بابل تعداد ۱۱۹۸ کودک زیر ۱۰ سال بخون بیماران مشکوک و قطعی به کرونا بستری شده.  
در اینجا بخواهید ...

@ilnair""",  
"sender\_id" : 1006298883,  
"sender\_name" : "خبرگزاری کل ایران\اینار",  
"sender\_username" : "ilnair",  
"send\_time" : 1.6266745589522269E9,  
"keywords" : [  
"پزشکی",  
"ابتدا",  
"کودک",  
"کرونا",  
"دانشگاه"  
],  
"hashtags" : [ ],  
"emails" : [ ],  
"timestamp" : "2021-07-19T06:02:42.938760"  
},  
{  
"\_index" : "telegram-data",

کوئری: تعداد پستهای ارسالی به ازای یک کلمه خاص به ازای هر کanal در یک بازه زمانی:

کوئری مورد نظر به صورت زیر می‌باشد:

```
POST /telegram-data/_search
{
  "size": 0,
  "query": {
    "bool": {
      "filter": [
        {
          "range": {
            "timestamp": {
              "gt": "now-1d",
              "lt": "now"
            }
          }
        },
        {"match": {
          "context": "سکم"
        }}
      ]
    }
  },
  "aggs": {
    "group_by_sender_name": {
      "terms": {
        "field": "sender_name.keyword"
      }
    }
  }
}
```

طبق این کوئری داده‌ها باید در بازه زمانی امروز تا روز گذشته باشد و باید در متن پیام واژه سکه وجود داشته باشد و سپس aggregation function اعمال می‌شود تا بر اساس نام کanal گروه بندی شود و تعداد پیام‌های با محتوای سکه در این بازه زمانی به تفکیک گروه به دست آید:

بخشی از خروجی به شرح زیر می‌باشد:

```

{
  "took" : 19,
  "timed_out" : false,
  "_shards" : {
    "total" : 1,
    "successful" : 1,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 74,
      "relation" : "eq"
    },
    "max_score" : null,
    "hits" : [ ]
  },
  "aggregations" : {
    "group_by_sender_name" : {
      "doc_count_error_upper_bound" : 0,
      "sum_other_doc_count" : 46,
      "buckets" : [
        {
          "key" : "قیمت آنلاین دلار، طلا و سکه و خودرو",
          "doc_count" : 5
        },
        {
          "key" : "اخبار، دلار، سکه، خودرو",
          "doc_count" : 4
        },
        {
          "key" : "اخبار فوری | ارز، طلا، خودرو",
          "doc_count" : 4
        },
        {
          "key" : "کانال خبری رادین",
          "doc_count" : 3
        },
        {
          "key" : "اخبار تازه | NewNews",
          "doc_count" : 2
        },
        {
          "key" : "ایران نیوز",
          "doc_count" : 2
        },
        {
          "key" : "ایکوئیتیور - خبرخوان اقتصادی",
          "doc_count" : 2
        }
      ]
    }
  }
}

```

## گام سوم – Channel/Hashtag History (Cassandra)

هدف از این مرحله آن است که داده‌ها را به کمک کاساندرا مکانیزم ذخیره سازی سطر گسترده آن، تاریخچه زمانی هر کanal و هر کلمه کلیدی و هشتگ را در آن ذخیره کرده و سپس اگر کاربر به اطلاعات یک کanal، هشتگ و یا کلمه کلیدی نیاز داشت کافیست داده‌هایی که در این جداول ذخیره شده خوانده شود و به راحتی به آن‌ها دسترسی داشته باشیم. از آن‌جا که این اطلاعات را در هنگام ورود به صورت مرتب شده وارد می‌کنیم و عملیات جوین و اتصال نداریم و افزونگی داده‌ها در کاساندرا یک امر طبیعی است در نتیجه سرعت استخراج داده‌ها بالا است.

در این بخش ما نیاز به `id` هر پست داریم و تنها با استخراج آن و با استفاده از الستیک سرج می‌توان سایر اطلاعات مورد نیاز را بدست آورد.

فایل های موجود برای این دوبخش دو فایل **cassandra** که برای اتصال به کافکا و دریافت اطلاعات و ذخیره سازی آنها در کاساندرا است و فایل **questions** که برای پاسخ به سوالات داده شده است.

## کاساندرا

کاساندرا یک پایگاه داده سطر گسترده است که در این پروژه به عنوان یک ساز و کار کلید اختصاصی برای جستجوی سریع پست ها، کاربران و نمادها استفاده می شود. این پایگاه داده توزیع پذیر برای پشتیبانی از مقیاس پذیری افقی، قابلیت دسترسی بالا و کارایی بالا طراحی شده است.

## نصب و راه اندازی

برای نصب کاساندرا دو راه وجود دارد یا با استفاده از نسخه مستقیم آن را نصب کنیم یا با استفاده از نسخه داکری در اینجا به نحوه نصب هر دو روش اشاره شده ولی خودمان با استفاده از روش مستقیم این کار را انجام داده ایم.

برای نصب نسخه داکری ابتدا باید داکر را اجرا کرده سپس در ترمینال با استفاده از دستور زیر داکر را دریافت می کنیم :

```
(base) soroush@Soroushs-MacBook-Pro ~ % docker pull cassandra ]
```

حال برای اجرای این نسخه به صورت زیر عمل می کنیم :

```
(base) soroush@Soroushs-MacBook-Pro ~ % docker run cassandra ]
```

برای نصب به صورت مستقیم مراحل اندکی بیشتر است ابتدا باید JDK 1.8 را نصب کنیم در ادامه با توجه به اینکه در این سیستم نسخه 16 وجود دارد و به صورت پیش فرض سیستم از این نسخه استفاده می‌کند باید محل استفاده از نسخه جاوا را تغییر دهیم که به صورت زیر عمل می‌کنیم :

```
(base) soroush@Soroushs-MacBook-Pro ~ % java -version  
java version "16.0.1" 2021-04-20  
Java(TM) SE Runtime Environment (build 16.0.1+9-24)  
Java HotSpot(TM) 64-Bit Server VM (build 16.0.1+9-24, mixed mode, sharing)
```

همانطور که در تصویر بالا قابل مشاهده هست نسخه ۱۶ و ۱.۸ بر روی سیستم نصب می‌باشد که با استفاده از دستور زیر نسخه ۱.۸ را برای سیستم تعریف می‌کنیم :

```
(base) soroush@Soroushs-MacBook-Pro ~ % export JAVA_HOME=`/usr/libexec/java_home`  
-v 1.8  
(base) soroush@Soroushs-MacBook-Pro ~ % java -version  
java version "1.8.0_291"  
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)  
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)
```

در ادامه کالساندرا را نصب می‌کنیم :

```
(base) soroush@Soroushs-MacBook-Pro ~ % brew install cassandra
```

حال با استفاده از دستور `cassandra -f` آن را اجرا می‌کنیم :

```
(base) soroush@Soroushs-MacBook-Pro ~ % cassandra -f
```

برای اینکه بتوان از زبان CQL استفاده کرد باید برای سیستم تعریف کنیم که از نسخه ۲ پایتون استفاده کند و سپس با استفاده از دستور cqlsh میتوان ترمینال آماده استفاده کرد :

```
[base] soroush@Soroushs-MacBook-Pro ~ % conda activate py2
[py2] soroush@Soroushs-MacBook-Pro ~ % cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.10 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> ]
```

حال کاساندرا نصب شده و آماده استفاده میباشد. در ادامه به تعریف جداول و استفاده از driver کاساندرا در پایتون خواهیم پرداخت.

با استفاده از دستورات محیط پایتون را آماده استفاده از کاساندرا میکنیم :

```
from cassandra.cluster import Cluster
clstr=Cluster()
session=clstr.connect()
✓ 0.8s
```

حال کاساندرا به پایتون متصل شده و آماده استفاده میباشد. در ادامه توضیحاتی در رابطه با اتصال به سرور کافکا و سپس طراحی جداول داده میشود.

## Kafka

کافکا یکی از محصولات اپاچی است که از آن می‌توان در مساله‌هایی که نیاز به صفت توزیع شده دارند، استفاده کرد. در این پژوهه برای دریافت اطلاعات باید به کافکا در یک سرور فیزیکی موجود در این [آدرس](#) متصل می‌شویم و اطلاعات دریافتی از کانال‌های تلگرامی را دریافت می‌کنیم و سپس آن‌ها را کاساندرا وارد می‌کنیم.

این سرور به صورت kafka producer عمل می‌کند و اطلاعات را به صورت لحظه‌ای از کانال‌های تلگرامی دریافت می‌کند و سپس ما به صورت kafka consumer این اطلاعات را خوانده و براساس نیاز در پایگاه داده‌ها ذخیره می‌کنیم و اطلاعات مورد نیاز را بدست می‌آوریم.

در زیر نحوه اتصال به کافکا و سپس ذخیره اطلاعات در کاساندرا آورده شده است :

```
import json
from kafka import KafkaConsumer
✓ 0.1s

consumer=KafkaConsumer(bootstrap_servers='185.236.37.254:9092')
consumer.subscribe(['preProcessedData'])
✓ 0.2s
```

با استفاده از دستورات بالا به کافکا متصل می‌شویم و یک consumer می‌سازیم که دائما در حال stream کردن داده از سرور می‌باشد. حال برای ذخیره اطلاعات در کافکا از کدی استفاده می‌کنیم که در بخش‌های بعدی آمده است.

## ایجاد KEYSPACE

در ابتدا باید یک KEYSPACE ایجاد کنیم که با استفاده از دستور زیر این کار را انجام می‌دهیم :

```
qry_create_keyspace = '''  
CREATE KEYSPACE "keystore"  
WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : '2'};  
'''  
session.execute(qry_create_keyspace)  
✓ 0.1s  
<cassandra.cluster.ResultSet at 0x7fec88eff220>
```

+ Code + Markdown

```
session = clstr.connect['keystore']  
✓ 0.3s
```

با استفاده از دستورات بالا یک keyspace ایجاد می‌کنیم و سپس کلaster ایجاد شده را به این keyspace متصل می‌کنیم.

## طراحی جداول

در این پروژه برای بازیابی سریع اطلاعات سه جدول طراحی posts, channel, keyword شده‌است. باید دقیق داشت که این سه جدول در یک طراحی رابطه‌ای ممکن است در قالب یک جدول پیاده‌سازی شوند اما در طراحی سطر گسترده هدف ایجاد ساز و کار بازیابی و جستجوی سریع داده‌ها بوده و به همین دلیل افزونگی داده‌ای به شکلی که در طراحی پایگاه داده‌های رابطه‌ای وجود دارد، در اینجا مطرح نیست. باید این نکته افزوده شود که زمان پست‌های استخراج شده به زمان ایران می‌باشد و به همین طریق در کاساندرا ذخیره می‌شوند.

### جدول posts

هدف این جدول ذخیره سازی تمام پست‌های استخراج شده در کاساندرا و مرتب سازی آن بر اساس زمان هر پست می‌باشد سپس با استفاده از این جدول می‌توان پست‌های یک ساعت اخیر، ۲۴ ساعت اخیر و ماهیانه را جمع آوری کرد. این جدول با فیلدهای id, sender (channel id), sendtime, day, month, hour است. برای partition key از ترکیب day, month, hour استفاده شده است که داده‌ها با استفاده از sendtime که یک متغیر از نوع timestamp می‌باشد مرتب می‌کنیم. این جدول به صورت زیر تعریف شده است و سپس کدهای مربوط به آن آورده شده است:

Day	Sendtime	id	sender
Month			
hour			
Partition key		Clustering key	

```

qry_create_table_posts = """
CREATE TABLE IF NOT EXISTS posts(
    id TEXT,
    sender TEXT,
    sendtime TIMESTAMP,
    day int,
    month int,
    hour int,
    PRIMARY KEY((day, month, hour), sendtime)
    WITH CLUSTERING ORDER BY (sendtime desc); """
session.execute(qry_create_table_posts)
✓ 0.1s
<cassandra.cluster.ResultSet at 0x7fec78146790>

```

## جدول channel

هدف این جدول ذخیره سازی پست‌های استخراج شده براساس نام کانال منتشر کننده آن پست در کاساندرا و مرتب سازی آن بر اساس زمان هر پست می‌باشد سپس با استفاده از این جدول می‌توان پست‌های یک کانال در ساعت اخیر، ۲۴ ساعت اخیر و ماهیانه را جمع آوری کرد. این جدول با فیلدهای id, sender partition key (channel id), sendtime است و داده‌ها با استفاده از sender key از timestamp می‌باشد مرتب می‌کنیم. این جدول به صورت زیر تعریف شده است و سپس کدهای مربوط به آن آورده شده است:

sender	sendtime	id
Partition key	Clustering key	

```
qry_create_table_channel = """
CREATE TABLE IF NOT EXISTS channel(
    id TEXT,
    sender TEXT,
    sendtime TIMESTAMP,
    PRIMARY KEY(sender, sendtime))
WITH CLUSTERING ORDER BY (sendtime desc); """
session.execute(qry_create_table_channel)
✓ 0.1s
<cassandra.cluster.ResultSet at 0x7fec984191c0>
```

## جدول keyword

هدف این جدول ذخیره سازی پست های استخراج شده براساس هشتگ یا کلمه کلیدی موجود در متن پیام در کاساندرا و مرتب سازی آن بر اساس زمان هر پست می باشد سپس با استفاده از این جدول می توان پست هایی که شامل یک کلمه کلیدی و یا یک هشتگ مشخص هستند را در ساعت اخیر، ۲۴ ساعت اخیر و ماهیانه را جمع آوری کرد. این جدول با فیلدهای id, sendtime, keyword partition pر شده است. برای timestamp از keyword استفاده شده است و داده ها با استفاده از sendtime که یک متغیر از نوع key می باشد مرتب می کنیم. این جدول به صورت زیر تعریف شده است و سپس کدهای مربوط به آن آورده شده است :

keyword	sendtime	id
Partition key	Clustering key	

حال که جداول خواسته شده طراحی شد. در بخش بعد به سوالات داده شده در صورت سوال پاسخ خواهیم داد.

## ذخیره اطلاعات در جداول

با استفاده از دستور session.prepare و query های لازم اطلاعات دریافت شده را در جداول ذخیره می کنیم. باید به این نکته توجه داشت برای دریافت هرچه بیشتر اطلاعات این قطعه کد باید دائما در حال اجرا باشد تا اطلاعات کانال هایی که توسط سرور دریافت می شود را دائما با استفاده از consumer ایجاد شده بخوانیم.

```
prepared_posts = session.prepare("""INSERT INTO posts (id, sender, sendtime, day, month, hour) VALUES (?,?,?,?,?,?)""")  
prepared_channel = session.prepare("""INSERT INTO channel (id, sender, sendtime) VALUES (?,?,?)""")  
prepared_keyword = session.prepare("""INSERT INTO keyword (id, keyword, sendtime) VALUES (?,?,?)""")  
✓ 0.4s
```

دستورات بالا مورد نیاز برای ذخیره اطلاعات در جداول cassandra می باشد. حال در زیر کد مورد نیاز برای ذخیره اطلاعات دریافت شده آمده است :

```
for msg in consumer:  
    d=json.loads(msg.value.decode('utf-8'))  
  
    id = ''  
    sender = 'Undefiend'  
    sendtime = datetime.now().replace(microsecond = 0)  
    day = 0  
    month = 0  
    hour = 0  
  
    if d['id'] != None :  
        id = d['id']  
  
    if d['send_time'] != None :  
        day = datetime.fromtimestamp(d['send_time']).day  
        month = datetime.fromtimestamp(d['send_time']).month  
        hour = datetime.fromtimestamp(d['send_time']).hour  
        sendtime = datetime.fromtimestamp(d['send_time']).replace(microsecond=0)  
  
    if d['sender_username'] != None :  
        sender = d['sender_username']  
  
    session.execute(prepared_posts, [id, sender, sendtime, day, month, hour])  
    session.execute(prepared_channel, [id, sender, sendtime])  
    if len(d['keywords']) > 0 :  
        for i in d['keywords']:  
            session.execute(prepared_keyword, [id, i, sendtime])  
  
    if len(d['hashtags']) > 0 :  
        for i in d['hashtags']:  
            session.execute(prepared_keyword, [id, i, sendtime])
```

همانطور که در تصویر بالا میتوان مشاهده کرد ابتدا متغیرهای id, sender, sendtime, day, month, hour یک مقدار پیش فرض برایشان تعریف میشود تا در صورتیکه اطلاعات دریافتی این مقادیر را نداشتند کد ما برای اجرا با مشکل مواجه نشود. با توجه به اینکه در جدول channel متغیر sender یک partition است نمیتواند خالی باشد پس در نتیجه مقدار Undefined را برایش انتخاب میکنیم.

## سوالات

در این بخش به سوالات داده شده پاسخ دادهایم. در ابتدا به این موضوع میپردازیم که در هر جدول چه تعداد داده ذخیره شده است.

با توجه به کدهای زیر میتوان مشاهده کرد که در جدول posts و channel به تعداد برابر داده قرار دارد و انتظار همین موضوع را هم داشتیم. در حالی که در جدول keyword با توجه به اینکه تعداد keywordهای هر پست حداقل ۱ عدد هست پس به ازای هر پست حداقل یک داده درون جدول جای میگیرد در زیر میتوان مشاهده کرد :

```
qry_count_post= '''select count(*) from posts;'''
result=session.execute(qry_count_post)
result._current_rows
✓ 0.7s
[Row(count=10972)]
```

```
qry_count_channel= '''select count(*) from channel;'''
result=session.execute(qry_count_channel)
result._current_rows
✓ 0.6s
[Row(count=10971)]
```

[+ Code](#) [+ Markdown](#)

```
qry_count_keyword= '''select count(*) from keyword;'''
result=session.execute(qry_count_keyword)
result._current_rows
✓ 0.1s
[Row(count=47412)]
```

در این بخش به سوالات گفته شده پاسخ خواهیم داد در بخش اول پست‌های یک ساعت گذشته را با استفاده از کد زیر استخراج کرده ایم به راحتی میتوان با توجه به فرمت زیر این بازه را تغییر داد:

```
qry_post= '''select id from posts where sendtime >= '2021-07-20 18:00:00+0000' and sendtime <= '2021-07-20 19:00:00+0000' LIMIT 20 ALLOW FILTERING;'''
result=session.execute(qry_post)
result._current_rows

✓ 0.5s
[Row(id='18ae062c-129a-4b49-89be-f7f46ad5ccc3'),
 Row(id='92012c08-bc9c-427c-a1e1-456dd3f308f7'),
 Row(id='e131c53b-d3a9-446d-9f39-248c030673e5'),
 Row(id='50354317-f5c9-4339-90e0-06ffd80de78'),
 Row(id='9de3fdc7-6b5d-48d8-b9a2-74fcfd530e2c'),
 Row(id='06cd6c4d-e8d6-414c-b5ec-8fb6c569e4b0'),
 Row(id='5df30c36-7f65-4bc4-9df2-49514f1c4f50'),
 Row(id='b90f0cf9-44a2-4ef1-b2a5-66f84978c98c'),
 Row(id='cac63704-6de7-4bca-a262-eb6f86be010c'),
 Row(id='d987a63c-94bf-4ea9-845d-0ac39925a0a9'),
 Row(id='fcfd484-761e-462a-ab2b-f02b20eab5e3'),
 Row(id='95fe1627-1d0f-4e30-8223-54ea766b1791'),
 Row(id='3bfb9fde-1d9a-430f-86dd-c8bb0cc53513'),
 Row(id='a4b9b034-94dd-4411-acd6-d99565acee85'),
 Row(id='c8b673c9-cf84-4514-ba97-5b7c6b107a0f'),
 Row(id='f265a8d5-0c32-4eb0-9ffb-a6ee76e65340'),
 Row(id='a1a53da0-ea3e-40e8-8a62-1bd95189cf37'),
 Row(id='a73c223e-3873-4fef-81ec-dae8b80c1f651'),
 Row(id='2a4c5459-8d50-4dd1-a2c0-897c63651bc2'),
 Row(id='cbaef3f8-5feb-435b-8eec-30e83a805bb7')]
```

در سوال بعد از ما خواسته شده است که پست های ۲۴ ساعت گذشته یک کanal مشخص را استخراج کنیم برای این کار میتوان از دو روش استفاده کرد روش ۱ اینکه بازه را به صورت دستی مشخص کرد روش دوم آن است که با استفاده از دستور `datetime.now()` زمان حال را بدست آورده و ۲۴ ساعت عقب تر را نیز از آن کم کرد ما در این سوال از روش اول استفاده کرده ایم کanal انتخاب شده در این روش به صورت رندم یک کanal با شناسه `Akhbare_tazeh` است که میتوان آن را در زیر مشاهده کرد :

```
qry_channel= '''select id from channel where sender='Akhbare_tazeh' and sendtime >= '2021-07-20 00:00:00+0000' and sendtime <= '2021-07-20 23:59:59+0000'
result=session.execute(qry_channel)
result._current_rows

✓ 0.4s
[Row(id='c8b7d21f-5f30-4c9c-bf65-5330286371dc'),
 Row(id='a3a8f95c-db1e-4665-a8c6-2ab76605d48d'),
 Row(id='7c4a33c4-19a7-45ff-aa53-a54af72a6892'),
 Row(id='030e5ae2-a5ed-4312-8202-089270abc9ab'),
 Row(id='752a52f8-6ba2-4d3b-ad49-e9874f678b07'),
 Row(id='1a9af9c6-bdd4-45d3-be98-3c7afc52157c'),
 Row(id='3e814870-388b-4dd6-a22d-f5c1d0ba4ca0'),
 Row(id='b0d91a49-76cd-4e92-9bf3-61d25cd22dd'),
 Row(id='910b496c-fd92-4135-a82d-e54d1d041d0e'),
 Row(id='2925857c-525f-4b5d-902b-5c7e8941868e'),
 Row(id='1400902d-ffe8-40ac-b33e-8b0370886a5c'),
 Row(id='2b8191a9-f2c9-4cbd-a0e8-4b9cac61ce14'),
 Row(id='e913a64c-ca26-4498-aef5-7bfd4832e113'),
 Row(id='8188d809-e305-4d58-b633-78757c41f764'),
 Row(id='83759dc8-9ef0-4b6f-a8ea-78f5ad99386e'),
 Row(id='ef61e309-dfd1-4188-8e00-11ae777fd137'),
 Row(id='3a3bc68c-33dc-41c8-9beb-69f6a133728e'),
 Row(id='6f8031d1-4863-406d-a8ea-47fc4bc6c5f7'),
 Row(id='da1ad405-c3ee-4c86-a352-7e89d1763172'),
 Row(id='f4808478-795d-4dc2-8db2-4e5d58bbd905')]
```

برای سوال سوم که از ما خواسته شده پست‌های یک هشتگ مشخص را در یک بازه دلخواه بدست آوریم که برای این کار هشتگ فوتبال و بازه زمانی ۲۴ ساعت گذشته انتخاب شده است :

```
qry_keyword= '''select id from keyword where keyword='فوتبال' and sendtime >= '2021-07-20 00:00:00+0000' and sendtime <= '2021-07-20 23:59:59+0000' LIMIT 20'''  
result=session.execute(qry_keyword)  
result._current_rows  
  
✓ 0.4s  
[Row(id='a73c223e-3873-4fef-81ec-dae80c1f651'),  
 Row(id='5cadb9f9-f7f1-4681-895c-589c4e0feb6b'),  
 Row(id='e9b710e5-29db-4e00-9e80-13d68e4ba3b'),  
 Row(id='6287e143-d271-45a4-b766-c150046bb307'),  
 Row(id='9c1af411-031b-403b-8871-7ee86109da4b'),  
 Row(id='adb8b90c-e7c2-4891-99e1-1bcd24ecad13'),  
 Row(id='960702fa-cb49-4adc-ae0f-f20262b2925d'),  
 Row(id='6184df50-e189-4b55-ae18-25bee94ac639'),  
 Row(id='a81172d8-a0bd-4a47-adbc-a1857f40fb74'),  
 Row(id='adc2d575-5eb0-477c-8ca1-eaef034c1991'),  
 Row(id='8457672d-a345-42c9-9d6f-a55a88620955'),  
 Row(id='8d33bb13-80d2-4aa5-b95b-a4b03ae68f1d'),  
 Row(id='3b5378ad-6242-45c8-92f2-98de9202762e'),  
 Row(id='a6b69834-4cd7-4dae-92cd-1a19f7cdf105'),  
 Row(id='5cc37c19-4469-46b3-ad5d-483ee411a832'),  
 Row(id='fb357700-2f3d-44fd-90d8-1233767d5e19'),  
 Row(id='e1330b6d-24a1-4ffa-88fb-c86d42f9cf40'),  
 Row(id='775f33c8-0b6b-4b05-b5a0-334b6a62027e'),  
 Row(id='e15f8a60-9a37-45c6-bbb2-351e07059893'),  
 Row(id='324c490c-c01e-4394-af09-8b9f4e85b031')]
```

+ Code + Markdown

باید به این نکته توجه داشت که در تمامی کوئری های بالا مقدار id برای ما برگشت داده می‌شود و همچنین تعداد موارد بازگشتی را با دستور LIMIT 20 به ۲۰ عدد محدود کرده ایم.

در این قسمت به قسمت دوم سوالات این بخش میپردازیم. در این سوال از ما در رابطه با اطلاعات آماری پرسیده شده است که آیا میتوان با استفاده از کاساندرا اطلاعات آماری را بدست آورد یا خیر. در پاسخ به این سوال باید گفت که هر چند که در پایگاه داده های سطر گسترده نیز می توان اطلاعات آماری را ذخیره کرد اما باید دقت داشت که این پایگاه داده ها برای این کار طراحی نشده اند. به طور خاص کاساندرا برای عملیات های نوشتن (چند صد هزار در ثانیه) و خواندن های سنگین طراحی شده است (البته با محدودیت هایی) اما برای به روز رسانی های زیاد مناسب نیست و البته در سناریوهایی که اولویت نوشتن بالاتر از خواندن است بیشتر استفاده می شود.

برای مثال می توان اطلاعات تعداد پست ها به تفکیک هفته، ماه و سال را برای هر پست در ستون های مجزایی پیش بینی کرد و در بازه های زمانی مشخص مقادیر این ستون ها را به روز رسانی کرد. اما این کار به نظر درست نمی رسد.

کلید تمامی جداول طراحی شده در این پروژه شامل کanal/کلمه کلیدی/پست به همراه یک timestamp زمانی هستند. بنابر این می توان برای هر کanal/کلمه کلیدی/پست، بر اساس زمان بر روی این جداول کوئری ز و آمار بدست آمده را در جای مناسب ذخیره سازی کرد. البته به طور کلی تمام این کوئری ها با در نظر گرفتن ملاحظه های بخش ۳ بایستی طراحی شوند. برای مثال در صورتی که در اکثر اوقات هدف کوئری زدن بر اساس ساعت بوده و تعداد رکوردها در هر ساعت نیز زیاد است، بایستی ترکیب تاریخ و فیلد به عنوان partition key تعریف شود و ساعت در بخش clustering key استفاده شود.



در این بخش با راه اندازی دیتابیس Redis و دریافت اطلاعات از 500 کanal تلگرامی و ذخیره آن در مجموعه های مختلفی بسته به نیاز هر سوال گزارش گیری ها انجام شده است. همچنین برای نمایش اطلاعات از یک وب اپلیکیشن ساده تحت پلتفرم Flask استفاده شده است.

## راه اندازی Redis

برای استفاده از پایگاه داده Redis در این تمرين نسخه ویندوزی آن دانلود و نصب گردید که پس از اجرای redis-server.exe و راه اندازی سرویس ویندوز آن از طریق آدرس محلی <http://127.0.0.1:6379> امکان دستیابی به این پایگاه داده وجود دارد. همچنین برای مشاهده اطلاعات درج شده در پایگاه داده از برنامه های رابط تحت ویندوز مانند Redis Desktop Manager نیز استفاده گردید.

## راه اندازی وب اپلیکیشن Flask

برای نمایش اطلاعات گزارشات دریافتی از Redis یک وب اپلیکیشن بر پایه Flask ایجاد شد که مراحل راه اندازی آن به شرح ذیل می باشد:

### نصب و راه اندازی اولیه Flask

مرحله اول نصب محیط مجازی

```
pip install virtualenv
```

مرحله دوم نصب افزونه ویندوزی محیط مجازی در صورت اجرا بر روی ویندوز

```
pip install virtualenvwrapper-win
```

مرحله سوم ایجاد محیط مجازی برنامه مورد نیاز پروژه

```
mkvirtualenv bigdata
```

با ایجاد محیط اجرا به نام برنامه می توان از برنامه های با ورژن های اختصاصی برای این برنامه استفاده نمود و نصب برنامه ها در این محیط تاثیری بر کل سیستم ندارد و اختصاصی این محیط است.

حال وارد پوشه اصلی وب اپلیکیشن می شویم و با دستور زیر این پوشه را به عنوان پوشه اصلی پروژه تعیین می کنیم ( نقطه در انتهای دستور به معنی پوشه جاری است).

setprojectdir .

می توان با دستور deactivate از محیط فعلی خارج شد و با دستور workon bigdata به محیط اختصاصی برنامه بازگشت.

پس از طی این مراحل نوبت به نصب flask در محیط برنامه می رسد:

pip install flask

حال می توان با ایجاد فایل برنامه تحت وب و اجرای آن با پایتون وب سرور فلسفک را فعال کرد و نتیجه اجرای سایت را بر روی آدرس <http://127.0.0.1:5000/> یا آدرس تعیین شده مشاهده نمود.

قالب صفحه اصلی به صورت زیر طراحی گردید و گزارشات هر بخش در قالب صفحه ای جدید نمایش داده می شود.

## بخش های مختلف پروژه Redis

### هشتگ ها و پست ها

کوئری های آماری پست ها و هشتگ ها

تعداد هشتگ های یکنای یک ساعت اخیر  
هزار هشتگ اخیر

مشاهده صد پست آخر

کل مطالب ذخیره شده

### تعداد پست ها

تعداد پست ها در بازه زمانی مشخص

انتخاب شروع بازه:

2021071704

انتخاب پایان بازه:

2021071705

نمایش نتایج

### گزارش های ساعتی

گزارشات بر اساس زمان خاص

تعداد توبیت های ارسال شده کانال در 6 ساعت گذشته:  
انتخاب کانال:

farsivoa

نمایش نتایج

## دریافت اطلاعات

برای دریافت اطلاعات دو مسیر استفاده گردید. مسیر اول دریافت اطلاعات آرشیو شده بر روی سرور مجازی می باشد که در قالب پایگاه داده MongoDB اطلاعات لحظه ای را آرشیو می نماید و با دستورات زیر اطلاعات از این پایگاه داده دریافت و وارد مرحله پردازش شد:

```
def GetArchiveData():
    try:
        mongoClient = MongoClient('185.236.37.254:9093')
        db=mongoClient.messages

    except Exception as e:
        print(e)

    i=0
    res=db.messages.find()
    for msg in res:
        ProcessData(msg)
        i+=1
        print(i, ' records Added')
```

و در روش دوم اطلاعات لحظه ای از سرویس کافکا در حال اجرا بر روی سرور مجازی دریافت و وارد مرحله پردازش گردید. کد تابع دریافت اطلاعات لحظه ای:

```
def GetOnlineData():
    print('start')
    consumer=KafkaConsumer(bootstrap_servers='185.236.37.254:9092')
    print(consumer)
    consumer.subscribe('preProcessedData')
    i=0
    for msg in consumer:
        d=json.loads(msg.value.decode('utf-8'))
        ProcessData(d)
        i+=1
        print(i, ' records Added')
```

پاسخ به سوالات

1- تعداد پست های یک کanal خاص در 6 ساعت گذشته: برای این کار یک لیست مرتب شده ایجاد شد که به ازای هر دقیقه یک کلید ایجاد و با ارسال هر مطلب مجموعه کanal های ذخیره شده در هر کلید بروز رسانی و عدد تعداد مطالب هر کanal افزایش می یابد.

با استفاده از دستور Zadd و پارامتر incr می توان هم زمان وجود آی دی کanal در لیست و افزایش تعداد مطالب آن را انجام داد. کد این بخش به شرح ذیل می باشد:

```
def SaveMinChannelPostsCount(data):
    username=str(data['sender_username'])
    stime=data['send_time']
    stimeformatted=datetime.fromtimestamp(stime).strftime('%Y-%m-%d %H:%M:%S')
    septime=datetime.strptime(stimeformatted, '%Y-%m-%d %H:%M:%S')
    zaman=datetime.fromtimestamp(stime).strftime('%Y%m%d%H%M')
    key="minChPostCount:"+str(zaman)
    r.zadd(key,{username:1},incr=True)
    r.expire(key,604800)
```

همچنین لیست کanal ها که با کلید جداگانه ذخیره شده اند نیز برای ایجاد منوی کشویی انتخاب نام کanal ذخیره سازی شده اند. کد این بخش به شرح ذیل می باشد:

```
def SaveChannelName(data):
    username=str(data['sender_username'])
    key="ch_name:"+username
    r.set(key, username)
    r.expire(key,604800)
```

برای کنترل تعداد پستهای کanal در شش ساعت گذشته ساعت فعلی سیستم اخذ و با استفاده از کتابخانه timedelta زمان دقیق 6 ساعت قبل محاسبه گردید.

```
def Get6HoursAgo():
    current_time = datetime.datetime.now()
    decr=current_time -Timedelta(hours=6)
    #d=decr.strftime('%Y%m%d%H%M')
    return decr
```

و سپس با استفاده از یک تابع دیگر لیست تمام دقایق بین این دو زمان شروع و پایان محاسبه می گردد.

```

def GetMinRangeTime(start,end):
    listofTime=[]
    listofTime.append(ConvTimetoStr(start))
    while (start<=end):
        start=start+Timedelta(minutes=1)
        listofTime.append(ConvTimetoStr(start))

    return listofTime

```

نتایج این محاسبه در فرمت 2012072012 ذخیره می شود که به ترتیب از سمت چپ سال ماه روز و ساعت می باشد.

حال برای به دست آوردن تعداد پست های کanal منتخب ابتدا لیست کanal ها که در پایگاه داده ذخیره شده بود در فلسک دریافت و در بخش مربوطه از صفحه خانه به صورت لیست کشویی نمایش داده شد.

کد مسیریابی فلسک برای صفحه اصلی و فراخوانی تابع دریافت آی دی کanal ها:

```

***** home page *****
app = Flask(__name__)

@app.route('/',methods = [ 'POST', 'GET'])
def home():
    print('start')
    chns=GetChannelNames()
    hours=GetHourSet()
    #print(hours)
    hoursSorted=sorted(hours)
    print(hoursSorted)
    return render_template("index.html",chns=chns,hours=hoursSorted)

```

این صفحه چون باید نتیجه انتخاب نام کanal را به صفحه بعدی ارسال نماید دارای متدهای Post و Get می باشد.

دريافت اطلاعات آي دى کانال ها از ديتابيس ردیس:

```
def GetChannelNames():
    all=[]
    for key in r.scan_iter("ch_name:*"):
        # delete the key
        msg=r.get(key)
        all.append(msg)
        #print(msg)
    return all
```

اطلاعات کانال ها در صفحه وب به صورت زير نمايش داده شد:

تعداد توييت های ارسال شده کانال در 6 ساعت گذشته

```
<form action="http://localhost:5000/count6hchz" >
  <div class="form-group">
    <label for="sel1">انتخاب کانال:</label>
    <select class="form-control" id="sel1" name="chn">

      {% for channel in chns %}
      <option>{{ channel }}</option>

      {% endfor %}
    </select>
    <button type="submit" class="btn btn-default">نمایش نتایج</button>
  </div>
</form>
```

نمایش تحت وب:

پس از انتخاب و ارسال اطلاعات به صفحه دریافت اطلاعات ابتدا پارامتر انتخاب شده در منوی کشویی به صورت زیر استخراج گردید:

```
@app.route('/count6hchz', methods = [ 'POST', 'GET'])
def count6hchz():
    choice=""
    if request.method == 'POST':
        print('post')
        choice = str(request.form['chn'])
    else:
        choice = str(request.args.get('chn'))
    currentTime=GetCurrentHour()
    decresedTime6h=Get6HoursAgo()
    count=GetCount6HourPostsz(choice,decresedTime6h,currentTime)
    print('count of ',choice,' is ',count)
    data=[]
    data.append(count)
    return render_template("singleData.html", data=data)
```

و زمان فعلی سیستم و زمان 6 ساعت قبل نیز طبق تابعی که قبلاً ذکر گردید محاسبه شده و برای شمارش تعداد پست‌ها به تابع زیر ارسال می‌شود.

```
def GetCount6HourPostsz(choice,decreasedTime6h,currentTime):
    count=0
    minList=GetMinRangeTime(decreasedTime6h,currentTime)

    for t in minList:
        #print(t)
        key="minChPostCount:"+str(t)
        #print(str(r.zscore(key,choice)))
        if r.zscore(key,choice) is not None:
            count+=r.zscore(key,choice)

    return count
```

در این تابع به ازای هر کلید زمان لیست زیرمجموعه آن کلید برای یافتن نام کanal با دستور zscore جستجو و عدد امتیاز آن کanal که عبارت است از تعداد پست‌های آن کanal در آن دقیقه استخراج شده و با تعداد کل جمع می‌گردد. بدین صورت یک بازه 6 ساعته به صورت دقیقه به دقیقه مورد بررسی قرار رفته و تعداد مطالب کanal مورد نظر تجمیع می‌گردد و سپس در صفحه وب مربوطه با پارامتر data نمایش داده می‌شود:

کد صفحه وب:

```
<body dir="rtl">

    <h3>نتایج کوئری</h3>
    <table>
        {% for d in data %}
            <tr>
                <td> {{ d }}</td>
            </tr>
        {% endfor %}
    </table>

</body>
```

نتیجه نمایش اطلاعات کوئری تحت وب:

نتایج کوئری

12.0

2- تعداد کل پست های دریافتی در بازه زمانی مشخص: برای اینکار یک کلید مبتنی بر زمان به صورت سال، ماه، روز، ساعت و دقیقه ارسال پست ها ایجاد گردید که به ازای هر دقیقه یک زیر مجموعه ایجاد می شود از کanal هایی که در آن دقیقه ارسال پست داشته اند و تعداد پست های هر کanal در آن دقیقه به صورت یک لیست دارای مقادیر امتیاز دار با استفاده از دستور `zadd` و پارامتر `incr` ذخیره شدند. (پارامتر `incr` دستور `zadd` را تبدیل به دستور `zincrby` می نماید)

کد مربوط به ذخیره سازی اطلاعات:

```
def SaveMinPostsCount(data):
    stime=data['send_time']
    stimeformatted=datetime.fromtimestamp(stime).strftime('%Y-%m-%d %H:%M:%S')
    septime=datetime.strptime(stimeformatted, '%Y-%m-%d %H:%M:%S')
    zaman=datetime.fromtimestamp(stime).strftime('%Y%m%d%H%M')
    key="minPostCount:"+str(zaman)
    r.zadd(key,{ 'Count':1},incr=True)
    r.expire(key,604800)
```

همچنین با توجه به شرط نگهداری یک هفته ای اطلاعات در دیتابیس دستور `expire` برای حذف خودکار اطلاعات واردشده پس از مدت زمان تعیین شده به ثانیه استفاده گردید که معادل زمانی یک هفته به ثانیه به عنوان پارامتر برای هر کلید تعیین و ثبت شد.

همچنین به جهت سهولت اخذ گزارشات دوره ای ساعت درج مطالب نیز در دیتا بیس ذخیره شده که در زمان انجام محاسبات دوره زمانی خاص از این اطلاعات استفاده می شود و شروع و پایان دوره بر اساس سال ماه روز و ساعت تعیین می گردد.

ذخیره ساعت:

```
def SaveHours(data):
    stime=data['send_time']
    stimeformatted=datetime.fromtimestamp(stime).strftime('%Y-%m-%d %H:%M:%S')
    septime=datetime.strptime(stimeformatted, '%Y-%m-%d %H:%M:%S')
    zaman=datetime.fromtimestamp(stime).strftime('%Y%m%d%H')
    print(data,zaman)
    key="hourset:"+str(zaman)
    r.set(key, zaman)
    r.expire(key,604800)
```

و در زمان بارگزاری صفحه اصلی این اطلاعات این در بخش مسیر یابی فلسفک دریافت و در فیلد مربوطه در صفحه وب نمایش داده می شود:

```
@app.route('/',methods = [ 'POST', 'GET'])
def home():
    print('start')
    chns=GetChannelNames()
    hours=GetHourSet()
    #print(hours)
    hoursSorted=sorted(hours)
    print(hoursSorted)
    return render_template("index.html",chns=chns,hours=hoursSorted)
```

دریافت ساعت:

```
def GetHourSet():
    all=[]
    for key in r.scan_iter("hourset:*"):
        msg=r.get(key)
        all.append(int(msg))
        #print(msg)
    return all
```

نمایش در صفحه تحت وب:

بخش های مختلف پروژه Redis

**هشتبگ ها و پست ها**

کوئری های آماری پست ها و هشتبگ ها

تعداد هشتبگ های یک ساعت اخیر  
هزار هشتبگ اخیر

مشاهده صد پست آخر

کل مطالب ذخیره شده

**تعداد پست ها**

تعداد پست ها در بازه زمانی مشخص

انتخاب شروع بازه:

- ▼ 2021071701
- ▲ 2021071701
- 2021071702
- 2021071703
- 2021071704
- 2021071705
- 2021071706
- 2021071707
- 2021071708
- 2021071709
- 2021071710
- 2021071711
- 2021071712
- 2021071713
- 2021071714
- 2021071715
- 2021071716
- 2021071717
- 2021071718
- 2021071719
- 2021071720

**گزارش های ساعتی**

گزارشات بر اساس زمان خاص

تعداد توییت های ارسال شده کانال در 6 ساعت گذشته

انتخاب کانال:

- ▼ goldooneman

نمایش نتایج

## کد صفحه :html

```
<form action="http://localhost:5000/countTime" >
  <div class="form-group">
    <label for="sel2">انتخاب شروع بازه</label>
    <select class="form-control" id="sel2" name="startt">
      {% for h in hours %}
        <option>{{ h }}</option>
      {% endfor %}
    </select>
    <label for="sel3">انتخاب پایان بازه</label>
    <select class="form-control" id="sel3" name="endt">
      {% for h in hours %}
        <option>{{ h }}</option>
      {% endfor %}
    </select>
    <button type="submit" class="btn btn-default">تمامیش تایم</button>
  </div>
</form>
```

اطلاعات این صفحه پس از انتخاب به لینک ابتدای فرم ارسال می گردد و در این صفحه اطلاعات دریافت و تعداد کل پست ها در بازه تعیین شده با به دست آوردن لیست کل دقیقه های بین دو زمان شروع و پایان به عنوان کلید جستجو و استخراج اطلاعات از ردیس و تجمع آن به دست آمد.

کد مسیر یابی و اخذ اطلاعات بازه زمانی:

```
#=====
part 2 count of posts in range

@app.route('/countTime', methods = [ 'POST', 'GET'])
def countTime():
    choice=""
    if request.method == 'POST':
        start = str(request.form['startt'])
        end = str(request.form['endt'])
    else:
        start = str(request.args.get('startt'))
        end = str(request.args.get('endt'))

    tcount=GetRangePostCount(start,end)
    print(start,end,tcount)
    data=[ ]
    data.append(tcount)
    return render_template("singleData.html", data=data)
```

کد تابع محاسبه تعداد پست ها در این بازه:

```
def GetRangePostCount(start,end):
    sdate=datetime.datetime.strptime(start, '%Y%m%d%H')
    print(start,sdate)
    edate=datetime.datetime.strptime(end, '%Y%m%d%H')
    print(end,edate)
    mList=GetMinRangeTime(sdate,edate)
    print(mList)
    count=0
    for t in mList:

        key="minPostCount:"+str(t)
        print(t,r.zscore(key,'Count'))
        if r.zscore(key,'Count') is not None:
            count+=r.zscore(key,'Count')

    return count
```

که نتیجه طبق بازه انتخابی (به عنوان نمونه برای یک روز گذشته) نمایش داده می شود:

نتایج کوئری

26346.0

3-تعداد هشتگ های دریافت شده در یک ساعت گذشته

برای به دست آوردن تعداد هشتگ های یکتا در یک ساعت اخیر هشتگ ها با کلید زمانی شامل سال، ماه، روز، ساعت و دقیقه ارسال پست با نام متغیر اولیه MinHashtagSet ذخیره شدن.

```
def SaveMinHashtags(data):

    stime=data['send_time']
    stimeformatted=datetime.fromtimestamp(stime).strftime('%Y-%m-%d %H:%M:%S')
    septime=datetime.strptime(stimeformatted, '%Y-%m-%d %H:%M:%S')
    zaman=datetime.fromtimestamp(stime).strftime('%Y%m%d%H%M')
    hashtags=data['hashtags']
    print(hashtags)
    for hasht in hashtags:
        print(hasht)

        key="MinHashtagSet:"+str(zaman)
        r.sadd(key,hasht)
        r.expire(key,604800)
```

به ازای هر دقیقه یک کلید حاوی مجموعه ای از هشتگ ها ذخیره شد.

سپس برای بازیابی هشتگ های یک ساعت قبل ساعت فعلی سیستم اخذ و زمان یک ساعت قبل نیز محاسبه شد و سپس لیست کل دقیقه های بین این دو زمان به عنوان کلید جستجو استخراج و اطلاعات هشتگ های هر کدام از این دقیقه ها با دستور smembers از ردیس استخراج و با کنترل عدم تکراری بودن در لیست ذخیره شد:

```
#===== part 3 count unique hashtags in 1 hour

def GetLastHourUniqueHashtags():
    current_time=datetime.datetime.now()
    lastHour=current_time-Timedelta(hours=1)
    minList=GetMinRangeTime(lastHour,current_time)
    allhourHashtags=[]
    for t in minList:

        key="MinHashtagSet:"+str(t)
        minhash=r.smembers(key)
        print(t,minhash)
        for mh in minhash:
            if mh not in allhourHashtags:
                allhourHashtags.append(mh)

    count=len(allhourHashtags)
    print(allhourHashtags)
    print(count)
    return count
```

سپس این لیست که از فراخوانی در بخش مربوطه در مسیریابی فلسفک دریافت شده است در صفحه وب نمایش داده می شود:

```
<body dir="rtl">

    <h3>نتایج کوئری</h3>
    <table>
        {% for d in data %}
            <tr>
                <td> {{ d }}</td>
            </tr>
        {% endfor %}
    </table>

</body>
```

نتیجه اجرا:

نتایج کوئری

58

4- آخرین هشتگ های دریافت شده شامل یک لیست هزارتاوی که با ورود داده های جدید داده های قدیمی حذف می شوند.

برای دریافت این گزارش یک لیست از هشتگ ها ایجاد می شود که در هر بار دریافت اطلاعات هشتگ های مطلوب در ابتدای این لیست با دستور lpush درج شده و با استفاده از دستور ltrim هزار داده اول لیست نگه داری می شود و بقیه داده ها از لیست حذف می شود.

کد ذخیره هشتگ ها و حذف هشتگ های قدیمی بیشتر از هزار عدد:

```
def SaveHashtaginList(data):
    hashtags=data[ 'hashtags' ]
    #print(hashtags)
    for hast in hashtags:
        #print(hast)
        key="ListOfHashtags>List"
        r.lpush(key,hast)
        r.ltrim(key,0,999)
```

کد بازیابی و نمایش هشتگ ها:

```
@app.route('/last1000hashtags')
def last1000hashtags():
    hashtags=GetLast1000Hashtags()
    #print(hashtags)
    #print(type(hashtags))
    #print(type(allmsg))

    return render_template("singleData.html", data=hashtags)
```

تابع مربوطه برای فراخوانی اطلاعات از ردیس:

```

def GetLast1000Hashtags():
    key="ListOfHashtags:List"
    listofH=r.lrange(key,0,999)
    return listofH

```

تعداد داده های ذخیره شده در لیست پایگاه داده:

nw::db0::ListOfHashtags:List		Size: 1000 TTL: -1 Rename
row	value	
992	ضباءآباد_خ...	
993	زبان	
994	کره	
995	اجتماعی	
996	خبرهای_داغ_ مهم	
997	خبرهای_داغ_ مهم	
998	خبرهای_داغ_ مهم	
999	خبرهای_داغ_ مهم	
1000	سیستان_بلوچستان	

نتایج اجرای وب پیج مربوطه:

### نتایج کوئری

خوزستان_مظلوم
فاکس_نبور
فاکس_نبور
فوری
روزنامه
تعویم
تعویم
طرح
مطبوعات
حدیث_روز
فوتبالی_کات
فوتبالی_کات
پیج_ما_در_اینستاگرام

## 5- آخرین پست های ارسال شده در کانال ها در یک لیست صد تایی:

برای این بخش نیز مشابه حالت بالا یک لیست ایجاد می شود جهت ذخیره اطلاعات پست های ارسالی در کانال ها و در هر بار درج مقدار جدید ردیف های بیشتر از صد از لیست حذف می شود.

درج اطلاعات پست ها و حذف مازاد بر صد پست:

```
def SavePostinList(data):
    text=str(data['context'])
    key="ListOfPosts:List"
    r.lpush(key,text)
    r.ltrim(key,0,99)
```

فراخوانی اطلاعات از ردیس:

```
def GetLast100posts():
    key="ListOfPosts:List"
    listofP=r.lrange(key,0,99)
    return listofP
```

کد بخش مسیریابی فلسفک:

```
@app.route('/last100posts')
def last100posts():
    posts=GetLast100posts()

    return render_template("singleData.html", data=posts)
```

اطلاعات ذخیره شده در دیتابیس ردیس:

row	value	Size: 100 TTL: -1	Rename	Delete	Set TTL
87	@kermanshah19				
88	محتواهای فیبر...				
89	دستور دشنه...				
90	\xE2\x9A\xBD\x...				
91	\xF0\x9F\x94\xB...				
92	فوبیا کات...				
93	\xD8\xA7\xD8...				
94	\xD8\xAA\xD9\x...				
95	\xD8\xA4\xD9\x...				
96	فوبیا کات...				
97	\xF0\x9F\x94\xB...				
98	فوبیا کات...				
99	\xE2\x99\xA6 ...				
100	\xE2\x9A\xA1\x...				

Value: size: 426.00 bytes [Binary]

ویدئوهای رسیده به صدای آمریکا حاکی از تداوم اعتراضات علیه بی‌آبی در خودستان برای روز دوم است. کریم دحیمی، فعال حقوق بشر ساکن بریتانیا که کارشناس مسائل جامعه عرب و آشنا به ملطفه است انتساب ویدئوها به روز جمعه ۲۵ تیر را تائید نمی‌کند.

## نتایج اجرای وب پیچ:

### نتایج کوئی

▲ اعصاب باریکات استغلال؛ بدقولی مدیران تمرین امروز آنها را محل کرد.♦ تمرین نیم فوتیان استغلال امروز بدبان اعصاب باریکات این نیم در برداخت مطالبات اشان حاضر به احجام تمرین سیستم و تعزیز نکردند. @khabaritel
ناره همون به کدهم رو هم حوابیده میزید. *کسول* sut_tw@ sut_tv @alf_rssin_tw
امرا علاوه با کاهه حلی بیاست. » ب الف رسین الف «
اصدوارم به بواه شناسایی قالب هم طواب خوشیان، هموطیان بگزید بی دلیل محاکمه نشود؛ درخواست می کنیم اکر شخص با اشخاصی بایت اتهام قتل بارداشت شدند، اجازه دهدید از همان ایندا و کل انتخابی داشته و مردم از جزیات مسائل حساس میهمایم. Babak Paknia @sut_tw https://www.clubhouse.com/join/%D8%B3%D8%A7%D8%AA%7 : سهله بادیاران، آنچه باید و نباید بدانیم!! عنوان اینست اف سایت حری بحالی نکام در کتاب هاوس ساخت ۷۷ به وقت این روز با حضور کارشناسان و مهندسان ورق اینست باید بروز و سخنورد : @AF%D8%A7%D9%86%D9%AF%D8%A7%D9%85%5CcBgPj/MED7oJWa
● هفده مجدد در حاضر تمرین امروز استغلال؛ اسپاسر باشکاه بیعنی هاکوب به نیال آوردن هری بگزید بود. حمایت از باریکات و مددی هم موافق این موضوع بودند اما سلطانی فرم موافقت نکرد.♦ تمرین نیم است و ناید عقده های شخصی را در این احرار کرد، اکر با من منسلک دارد این باریکات و هدایات را که کاهی کردند. همچنان که درخواست ایندا بعد از برد در دری یک تیرک هم به من نکنند، مددی حینی یک تعاس هم نکرفت.♦ من حینی دامن برای باری سا الهال کدام باریکات را در اختیار دارم مراجعاً گویم افانی منظر باخت استغلال هستند. varzeshs3@ varzeshs سه استغلالیها اعصاب کردند باریکات و کادری استغلال به دلیل مشکلات مالی، روحی و روانی به وجود آنده در این نیم اعصاب کرده و خاصر به تمرین نشندن. Yjc @OfficialPersianTwitter
▲ اعصاب باریکات استغلال؛ بدقولی مدیران تمرین امروز بدبان اعصاب باریکات این نیم در برداخت مطالبات اشان حاضر به احجام تمرین سیستم و تعزیز نکردند. @irannews@ irannews@
♦ وزیر نور؛ شرمende مردم سیستم وزیر نور؛ خوشخانه همه و احدهای تویید بر ق شفورد در مدار هستند. همچنان که اسپاسر است، به مددی که درخواست اینها خود مردم هستند که درخواستها باخت نیفت دادند. ما هم شرمende مضافی سیستم که باید اینجا ایجاد شده و همچنین ممنوع این ارتقای هستیم که وجود داشته و درخواستها باخت نیفت دادند. radinnews@ Charanndism@ havadarn_Esteghlal@
♦ لزله بخششواری از اسنایهای فارس و بوشهر را تکاب داد #فاکس_نور Foxnewsirtv@ Foxnewsirtv@
#قریب لزله ای به بزرگ ۰.۷ ریترین اسنای فارس را ازبرد [فاکس_نور] #فاکس_نور @ havadarn_Esteghlal@
سروبرین استغلال در مورد اینکه به مشکل این است که من اصول را رعایت می کنم، نظم و انصاف ای اولت من است. فرشاد ماجدی دوست ۲۰ ساله من است اما انت مسابقه بیون اجازه هنل ترک کرد و به او گفتم دیگر بس تمرین باید، من با برادرم هم شوچی ندازم ۴۵ روز دیگر با الهال در لیگ فوتبال اسپا بایر کیم اما هنوز فرارداد برسی باریکات تعذیر شده باید بایم که من خواهد فوهران شد و باید برگایه داشت اما من حینی دامن باری با الهال کدام باریکات را در اختیار دارم، افانی من خالیاتش هم نیست، مددی حینی یک تعاس هم نکرفت، صراحتاً گویم افانی منظر باخت استغلال هستند و با فاقضت من گویم انها بر داد ما در دری چوشنحال بینند، ادامه دار.... اسنایور دید مکالمه اینستی Perspolis_FC_Tehran@ sandalikhabar@
باشون خسارتی از زمین لرزه را کارشناسی شدند تا آن را بر روی سیستمان به کمک دستور زیر فعال کنیم :

## گام پنجم clickhouse-

### ClickHouse

پس از نصب ClickHouse لازم است تا آن را بر روی سیستمان به کمک دستور زیر فعال کنیم :

```
sudo service clickhouse-server start
```

برای ارتباط با ClickHouse ، driver پایتونی مناسب را نصب و سپس یک client میسازیم تا

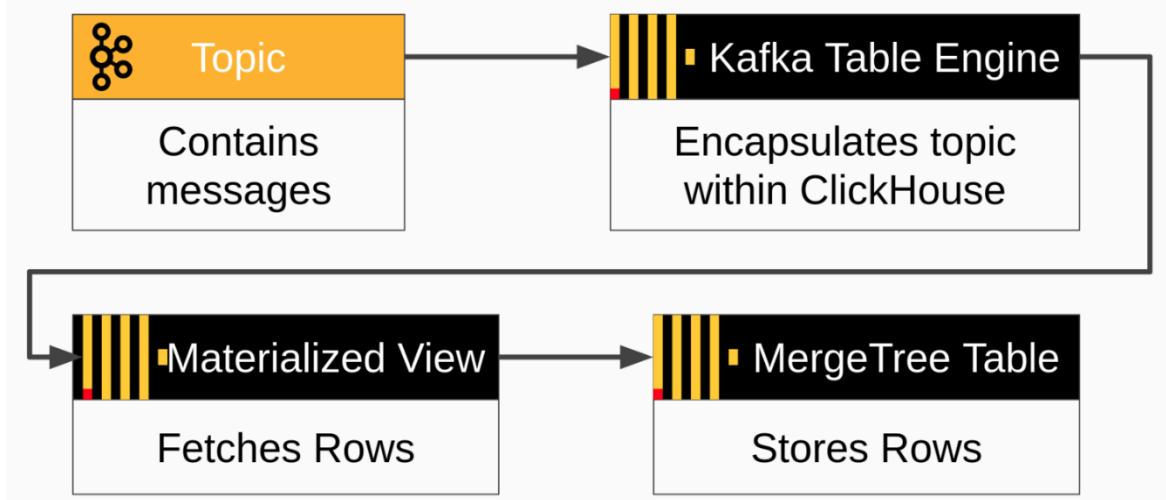
توانیم دستورات مورد نظرمان را در ClickHouse اجرا کنیم

```
1 from clickhouse_driver import Client
2 client = Client(host='localhost', password='bahar1375')
3 client.execute('SHOW DATABASES')

[('default',), ('system',)]
```

سپس در این مرحله لازم است تا داده ها را به نوعی از kafka بخوانیم و در داخل ClickHouse در ساختار مناسب ذخیره کنیم.

برای این کار روش از ساختار زیر استفاده میکنیم:



1. ابتدا تیبل مورد نظرمان در ClickHouse را ایجاد میکنیم :

```
1 client.execute(' DROP TABLE IF EXISTS default.posts')
2 client.execute(' CREATE TABLE IF NOT EXISTS default.posts \
3 (\
4     ID UUID,\n
5     sendTime DateTime,\n
6     senderName String,\n
7     senderUsername String,\n
8     keywords Array(String),\n
9     hashtags Array(String)\n
10 ) ENGINE = MergeTree()\n11 PARTITION BY toYYYYMMDD(sendTime)\n12 ORDER BY (sendTime);')
```

در این تیبل همه داده ها به جز متن پیام را ذخیره میکنیم و پارتیشن بندی را بر اساس تاریخ به فرمت سال-ماه-روز انجام میدهیم. mergeTree Engine را تعریف میکنیم که مناسب برای ذخیره سازی داده ها با حجم زیاد است.

2. سپس یک تیبل دیگر میسازیم تا اطلاعات را از kafka به شکل مستقیم دریافت کند برای این کار کافی است تا kafka قرار دهیم و اطلاعات تاپیک مورد نظرمان را در آن مشخص کنیم . همه داده هایی که از kafka شنیده میشود را در قالب یک string ذخیره میکنیم .

```

1 client.execute(' DROP TABLE IF EXISTS readings_queue ')
2
3 client.execute("CREATE TABLE readings_queue  (
4     message String,
5     ) ENGINE = Kafka('185.236.37.254:9092', 'preProcessedData', 'group1', 'JSONAsString')");
6

```

3. سپس از تیبل readings\_queue یک view تعریف میکنیم، داده های مورد نظرمان را از ورودی استخراج میکنیم و ماحصل را در تیبل اولیه که ایجاد کرده ایم میریزیم .

```

1 client.execute(' DROP TABLE IF EXISTS posts_mv ')
2
3 client.execute("CREATE MATERIALIZED VIEW posts_mv TO posts AS SELECT \
4     toUUID(JSONExtractString(message, 'id')) AS ID ,\
5     JSONExtractString(message, 'sender_name') AS senderName ,\
6     JSONExtractString(message, 'sender_username') AS senderUsername ,\
7     toDate(FROM_UNIXTIME(toUInt64(JSONExtractFloat(message, 'send_time')), '%Y-%m-%d %R:%S')) AS sendTime, \
8     JSONExtractArrayRaw(message, 'hashtags') as hashtags, \
9     JSONExtractArrayRaw(message, 'keywords') as keywords \
10    FROM readings_queue")

```

نمونه ای از خروجی این مرحله:

```

client.execute(' select * from default.posts limit 3')

[(UUID('3dab8571-514f-4785-9df8-8079e4a15a2a'),
  datetime.datetime(2021, 7, 13, 23, 9, 58),
  'Aparat | اپارات',
  'Aparat_TV',
  ['سامسونگ', 'جوائز', 'پلاستیک', 'تلویزیون'],
  (UUID('489331b9-9e98-4bc1-8f5c-8ac3e86c4fe1'),
   datetime.datetime(2021, 7, 13, 23, 10, 2),
   'کانال آخرین خبر',
   'akharinkhabar',
   ['دعوای', 'نشایی', 'استرالیا', 'کانگروها'],
   (UUID('ddae39-e5b1-4345-8dff-e2f772d7db52'),
    datetime.datetime(2021, 7, 13, 23, 10, 7),
    'پایگاه خبری انتخاب',
    'entekehbar_ir',
    ['کرزی', 'مذکره', 'جزئیات', 'سیاستداران'])
  )
]
```

کد مربوط به این مرحله در آدرس [github](#) پروژه در فolder step5-clickhouse وجود دارد.

## Superset

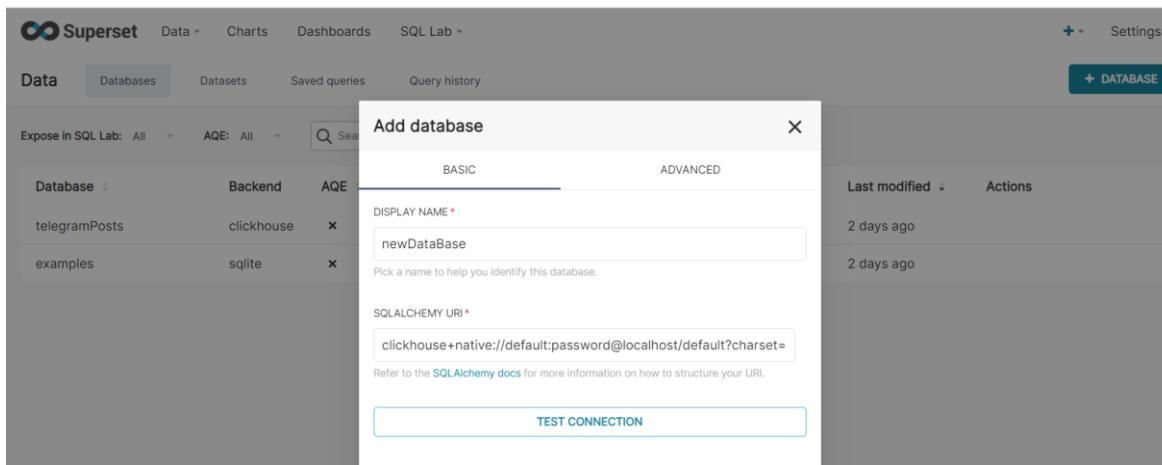
پس از نصب superset روی سیستم کافی است تا با دستور زیر آن را روی port مورد نظرمان run کنیم.

```
(superset-env) bahar1375@DESKTOP-2QPI504:~$ superset run -p 8000 -h 0.0.0.0
logging was configured successfully
```

برای تولید داشبورد های مناسب در این مرحله لازم است تا تیبلی که در مرحله پیش در ایجاد کردیم را به superset معرفی کنیم تا بتوانیم از داده های این تیبل در Superset استفاده کنیم.

برای این کار کافی است تا driver ClickHouse را نصب و سپس با url مناسب دیتابیس مان را به superset معرفی کنیم.

برای اینکار از Menu بالا گزینه Data و سپس database را انتخاب میکنیم



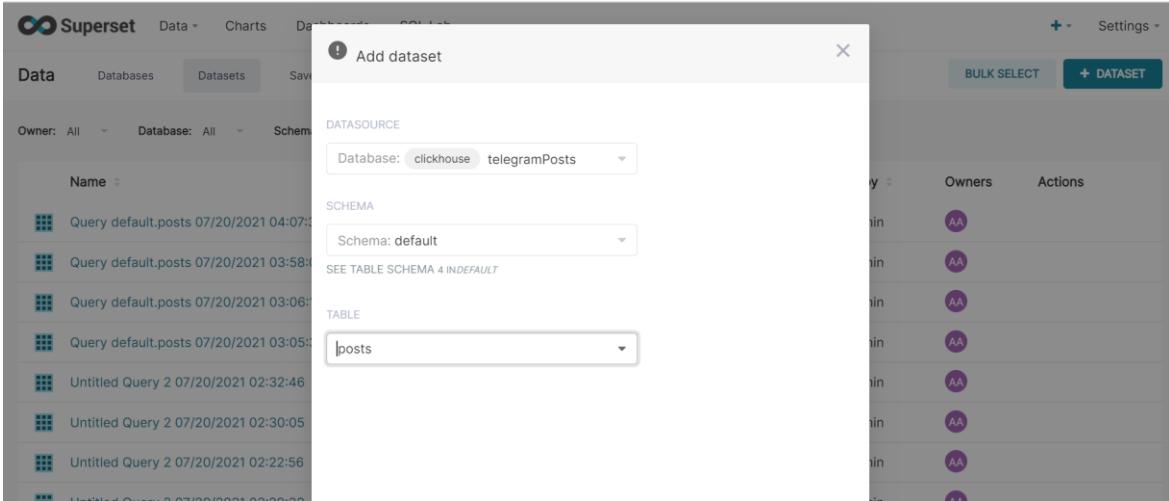
آنگاه با زدن دکمه + میتوان url مناسب را وارد کرد. این url باید شامل داده هایی مانند driver دیتابیس مورد نظرمان ادرس host , port ای که دیتابیس روی آن بالاست و نیز charset و password باشد.

مانند url زیر :

clickhouse+native://default:password@localhost/default?charset=utf8

بعد از اتصال Superset بے database باید تیبل مورد نظرمان را انتخاب کنیم :

برای این کار باید ابتدا گزینه Dataset را از Menu بالا انتخاب و سپس گزینه +DATASET را انتخاب کنیم در پنجره باز شده میتوانیم ، table و database , schema را انتخاب کنیم .



آنگاه داده ما آمادست و میتوانیم با اعمال query های مناسب و ساخت chart های مورد نظر، dashboard های مورد نیازمان را بسازیم.

### گزارشها مرتبط با هشتگها/کلمات کلیدی

برای اینکه ابتدا تمام hashtag های موجود در لیست hashtags را استخراج و سپس گروه بندی مناسب انجام میدهیم :

The screenshot shows the Superset interface with the SQL Lab tab selected. A query is being run against the 'clickhouse' database, schema 'default', and table 'posts'. The query is:

```

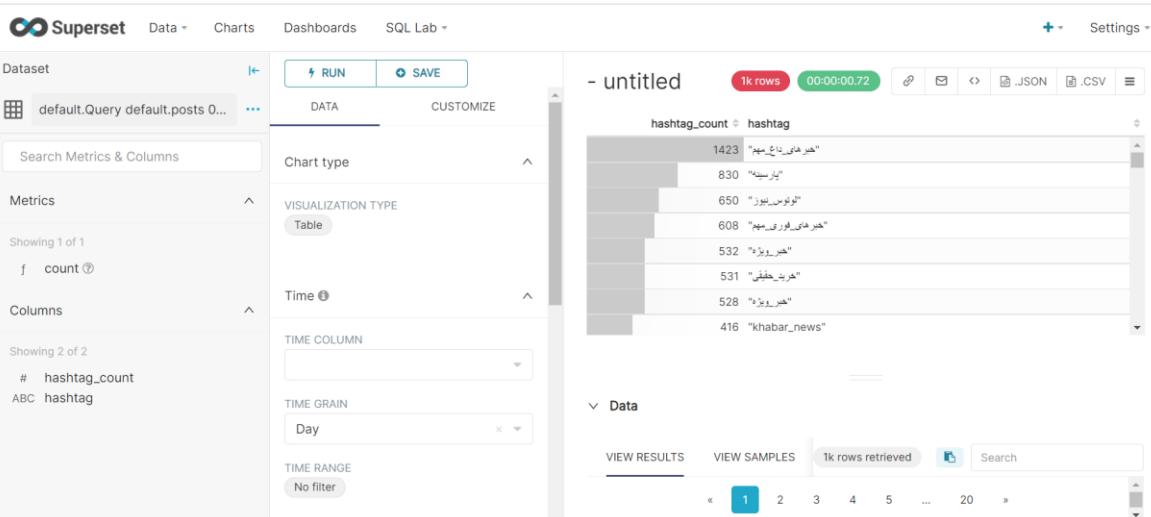
1 SELECT count(hashtag) as hashtag_count ,arrayJoin(hashtags) AS hashtag
2 FROM "default".posts
3 GROUP by hashtag order by hashtag_count desc

```

The results show 1000 rows returned, with one row displayed:

hashtag_count	hashtag
1421	"خبر_های_نار_میوه"

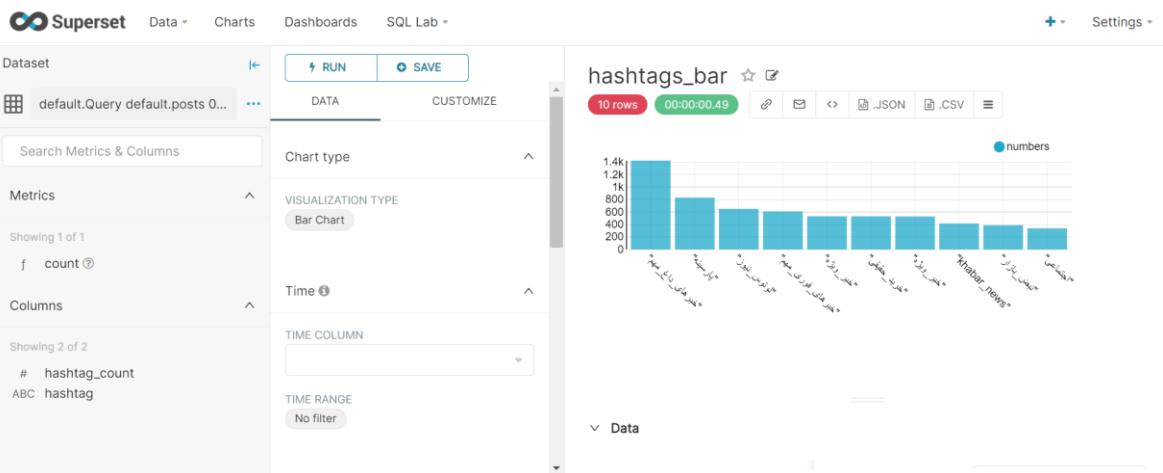
برای نمایش مناسب خروجی کافیست گزینه EXPLORE را انتخاب کنیم :



پس از انتخاب این گزینه اطلاعات فوق به ما نمایش داده میشود این ساده ترین نوع visualization است که خروجی query را به شکل table نمایش میدهد

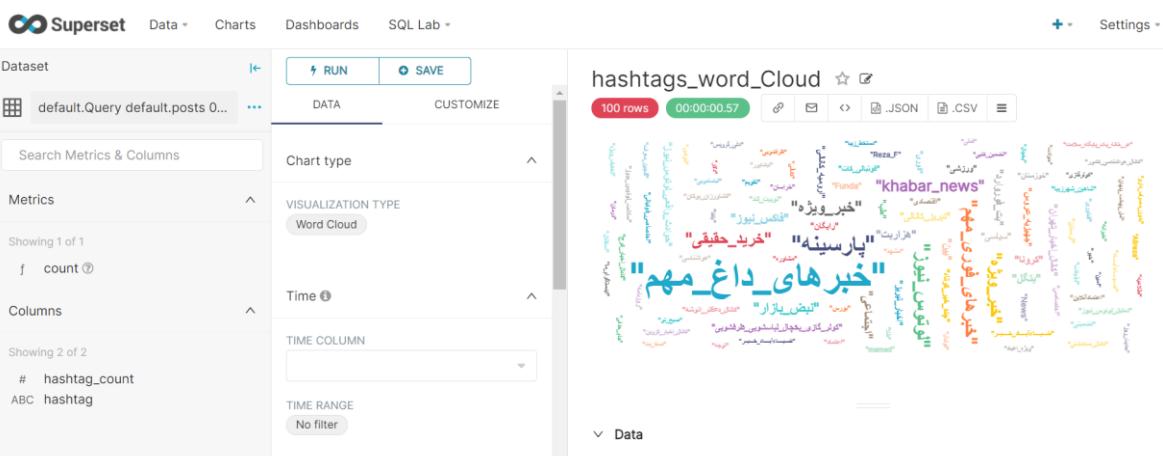
برای نمایش مناسب تر خروجی نوع نمایش را باید تغییر دهیم:

برای مثال با انتخاب barplot به عنوان نوع نمایش داده ها و نیز تنظیم پارامتر hashtag به عنوان سری نمودار زیر را خواهیم داشت :



در این نمودار hashtag های پر تکرار به ترتیب تکرار آورده شده اند.

همچنین میتوانیم این hashtag ها را به شکل wordCloud نمایش دهیم که خروجی آن به شکل زیر است

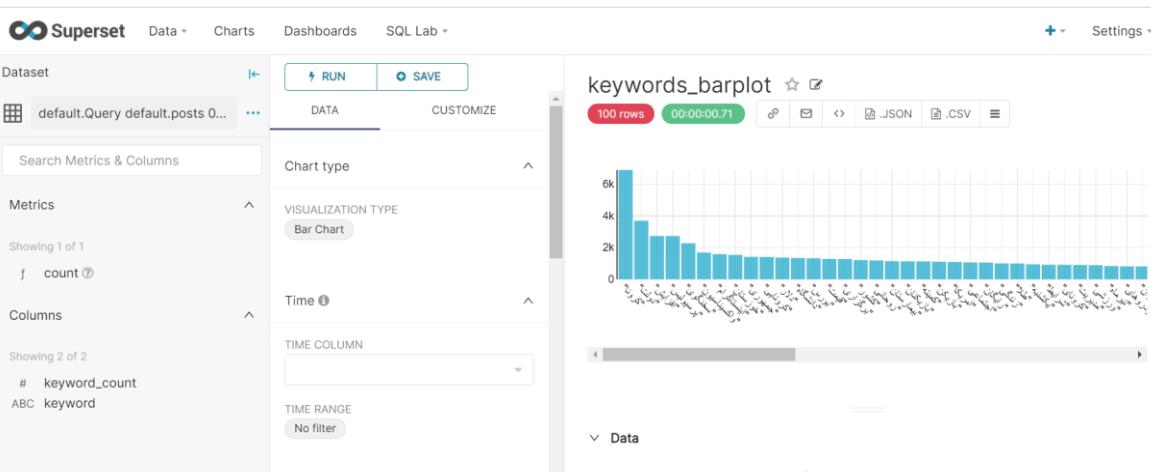
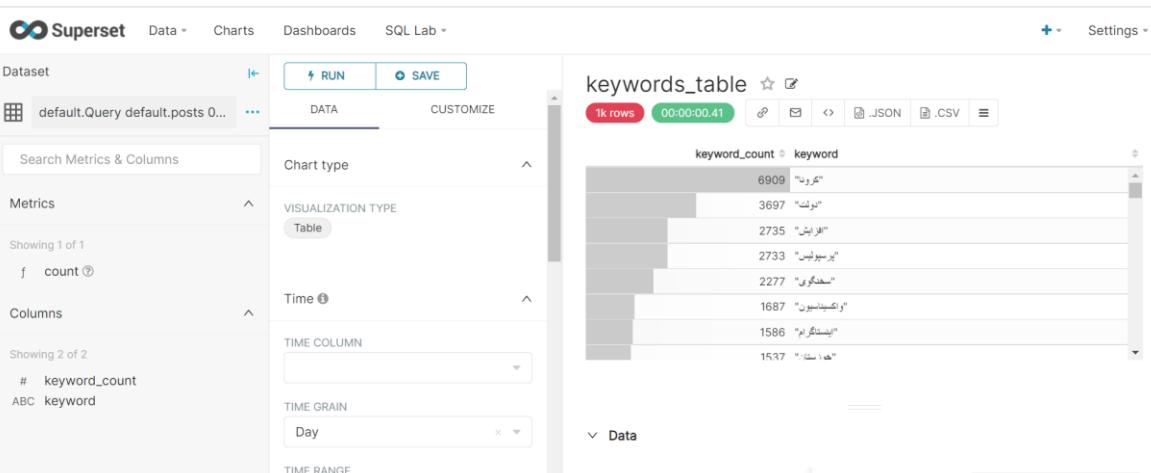


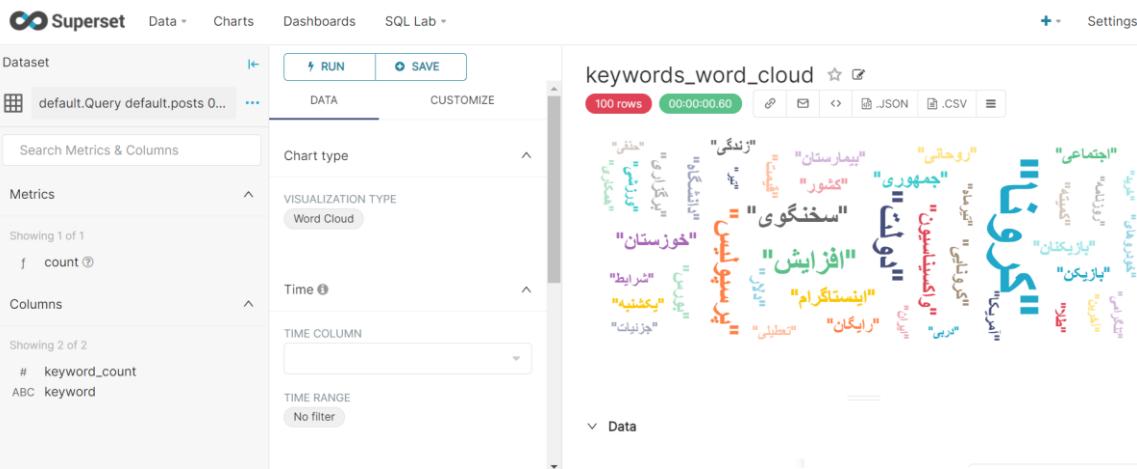
در مورد keyword ها نیز میتوانیم دقیقاً به همین شکل عمل کنیم با این تفاوت که کوئری اعمالی به شکل زیر خواهد بود:

و خروجی های آن به ترتیب به شکل زیر خواهد بود :

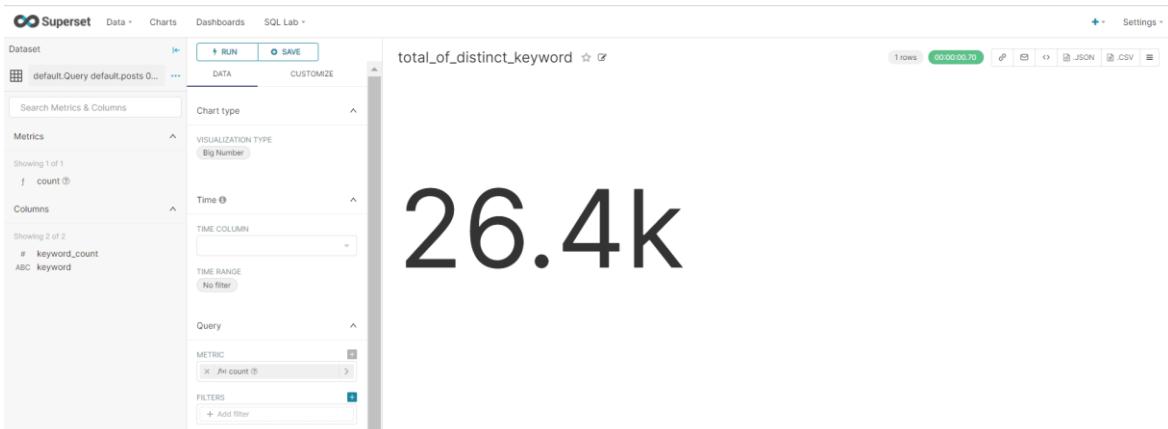
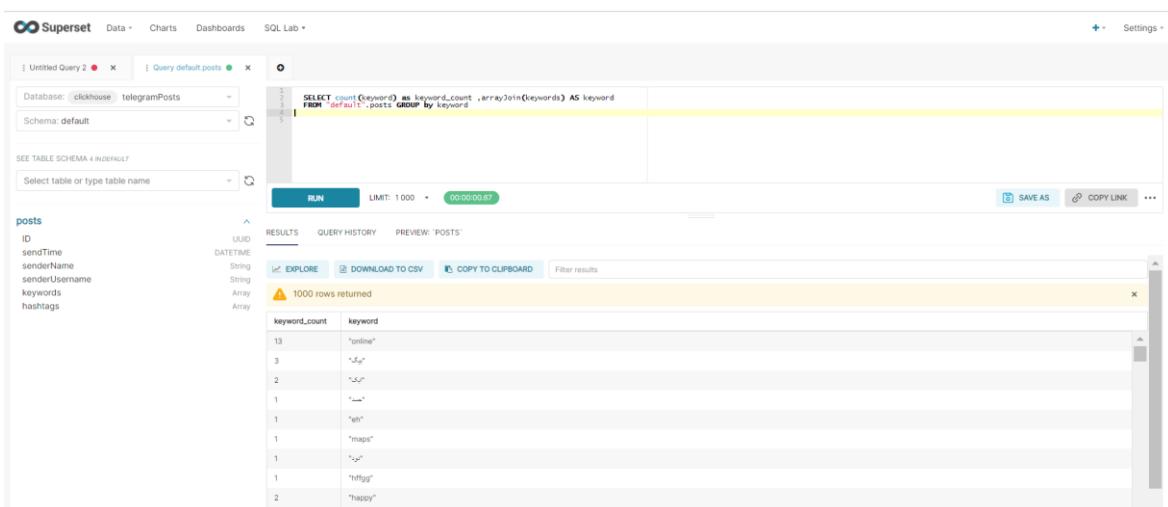
```

SELECT count(keyword) as keyword_count ,arrayJoin(keywords) AS keyword
FROM `default`.posts
GROUP by keyword order by keyword_count desc
    
```

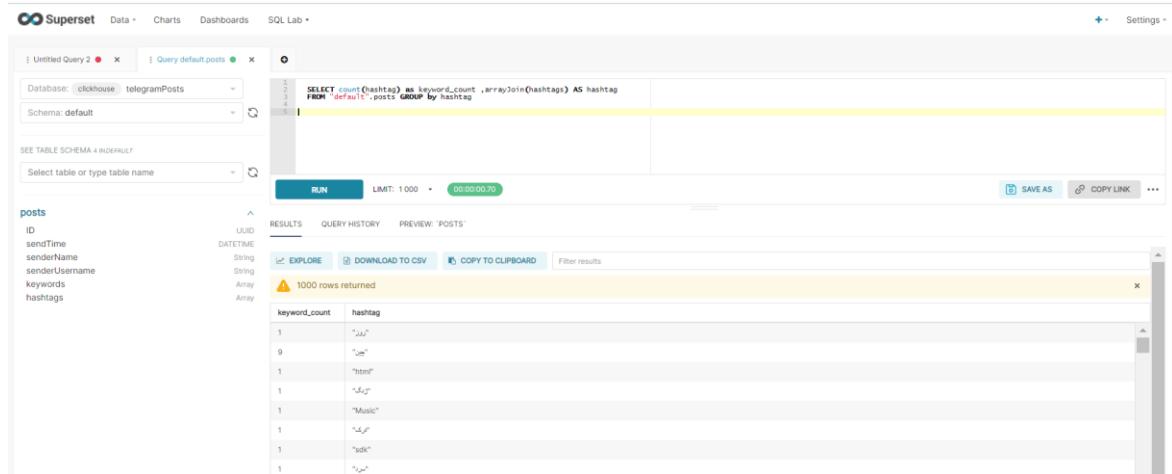




همچنین میتوان تعداد کل keyword های یکتا را به شکل زیر گرفت و با chart big Number نمایش داد :



این مقدار برای hashtags ها به شکل زیر استخراج میشود:



The screenshot shows the Superset SQL Lab interface. A query is being run against the 'clickhouse' database and the 'telegramPosts' table. The schema is set to 'default'. The query is:

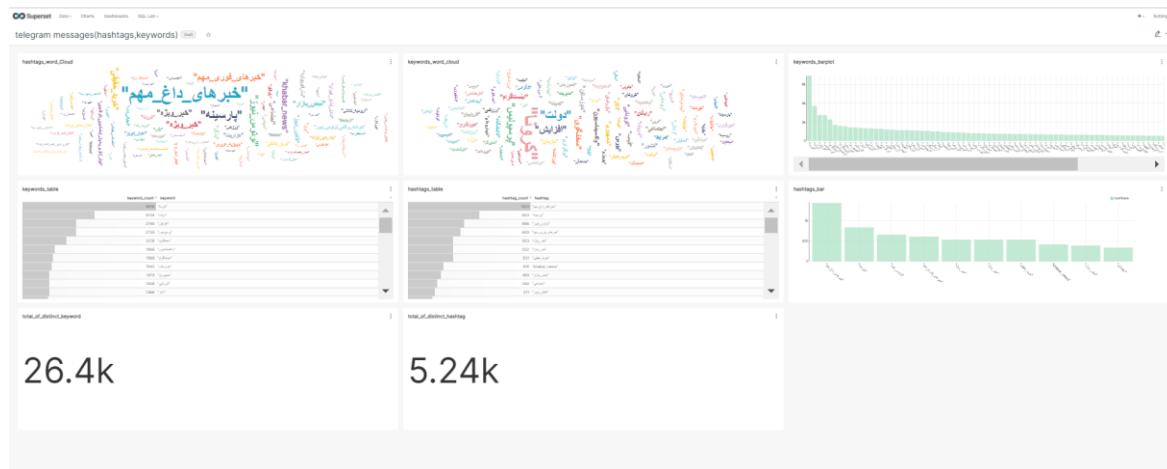
```
SELECT count(hashtags) as keyword_count ,arrayJoin(hashtags) AS hashtag  
FROM `default`.`posts` GROUP by hashtag
```

The results show 1000 rows returned, with columns 'keyword\_count' and 'hashtag'. The data includes:

keyword_count	hashtag
1	"چن"
9	"عنه"
1	"html"
1	"چیز"
1	"Music"
1	"موزیک"
1	"sdlc"
1	"من"

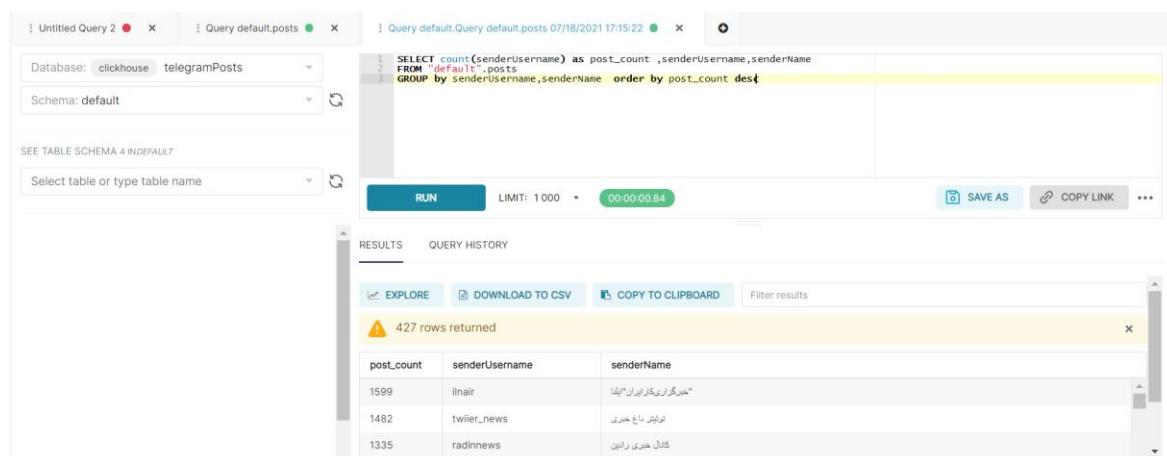


با ذخیره نمودار های فوق در داشبورد telegram message ، داشبوردی به شکل زیر خواهیم داشت :



## گزارشها مربوط به کانالها/کاربران (در صورت استفاده از تؤییتها)

برای دریافت کanal هایی با بیشترین پست کوئری زیر را اجرا میکنیم :



و برای نمایش آن از دو چارت table , barplot استفاده میکنیم که خروجی آنها به شکل زیر است :

Superset Data Charts Dashboards SQL Lab

Dataset default.Query d

Metrics f count

Columns post\_count, senderUsername, senderName

Chart type Table

TIME COLUMN Day

TIME GRAIN Day

TIME RANGE No filter

Query

channels\_table 427 rows 00:00:00.63

post_count	senderUsername	senderName
1599	linair	خبرگزاری ایران
1482	twiler_news	کوئلر نای خبری
1335	radinews	کال خبری رادین
1334	bourse24ir	بورس ۲۴
1288	khabarfouri	خبر فوری
1205	negaamnews	سایت خبری نگاه
1173	MyAsriran	مساری ان
1166	esteghlal_havadaraOn	کمال هرداران اسناد
1129	khabarai_rasm	خبر فوری و رسمی
1116	Khabare_vige	کلکس خبر و زندگانی
1085	lotosnews	لتوس
1071	khabar_Fori	خبر فوری

Superset Data Charts Dashboards SQL Lab

Dataset default.Query default.posts 0...

Metrics f count

Columns post\_count, senderUsername, senderName

Chart type Bar Chart

TIME COLUMN Day

TIME GRAIN Day

TIME RANGE No filter

METRICS numbers

channels\_barplot 10 rows 00:00:00.62

News Source	Number of Posts
خبرگزاری ایران	~1500
کوئلر نای خبری	~1400
کال خبری رادین	~1300
بورس ۲۴	~1300
خبر فوری	~1200
سایت خبری نگاه	~1200
مساری ان	~1100
کمال هرداران اسناد	~1100
خبر فوری و رسمی	~1000
کلکس خبر و زندگانی	~1000
لتوس	~1000
خبر فوری	~1000

در نمودار بعدی میخواهیم مشخص کنیم در فعل ترین کانال ها راجع به چه مباحثی بیشتر صحبت شده است

برای مشخص شدن این مساله کوئری زیر را اجرا میکنیم :

Superset Data Charts Dashboards SQL Lab Settings

Untitled Query 2 Database: clickhouse Schema: default

Query default.Query default.posts 07/18/2021 17:15:22 - Query default.Query default.posts 07/18/2021 18:56:30

```

1 select * from (
2     SELECT count(keyword) as keyword_count ,arrayJoin(keywords) AS keyword ,senderUsername
3         From
4             (
5                 select * from "default".posts where senderUsername in
6                     (
7                         select senderUsername
8                             from (
9                                 SELECT count(senderUsername) as post_count ,senderUsername,sende
10                                FROM "default".posts
11                                GROUP by senderUsername,sende
12                                ORDER BY post_count DESC LIMIT 10
13                            )
14                        )
15                    )
16                )as b
17            JOIN
18        (
19            select DISTINCT (senderUsername),sende
20                FROM "default".posts
21            ) as c
22            USING senderUsername
23
24

```

RUN LIMIT: 1000 00:00:01.35

RESULTS QUERY HISTORY

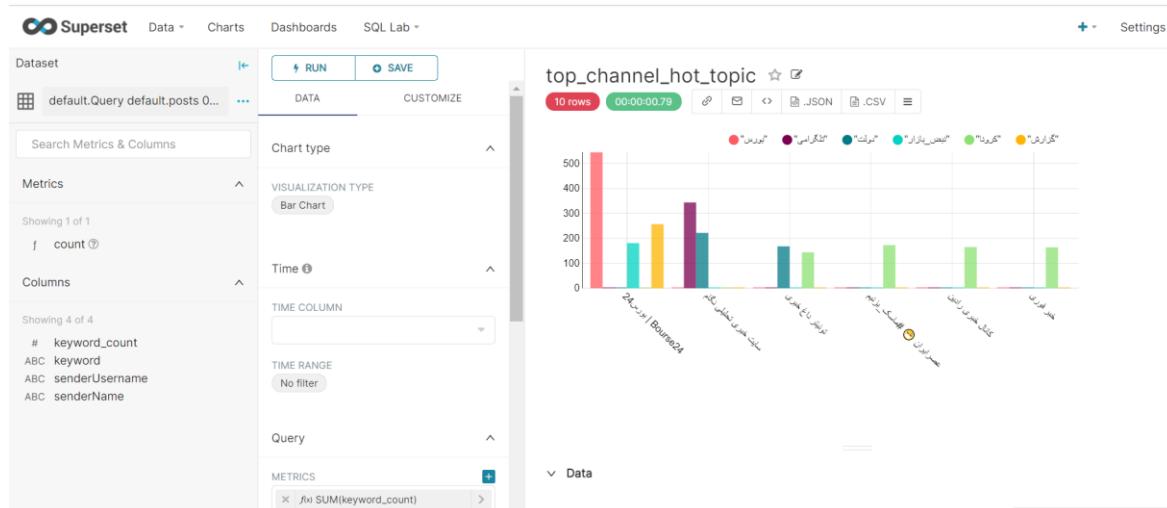
EXPLORE DOWNLOAD TO CSV COPY TO CLIPBOARD Filter results

1000 rows returned

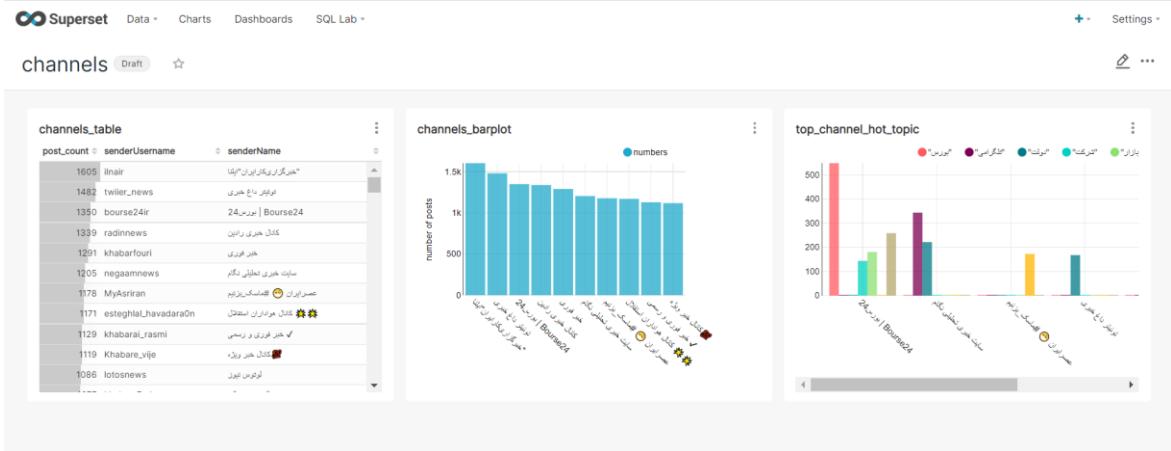
keyword_count	keyword	senderUsername	sende
500	تهران	user1	تهران
250	مشهد	user2	مشهد
300	کرج	user3	کرج
150	آذربایجان	user4	آذربایجان
150	گرگان	user5	گرگان
150	یزد	user6	یزد
150	اصفهان	user7	اصفهان
150	شیراز	user8	شیراز
150	همدان	user9	همدان
150	قزوین	user10	قزوین

SAVE AS COPY LINK ...

و نتیجه آن به شکل زیر خواهد بود:



و در انتهای این بخش داشبوردی به شکل زیر خواهیم داشت :



## گزارش‌های عمومی سامانه مانند آمارکلی دریافت اطلاعات در یک روز و یک ساعت گذشته

با استفاده از query زیر تعداد کل پست‌ها تا این لحظه را خواهیم داشت :

```
1 select count(ID) from default.posts
```

RUN LIMIT: 1 000 00:00:01.00

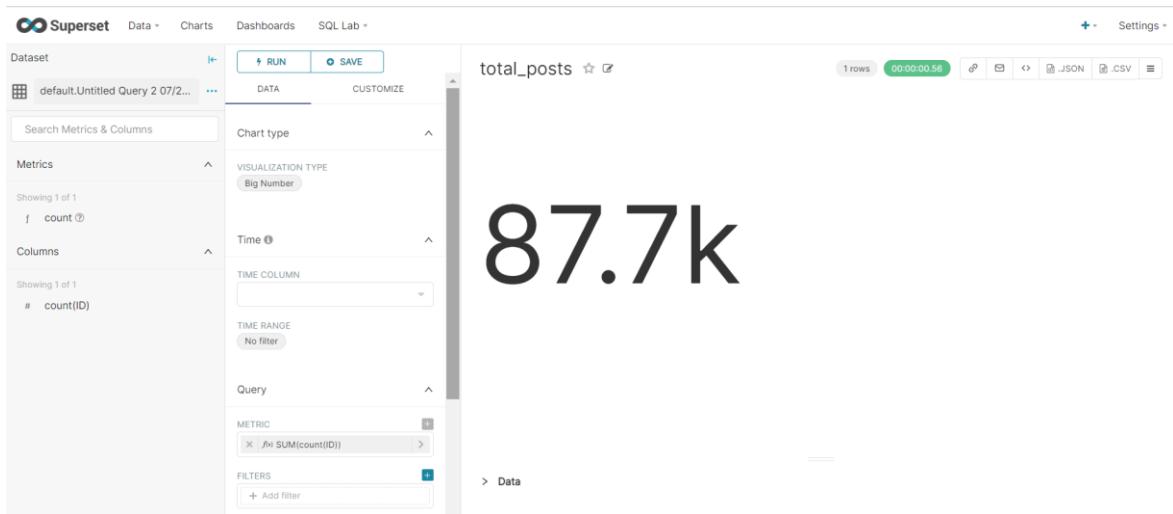
RESULTS QUERY HISTORY

EXPLORE DOWNLOAD TO CSV COPY TO CLIPBOARD Filter results

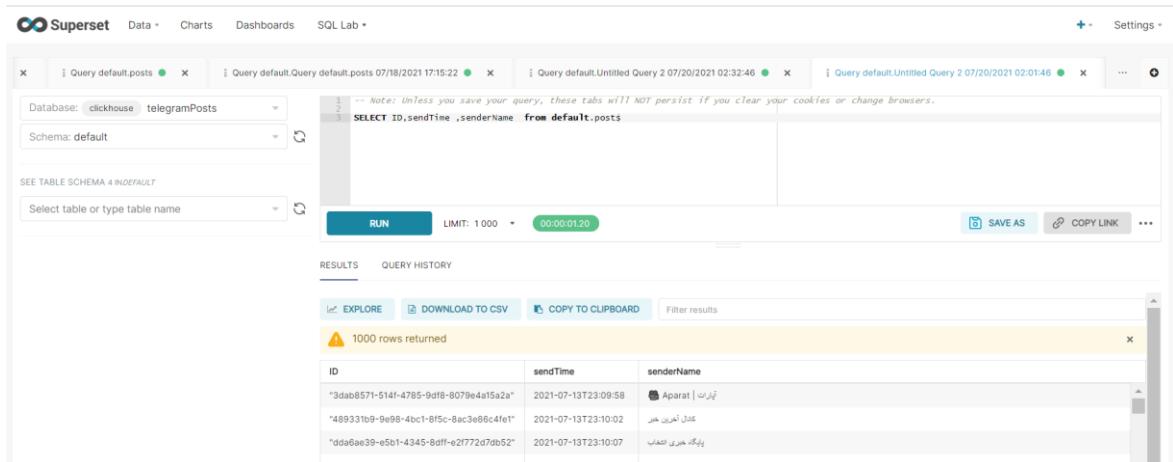
⚠ 1 rows returned

count(ID)
87672

که آن را به کمک big Number به شکل زیر نمایش میدهیم :



برای بدست آوردن تعداد پست ها در 1 ، 3 و ... روز ، ماه و ... گذشته میتوانیم از کوئری زیر استفاده کنیم و سپس روی خروجی کوئری بنابر نیازمندی فیلتر زمانی اعمال کنیم .

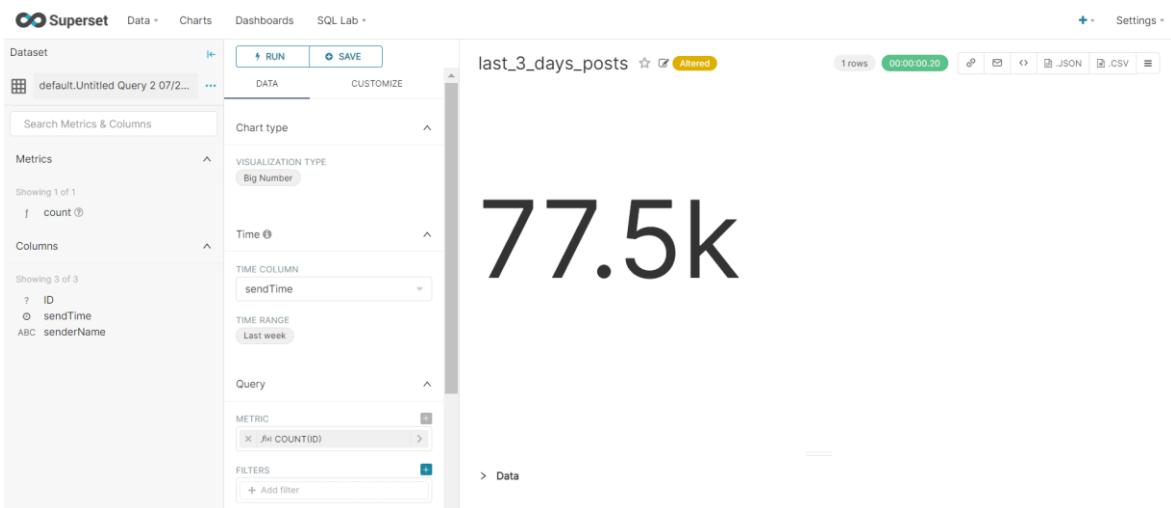


سپس میتوان با انتخاب chart bigNumber و فیلتر زمانی یک روز گذشته تعداد کل پست های یک روز گذشته را بدست آورد :

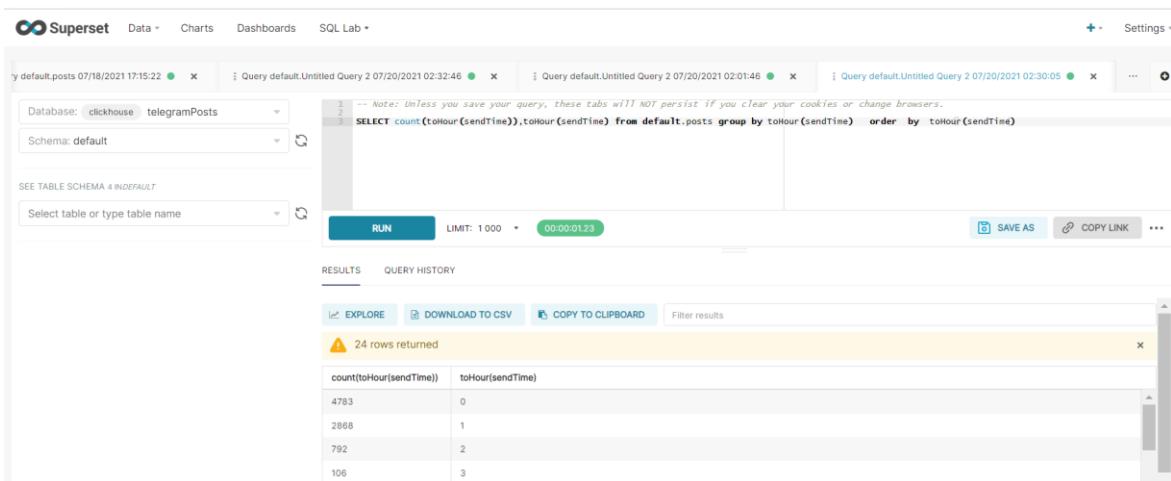
The screenshot shows the Superset interface with a query titled "last\_day\_posts". The configuration dialog is open, specifically for the "Time" section. Under "RANGE TYPE", "Last" is selected. Under "TIME RANGE", "Last day" is selected. The "Actual time range" shows the period from 2021-07-19 to 2021-07-20. At the bottom right of the dialog are "CANCEL" and "APPLY" buttons.

The screenshot shows the Superset interface with the same "last\_day\_posts" query. The result is displayed as a large, bold number "12.8k" in the center of the screen. The left sidebar shows the query configuration, and the top navigation bar includes "Superset", "Data", "Charts", "Dashboards", "SQL Lab", and "Settings".

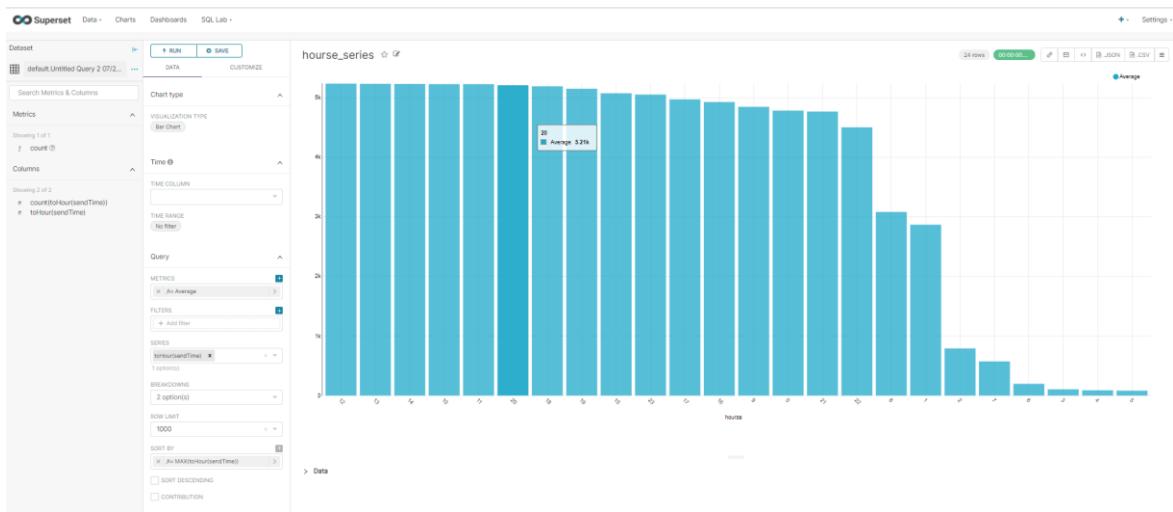
تعداد کل post ها در یک هفته گذشته نیز به شکل زیر است :



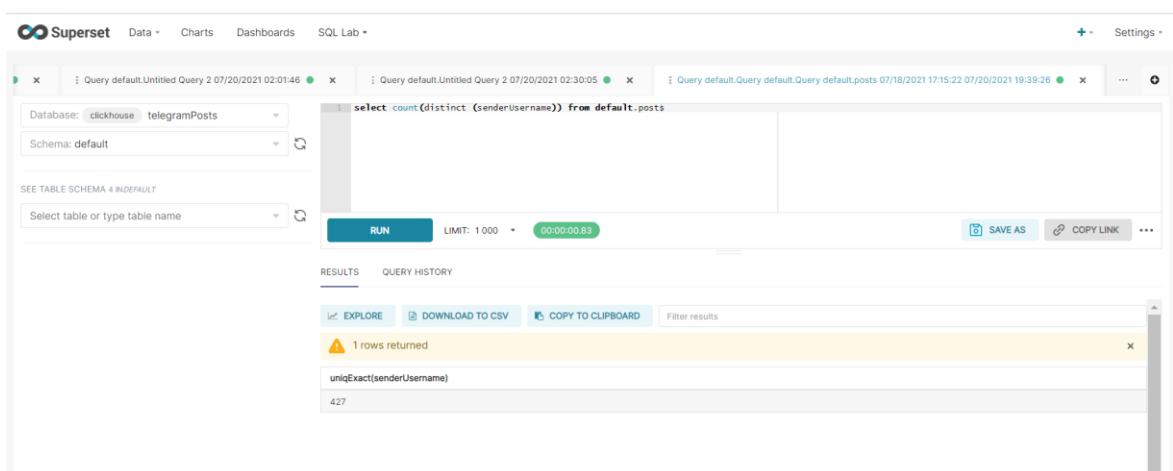
برای اینکه بدانیم به طور میانگین در هر ساعت از شبانه روز چه میزان post دریافت میشود میتوان از دستور استفاده کرد:



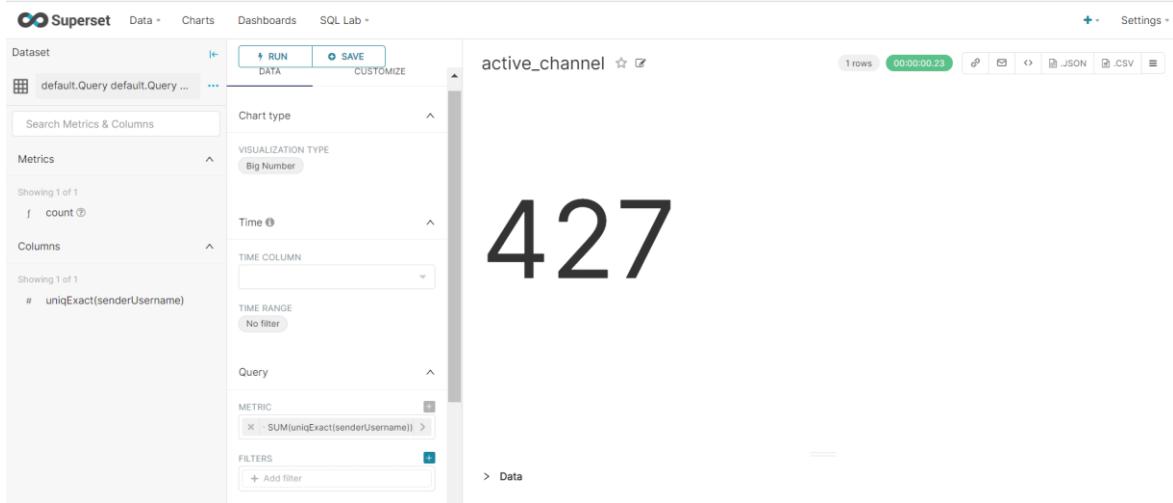
و نمایش آن به شکل زیر خواهد بود



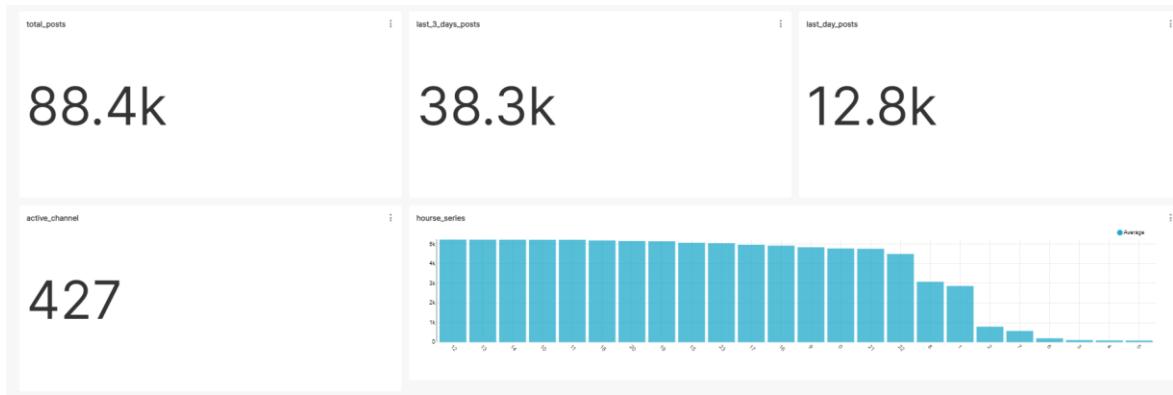
و تعداد کانال هایی که تا کنون حداقل 1 پیام ارسال کرده باشند به شرح زیر است :



که آن را در قالب big Number نمایش میدهیم .



با اضافه کردن چارت های فوق به یک داشبورد خواهیم داشت :



### گزارش‌های مرتبط با یک کانال خاص / یک هشتگ خاص

در اینجا کافی کوئری هایمان را کلی در نظر بگیریم و سپس بر اساس یک فیلد خاص خروجی را فیلتر کنیم :

برای مثال میخواهیم در مورد یک کانال خاص تعداد پست هایش در هر روز ببینیم :

برای این کار کوئری به شکل زیر اعمال میکنیم :

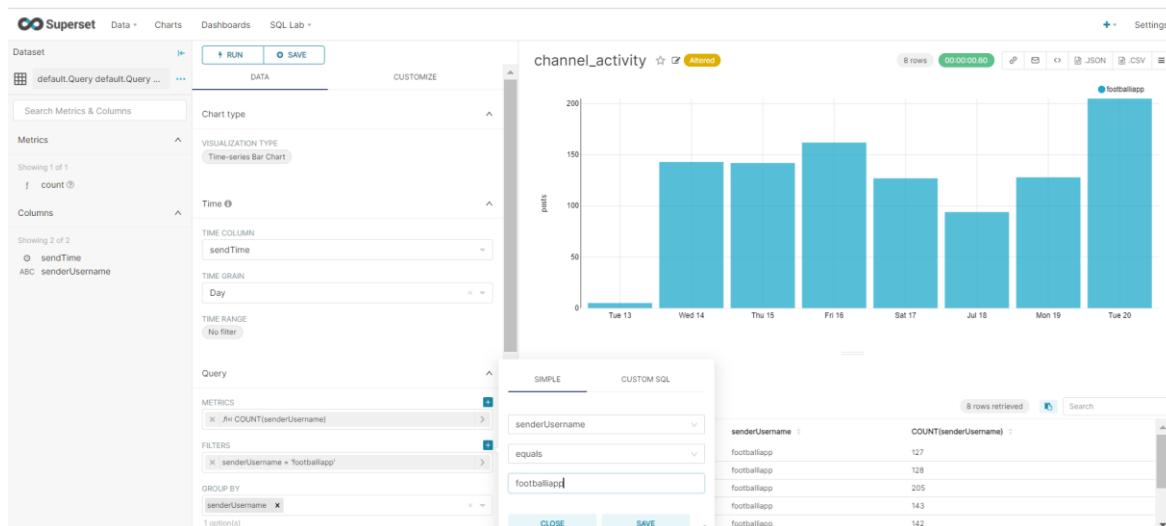
The screenshot shows the Superset interface with the SQL Lab tab selected. A query is running against the 'telegramPosts' table in the 'default' schema. The query is:

```
SELECT sendTime, senderUsername FROM telegramPosts
```

The results table has two columns: 'sendTime' and 'senderUsername'. It displays 1000 rows of data, with the first few rows being:

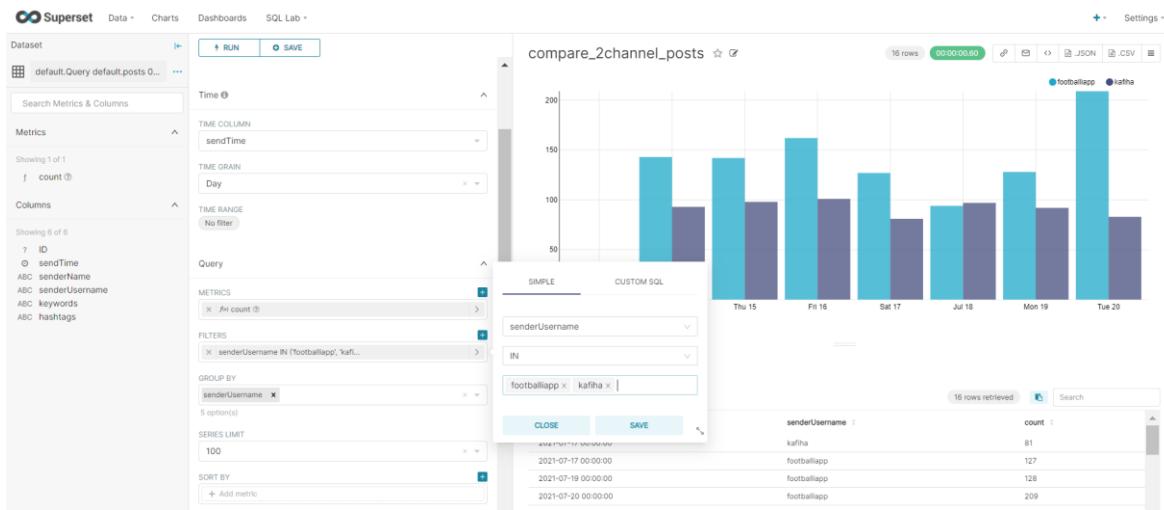
sendTime	senderUsername
2021-07-13T23:09:58	Ayman_Tv
2021-07-13T23:10:02	akhermehata
2021-07-13T23:10:07	emreheba_12
2021-07-13T23:10:12	khatami_yaser
2021-07-13T23:10:16	Pengobatanmed2
2021-07-13T23:10:20	khatamm
2021-07-13T23:10:25	khatamm
2021-07-13T23:10:29	MajlisPezeshki
2021-07-13T23:10:34	asmatof

حال میخواهیم این اطلاعات را فقط برای کanalی با یورزنيم footballiapp نمایش دهیم برای اين کار فیلتر equal را روی اين column اعمال میکنیم .



برای اينکه فعالیت دو کanal مشخص را از لحاظ تعداد پست با هم مقایسه کنیم کوئری فوق را به شکلی متفاوت باید filter کنیم :

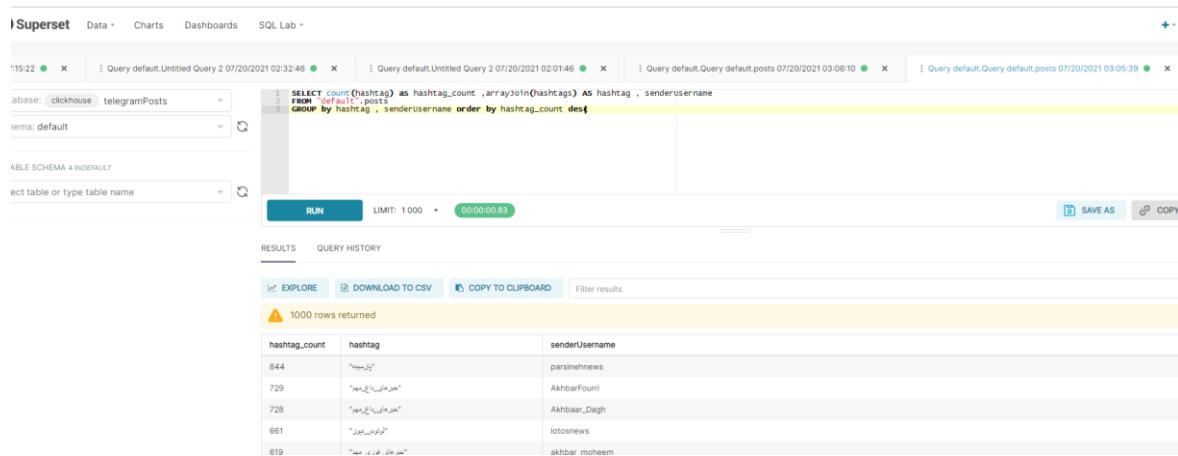
در زیر فعالیت دو کanal kafiha , footballiapp را در هر روز با هم مقایسه میکنیم :



این بار میگوییم که میخواهیم senderUsername یکی از این دو مقدار باشد .

حال در زیر میخواهیم هشتگ های یک کanal خاص را بررسی کنیم ، برای این کار کوئری زیر را اعمال

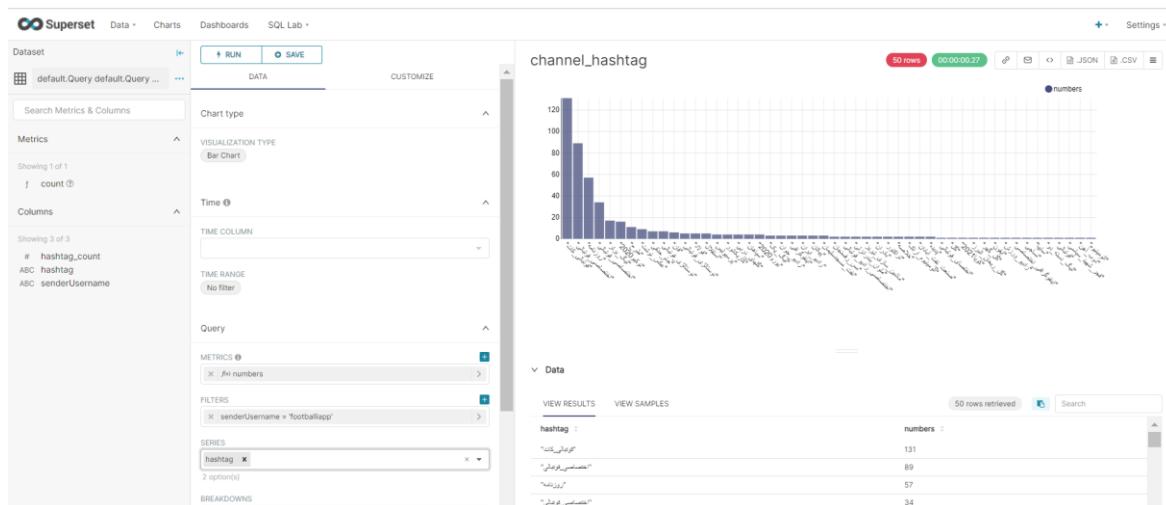
: میکنیم :



در اینجا به ازای هر کanal هشتگ هایش را داریم

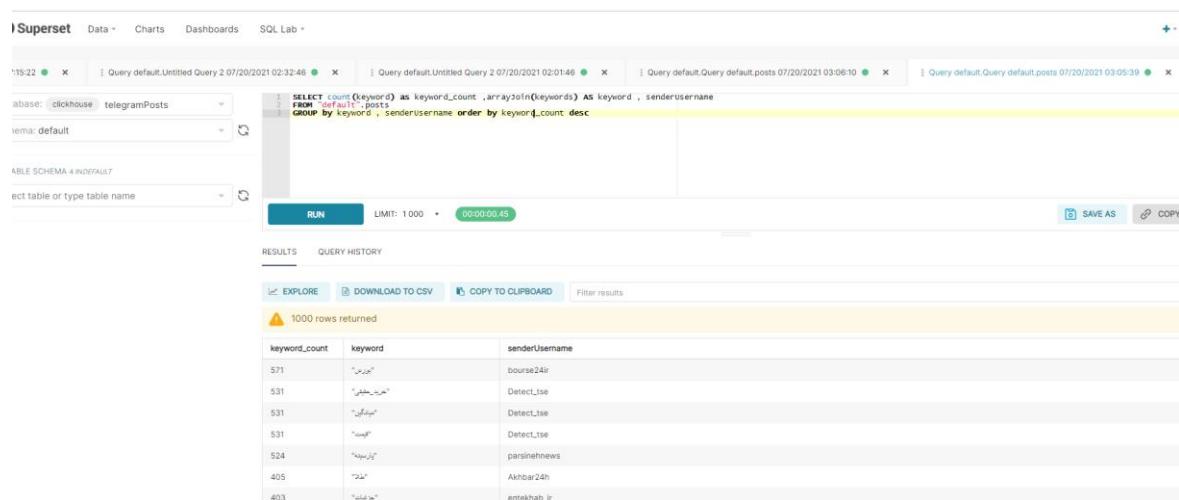
برای اینکه بتوانیم وضعیت hashtag های یک کanal خاص را استخراج کنیم به شکل زیر فیلتر اعمال

: میکنیم :

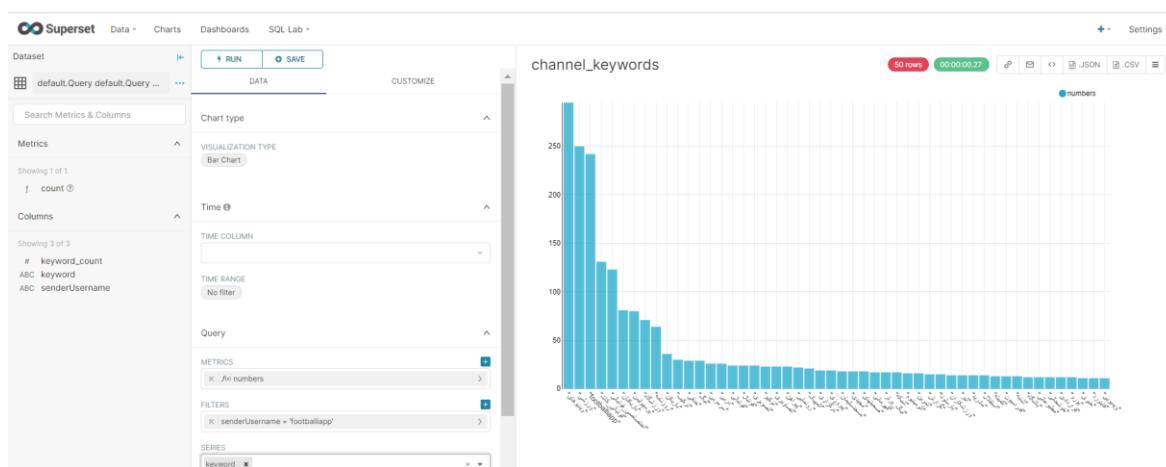
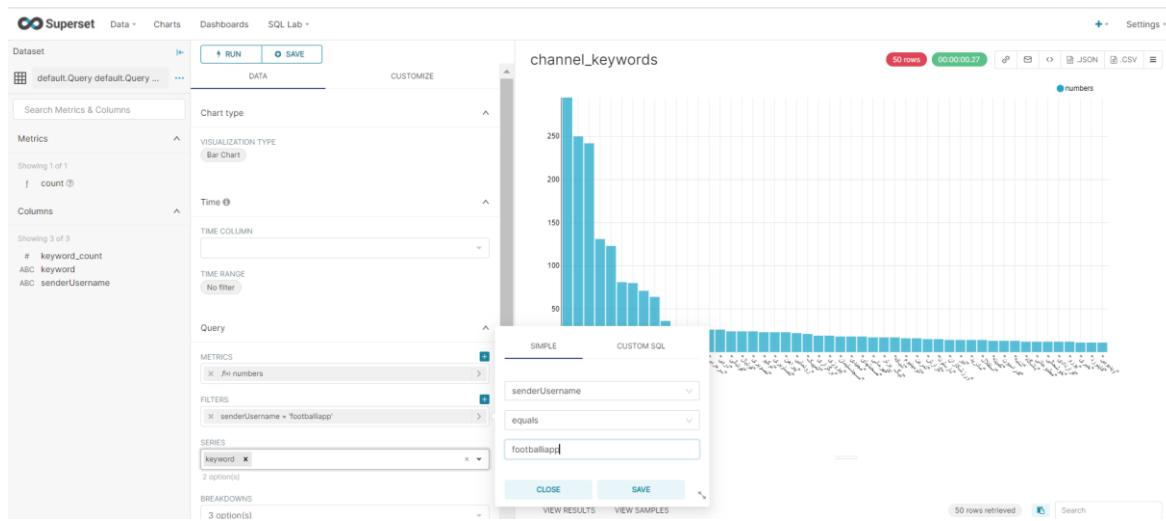


در اینجا هشتگ های کanal footballapp را استخراج کرده ایم.

برای بدست آوردن keyword های یک کanal خاص نیز مانند بالا رفتار میکنیم با این تفاوت که اینجا بر روی keyword ها تمرکز میکنیم:



برای مثال در زیر وضعیت keyword های کanal footballapp را بررسی میکنیم:



با اضافه کردن چارت های فوق به یک داشبورد خواهیم داشت :



در مورد بررسی یک hashtags خاص کافی است همانند بالا کوئری کلی بزنیم و سپس بر اساس مقادیر مورد نظر فیلتر کنیم . البته باید در نظر داشت که ما لیستی از hashtags ها به ازای هر پست داریم و لازم است که تک تک اعضا هر لیست بررسی شوند .

برای این کار کوئری زیر را اعمال میکنیم :

Superset Data Charts Dashboards SQL Lab

11:22 07/20/2021 02:32:46 x 07/20/2021 02:01:46 x 07/20/2021 03:06:10 x 07/20/2021 03:05:39 x

base: clickhouse telegramPosts schema: default

TABLE SCHEMA 4 IN DEFAULT select table or type table name

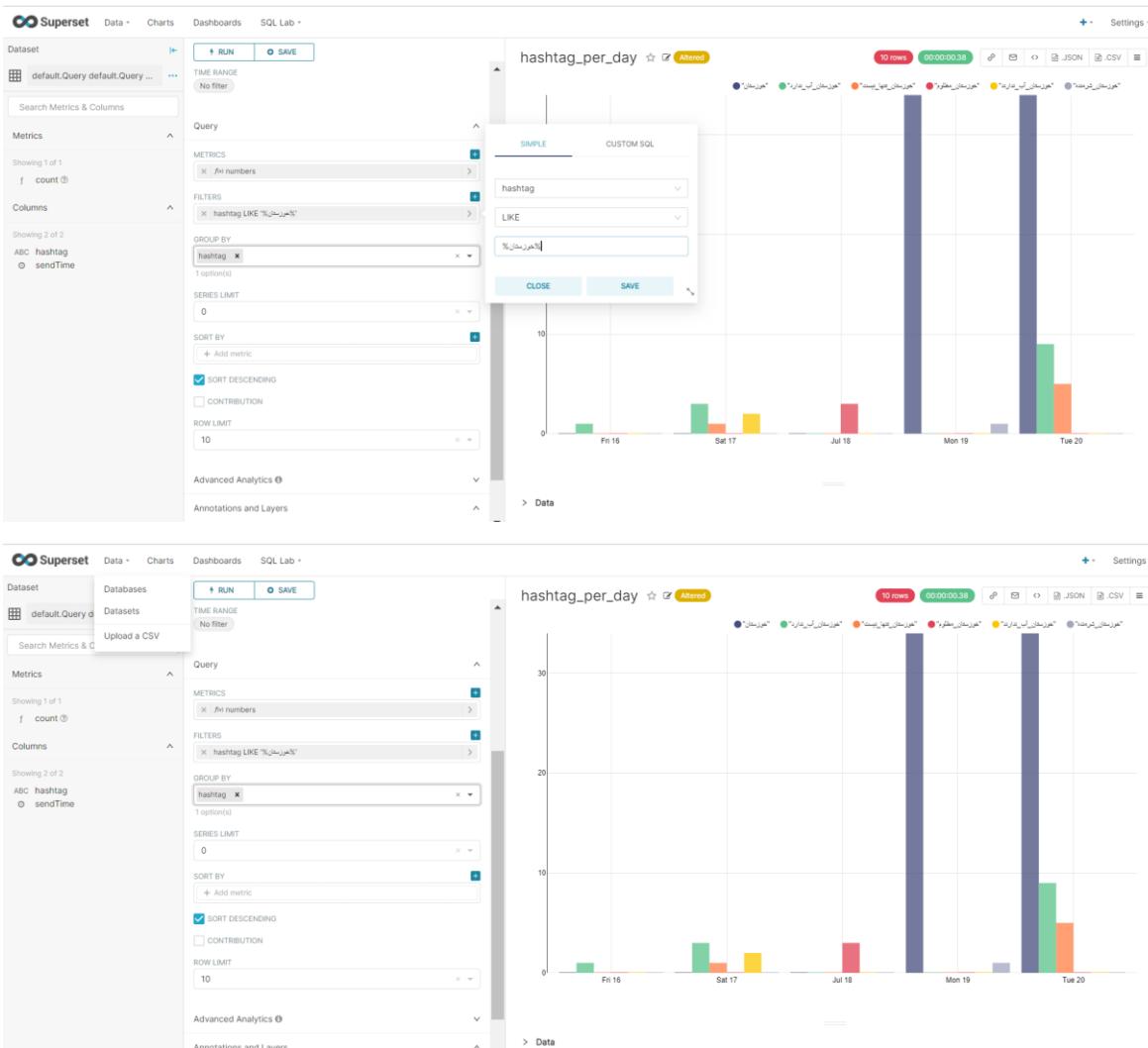
RUN LIMIT: 1000 00:00:00.62

RESULTS QUERY HISTORY PREVIEW: POSTS EXPLORE DOWNLOAD TO CSV COPY TO CLIPBOARD Filter results

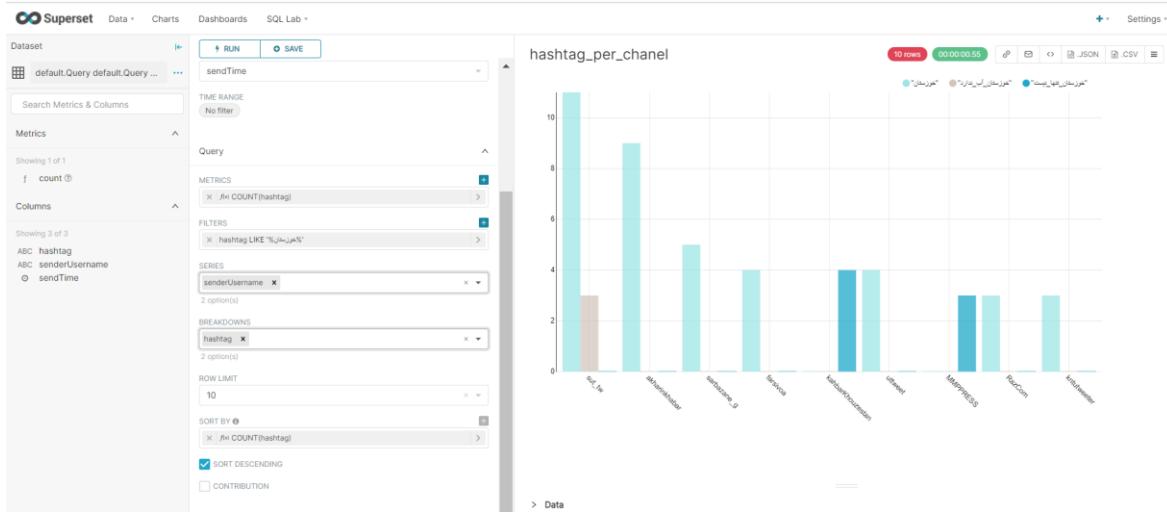
⚠ 1000 rows returned

keyword	ID	sendTime	senderName	senderUsername	keywords
"اسپورت"	"3dab8571-514f-4785-9df8-8079e4a15a2a"	2021-07-13T23:09:58	Apparat_Tv	Aparat_Tv	[{"": "06331u0627u06451u08331u0841"]
"جوان"	"3dae8571-514f-4785-9df8-8079e4a15a2a"	2021-07-13T23:09:58	Apparat_Tv	Aparat_Tv	[{"": "06331u0627u06451u08331u0841"]
"پندتک"	"3dab8571-514f-4785-9df8-8079e4a15a2a"	2021-07-13T23:09:58	Apparat_Tv	Aparat_Tv	[{"": "06331u0627u06451u08331u0841"]
"شطرنج"	"3dae8571-514f-4785-9df8-8079e4a15a2a"	2021-07-13T23:09:58	Apparat_Tv	Aparat_Tv	[{"": "06331u0627u06451u08331u0841"]
""	"489331b9-9e98-4bc1-8f5c-8ac3e86c4fe..."	2021-07-13T23:10:02	کل اخبار خبر	akharinhabar	[{"": "0627u06339u06481u0627u06cc"]
"دانلود"	"489331b9-9e98-4bc1-8f5c-8ac3e86c4fe..."	2021-07-13T23:10:02	کل اخبار خبر	akharinhabar	[{"": "0627u06339u06481u0627u06cc"]
"استرالیا"	"489331b9-9e98-4bc1-8f5c-8ac3e86c4fe..."	2021-07-13T23:10:02	کل اخبار خبر	akharinhabar	[{"": "0627u06339u06481u0627u06cc"]

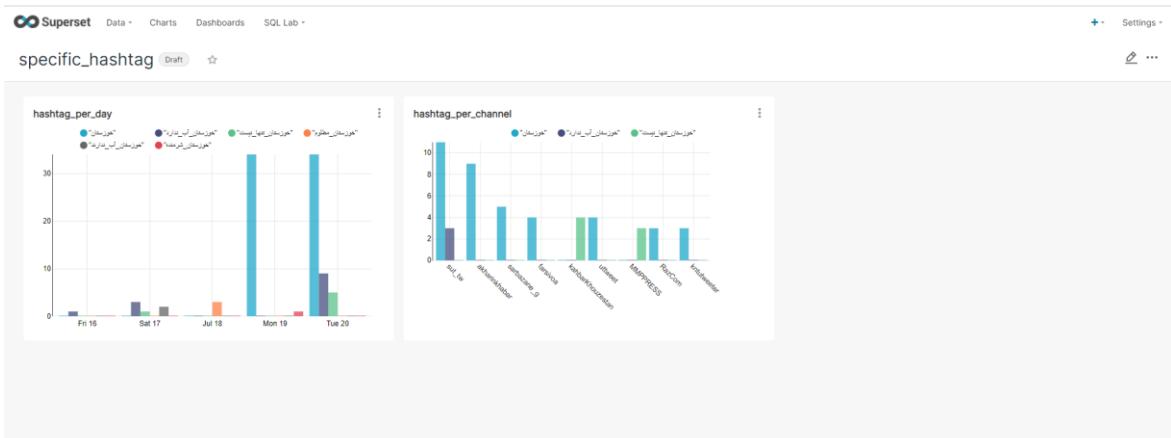
حال میخواهیم وضعیت hashtag "...خوزستان..." را در روز های مختلف بررسی کنیم. برای این کار کافی است همانند زیر عمل کنیم:



در زیر وضعیت هشتگ "...خوزستان...." را در کانال های مختلف بررسی میکنیم .



با اضافه کردن دو chart فوق به یک داشبور ، داشبوردی برای رصد hashtag ای خاص به شکل زیر خواهیم داشت :



## ساخت یک مدل پیش بینی کننده با اسپارک - بخش امتیازی

### پیش پردازش و دریافت داده ها

برای انجام این بخش باید از کاساندرا استفاده کرد که برای این منظور از داکر استفاده می کنیم. فایل آن به صورت زیر می باشد:

```

version: '2'

services:
  cassandra:
    image: bitnami/cassandra:latest
    volumes:
      - ./data/cassandra:/bitnami
    ports:
      - "9042:9042"

```

سپس دستور docker-compose up -d را اجرا می‌کنیم:



برای دسترسی به cqlsh دستور زیر را اجرا می‌کنیم:

docker exec -it spark-ml\_cassandra\_1 /bin/bash

سپس دستور زیر را وارد می‌کنیم:

cqlsh --username=cassandra --password=cassandra

در ابتدا با استفاده از cqlsh یک key\_space ایجاد می‌کنیم:

```

CREATE KEYSPACE telegram_data WITH replication = {'class':'SimpleStrategy',
'replication_factor':1};

```

ابتدا لازم است برای هر دو سوال جدول طراحی کنیم برای اینکار ابتدا تنظیمات اولیه را انجام می‌دهیم و کلاستر مورد نظر را مشخص می‌کنیم:

```

from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider
auth_provider = PlainTextAuthProvider(username='cassandra',
password='cassandra')
cluster = Cluster(auth_provider=auth_provider)
cluster = Cluster(['localhost'], auth_provider=auth_provider)
session = cluster.connect('telegram_data')

```

سپس جداول را طراحی می‌کنیم:

جدول سوال اول به شرح زیر می‌باشد:

```

create_channels = "CREATE TABLE IF NOT EXISTS channels (sender_username text, send_date text, send_hour text, message_id text, \n    PRIMARY KEY ((sender_username), send_date, send_hour, message_id))"
session.execute(create_channels)

```

این جدول دارای ستون‌های یوزرنیم کانال فرستنده، تاریخ ارسال، ساعت ارسال، آیدی پیام می‌باشد.

جدول سوال دوم به شرح زیر می‌باشد:

```

create_hashtags_Context = "CREATE TABLE IF NOT EXISTS hashtags_context2 (hashtag text, send_date text, context text, message_id text, \n    PRIMARY KEY ((hashtag), send_date, context, message_id))"
session.execute(create_hashtags_Context)

```

این جدول دارای ستون‌های هشتگ، تاریخ ارسال، متن پیام و آیدی پیام می‌باشد.

سپس هر پیام را از کافکا دریافت می‌کنیم و در جدول مربوطه قرار می‌دهیم برای اینکار هر پیام را می‌خوانیم

و بر اساس فیلدهای مورد نظر جدا می‌کنیم و در بخش مربوطه قرار می‌دهیم:

```

msg_value["timestamp"] = datetime.now()
message_id = str(msg_value['id'])
send_date = str(msg_value['timestamp'])
send_hour = str(msg_value['timestamp'].hour)
sender_username = str(msg_value['sender_username'])
context = str(msg_value['context'])
hashtags = msg_value['hashtags']
# print(send_date)

query = "INSERT INTO channels(sender_username, send_date, send_hour, message_id) VALUES (?, ?, ?, ?)"
prepared = session.prepare(query)
session.execute(prepared, (sender_username, send_date, send_hour, message_id))
if len(hashtags) != 0:
    for hashtag in hashtags:
        query = "INSERT INTO hashtags(hashtag, send_date, send_hour, message_id) VALUES (?, ?, ?, ?)"
        prepared = session.prepare(query)
        session.execute(prepared, (str(hashtag), send_date, send_hour, message_id))

```

## پیش بینی زمان ارسال پست بعدی:

برای این بخش لازم است با استفاده از اسپارک به کاساندرا متصل شویم و دادهها را بخوانیم به همین منظور نیاز داریم از jar فایلی استفاده کنیم که اتصال بین این دو را برقرار کند و این jar فایل را از اینترنت دانلود می کنیم و سپس کانفیگ های مورد نظر را انجام می دهیم:

```
conf = SparkConf() \
.setAppName("telegram-prediction") \
.setMaster("local[*]") \
.set('spark.jars', 'spark-cassandra-connector-assembly_2.12-3.0.1.jar')
```

```
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)
```

با کمک دستور زیر دادهها را از اسپارک می خوانیم:

```
df = sqlContext.read.format("org.apache.spark.sql.cassandra") \
.option("spark.cassandra.auth.username", "cassandra") \
.option("spark.cassandra.auth.password", "cassandra") \
.options(table="channels", keyspace="telegram_data").load()
```

لازم است یک سری پیش پردازش هایی صورت بگیرد به این دلیل که این timestamp داده های ذخیره شده ثانیه و میلی ثانیه را ذخیره می کند و این در حالی که است برای این سوال به چنین دقیقی نیاز نداریم پس میلی ثانیه را نادیده می گیریم و ثانیه را برابر با 0 می کنیم.

همچنین کanal مورد نظر را خبر فوری با یوزرنیم khabar\_For1 انتخاب می کنیم و در نهایت باید دادهها به فرمتی باشند که برای کتابخانه یادگیری ماشین مورد قبول باشد که باید ستون داده که زمان می باشد دارای نام ds باشد و ستون دیگری به نام y وجود داشته باشد که مشخص می کند در این زمان چه تعداد پست در کanal منتشر شده است:

```
train_data = df.where(F.col('sender_username') == 'khabar_For1')#uttweet
train_data = train_data.toPandas()
train_data['send_date'] = pd.to_datetime(train_data['send_date'])
train_data['send_date'] = [dt.strptime('%Y-%m-%d %H:%M:00') for dt in train_data['send_date']]
train_data["send_date"] = train_data["send_date"]
train_data["y"] = 1
train_data = train_data[["send_date", "y"]]
train_data.columns = ['ds', 'y']
```

سپس دیتا فریمی ایجاد می کنیم که به ازای زمان آغاز و پایان و به ازای هر دقیقه یک سطر ایجاد می کند و هر سطر آن را با داده اصلی که از کاساندرا به دست می آید مقایسه می شود و در صورتی که این زمان وجود داشته باشد تعداد پیامها در این زمان به دست می آید و باقی زمانها 0 باقی می ماند.

```
In [7]: def range_datetime(start, end, delta):
    current = start
    while current < end:
        yield current
        current += delta

In [8]: dts = [dt.strftime('%Y-%m-%d %H:%M:00') for dt in
           range_datetime(datetime(2021, 7, 13, 0, 0), datetime(2021, 7, 18, 23, 59), timedelta(minutes=1))]

In [9]: y = [0 for i in range(len(dts))]
for i in range(len(dts)):
    if dts[i] in list(train_data['ds']):
        y[i] = (sum((train_data[train_data['ds'] == dts[i]]['y']).values))

In [10]: midd_data = {'ds': dts, 'y': y}
df_train = pd.DataFrame(midd_data)
df_train
```

در نهایت از دادهها sample گرفته می شود.

```
df1 = df_train[df_train['y'] == 0].sample(len(df_train[df_train['y'] == 0]) - len(df_train[df_train['y'] == 1]))
df_train = df_train[~df_train.index.isin(df1.index)]
```

طبق مطالعات انجام شده بهترین کتابخانه یادگیری ماشینی در اسپارک برای سری زمانی fbprophet می باشد که با کمک anaconda نصب شده است:

conda install --channel conda-forge pystan fbprophet

و برای آن از دستور زیر استفاده شده است:

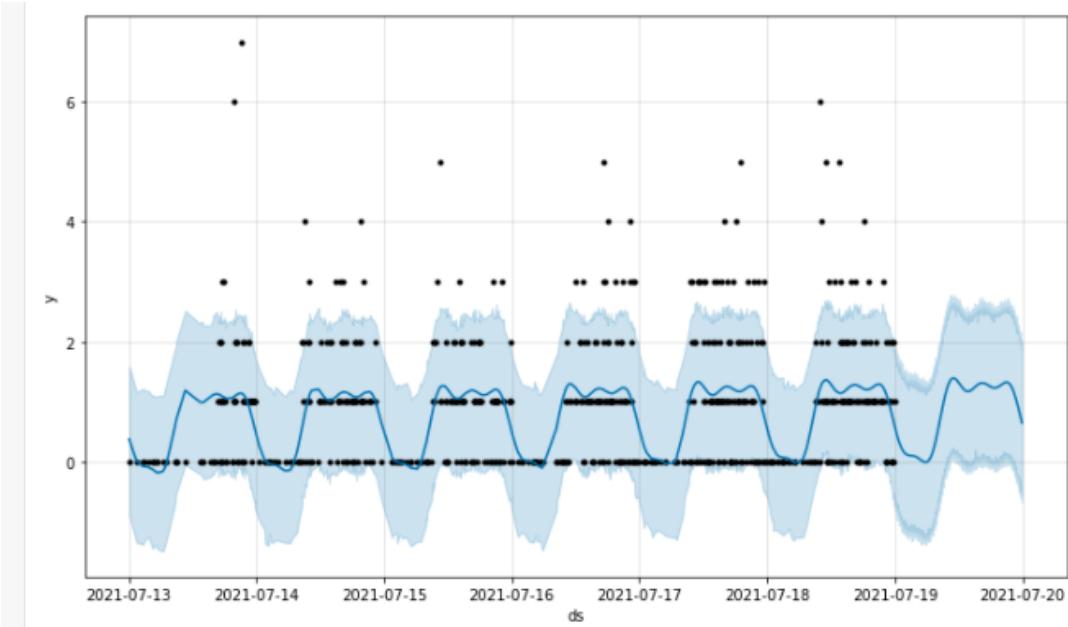
```
from fbprophet import Prophet
m = Prophet().fit(df_train)
```

سپس از مدل آموزش دیده شده برای پیش بینی آینده استفاده می شود که داده های روز 13 تا 18 ماه را وارد کردیم و می خواهیم داده های روز 19 ام را پیش بینی کنیم

برای این منظور از کد زیر استفاده می کنیم:

```
future = m.make_future_dataframe(periods=24*60, freq='min', include_history=True)
fcst = m.predict(future)
```

و سپس آن را به تصویر می کشیم:



نمودار فوق به ازای نقطه‌های مشکی رنگ عملکرد گذشته مدل را نشان می‌دهد و به ازای خط آبی رنگ روند تغییرات را نمایش می‌دهد که در انتهای نمودار که نقطه‌ای وجود ندارد عملکرد مدل نمایش داده شده است.

برای به دست آوردن اولین زمانی که این کanal پس از نیمه شب روز 18 ام پست منتشر می‌کند کد زیر را استفاده می‌کنیم:

```
: rslt_df = fcst.loc[(fcst['yhat'] > 1) & (fcst['ds']>datetime(2021,7,19,0,0,0))]

: rslt_df.iloc[0].ds
: Timestamp('2021-07-19 09:12:00')
```

همان طور که در بالا نیز مشخص است اولین زمانی که پست در این کanal منتشر خواهد شد ساعت 9:12 دقیقه می‌باشد که به این معنی است که زمان بعدی ارسال پست به دقیقه برابر با 552 دقیقه می‌باشد.

```
: ((rslt_df.iloc[0].ds-datetime(2021,7,19,0,0,0)).total_seconds())/60
: 552.0
```

### پیش بینی هشتگ‌های یک پست:

همانند بخش قبل عمل می‌کنیم و داده‌ها را از کاساندرا دریافت می‌کنیم با این تفاوت که نام جدول در اینجا برابر با hashtags\_context2 می‌باشد.

```

data = sqlContext.read.format("org.apache.spark.sql.cassandra") \
    .option("spark.cassandra.auth.username", "cassandra") \
    .option("spark.cassandra.auth.password", "cassandra") \
    .options(table="hashtags_context2", keyspace="telegram_data").load()
data=data.distinct()

```

داده‌های زیادی جمع آوری شده‌اند که دارای هشتگ می‌باشند ولی اگر از آن استفاده کنم چون مدل‌های یادگیری ماشین spark منبع محاسباتی زیادی لازم دارد برنامه crash می‌کند و متوقف می‌شود به همین دلیل با هماهنگی آقای بنایی بر روی داده‌های کمتری این سوال حل شده است.

نمای کلی از هشتگ‌ها و تعداد هر کدام به شرح زیر می‌باشد:

```

In [6]: from pyspark.sql.functions import col
data.groupBy("hashtag") \
    .count() \
    .orderBy(col("count").desc()) \
    .toPandas()

```

Out[6]:

	hashtag	count
0	خبرهای_داعمهم	13
1	خبرهای_غیرمهم	9
2	فناوری_بیور	7
3	khabar_news	6
4	کابل_اخبار_کرج	5
...	...	...
123	خوبیهای_چینی	1
124	چری	1
125	فروع_فرخزاد	1
126	پیاد_مسکن_گفتان	1
127	اهنگها	1

128 rows × 2 columns

پیش پردازش‌های زیادی برای این سوال در نظر گرفته شده است از جمله آن‌ها می‌توان به حذف emoji‌ها اشاره کرد که با بررسی‌های به دست آمده مشخص شد که حذف آن‌ها دقیق مدل را پایین می‌آورد به همین دلیل از آن‌ها استفاده نشده است و کد آن کامنت شده است و کد آن به شرح زیر می‌باشد:

```

import re
import pyspark.sql.functions as F
from pyspark.sql.types import *
def remove_emoji(string):
    emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F" # emoticons
        u"\U0001F300-\U0001F5FF" # symbols & pictographs
        u"\U0001F680-\U0001F6FF" # transport & map symbols
        u"\U0001F1E0-\U0001F1FF" # flags (iOS)
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
    "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)

udfsomefunc = F.udf(remove_emoji, StringType())
data = data.withColumn("context", udfsomefunc("context"))
data.toPandas()

```

پیش پردازش دیگری که به کار رفته است tokenize کردن جملات می باشد که برای اینکار از کتابخانه استفاده شده است و نتایج آن در ستون words ذخیره شده است:

```

import pyspark.sql.functions as F
from pyspark.sql.types import *
from hazm import *
udfsomefunc = F.udf(word_tokenize, ArrayType(StringType()))
data = data.withColumn("words", udfsomefunc("context"))
# data.toPandas()

```

پیش پردازش دیگری که استفاده شده است حذف stopword ها می باشد که از لیست واژه های کتابخانه هضم استفاده شده است

همچنین لازم است برای اعمال داده های متنی آنها را به بردار تبدیل کنیم برای این منظور از CountVectorizer کتابخانه spark.ml استفاده شده است.

همچنین هشتگ ها تبدیل به اعداد شدند تا به فرمت قابل قبول برای یادگیری ماشین تبدیل شوند.

همچنین در یکی از روش های یادگیری ماشین از TFIDF استفاده شده است که در این مرحله این تبدیل صورت گرفته است. که آن به صورت زیر می باشد:

```

from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, CountVectorizer
from pyspark.ml.classification import LogisticRegression
from hazm import *
# regular expression tokenizer
stopwords=[]
with open("stop-words.txt") as stop_word_file:
    stopwords = stop_word_file.readlines()
for i in stopwords: add_stopwords.append(i.replace("\n",""))
# stop words
stopwordsRemover = StopWordsRemover(inputCol="words", outputCol="filtered").setStopWords(add_stopwords)
# # bag of words count
countVectors = CountVectorizer(inputCol="filtered", outputCol="features", vocabSize=10000, minDF=5)

from pyspark.ml import Pipeline
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler
from pyspark.ml.feature import HashingTF, IDF
hashingTF = HashingTF(inputCol="filtered", outputCol="rawFeatures", numFeatures=10000)
idf = IDF(inputCol="rawFeatures", outputCol="features_TFIDF", minDocFreq=5) #minDocFreq: remove sparse terms
label_stringIdx = StringIndexer(inputCol = "hashtag", outputCol = "label")
pipeline = Pipeline(stages=[stopwordsRemover, hashingTF, idf, countVectors, label_stringIdx])
# Fit the pipeline to training documents.
pipelineFit = pipeline.fit(data)
dataset = pipelineFit.transform(data)
dataset.toPandas()

```

پس از اعمال همه این پیش پردازش‌ها (البته به جز حذف emoji‌ها) داده‌ها به دیتا فریم به شکل زیر خواهد بود:

	hashtag	context	words	filtered	rawFeatures	features_TFIDF	features	label
0	موزلان	چنان‌نوبی کلید و مدن جون\in... گه تکیه‌گاه‌اصنایی خ...	چنان‌نوبی کلید و مدن... چنان‌نوبی کلید و مدن... چنان‌نوبی کلید و مدن... چنان‌نوبی کلید و مدن...	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,...	(1.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,...	112.0	
1	اصل_کرونا_ویروس	اصل_کرونا_ویروس#,... اصل_کرونا_ویروس#,... اصل_کرونا_ویروس#,... اصل_کرونا_ویروس#,...	اصل_کرونا_ویروس#,... اصل_کرونا_ویروس#,... اصل_کرونا_ویروس#,... اصل_کرونا_ویروس#,...	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,...	37.0	
2	ان_را_این_امید_داده_دارد	برنامه_پردازن_کلیدیک تمصیی و فوق تخصصی خواره	برنامه_پردازن_کلیدیک... تمصیی و فوق تخصصی خواره	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,...	(3.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 1.0, 0.0,...	18.0	
3	گزارش	گزارش_تصویری_... دانه_برگزاری_د...	گزارش_تصویری_... دانه_برگزاری_د...	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,...	(6.0, 1.0, 0.0,... 1.0, 0.0, 0.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	18.0	
4	خبر	استریم_محیط_ویندوز_11... توضیح_سریع_این_Windo...	استریم_محیط_ویندوز_11... توضیح_سریع_این_Windo...	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,...	(8.0, 4.0, 1.0,... 1.0, 0.0, 0.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	70.0	
...	...	...	...	...	...	...	...	...
196	خبر_های_فری_مه...	مندجوین_از_لهم_ارزیلی... دیگران_دیگران_کرو...	مندجوین_از_لهم_ارزیلی... دیگران_دیگران_کرو...	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.2, 0.0,...	(1.0, 1.0, 1.0,... 0.0, 0.0, 0.0,... 0.0, 2.0, 0.0,...	1.0	
197	خبر_های_فری_مه...	دان_ایم_و_اعتشال_در... جوانان_های_چند_...	دان_ایم_و_اعتشال_در... جوانان_های_چند...	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	(2.0, 2.0, 0.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	1.0	
198	خبر_های_فری_مه...	ویندویی_فوق_المده... ت_کون_شید_د...	ویندویی_فوق_المده... ویندویی_فوق_المده... و_حشتک_که_ت_کون...	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	(4.0, 0.0, 0.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	1.0	
199	خبر_های_فری_مه...	علم_الهی_ایام_جمجمه_مشهد...	علم_الهی_ایام_جمجمه_مشهد...	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	(1.0, 3.0, 1.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	1.0	
200	سازمان_جهانی_پیاده_ث...	سازمان_جهانی_پیاده_ث...	سازمان_جهانی_پیاده_ث...	(0.0, 0.0, 0.0, 0.0,...	(0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	(0.0, 0.0, 2.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,... 0.0, 0.0, 0.0,...	121.0	

201 rows × 8 columns

حال داده‌ها را به دو دسته train و test تقسیم می‌کنیم:

```

# set seed for reproducibility
print((dataset.count()))
(train, test) = dataset.randomSplit([0.8, 0.2], seed = 100)
print("Training Dataset Count: " + str(train.count()))
print("Test Dataset Count: " + str(test.count()))

```

201  
Training Dataset Count: 158  
Test Dataset Count: 45

در این مرحله لازم است مدل را آموزش دهیم و پیش بینی مدل به دست آورده شود و همچنین دقت مدل به دست آورده شود برای این منظور از ۳ مدل مختلف استفاده شده است:

TF\_IDF با استفاده از Logistic regression .1

countVectorize با استفاده از ویژگی Logistic regression .2

Naïve bayse .3

TF\_IDF با استفاده از ویژگی Logistic regression -1

در این بخش مطابق کد زیر مدل را آموزش می دهیم و feature مورد نظر را به مدل می دهیم و سپس به ازای داده های تست پیش بینی می کنیم:

```
In [13]: lr = LogisticRegression(featuresCol = 'features_TFIDF', maxIter=20, regParam=0.3, elasticNetParam=0)
lrModel = lr.fit(train)
```

```
In [14]: predictions = lrModel.transform(test)
```

بخشی از خروجی به دست آمده برای این بخش به صورت زیر می باشد:

	context	hashtag	probability	label	prediction
0	الطبخ_آشنا: گام بلند عمان برای اختیار دانن... ...	خبر_های_نایخ_مه...	[0.8435687330419707, 0.0025867390916262134, 0...	0.0	0.0
1	اقوٽ ۷۷ کوک براز کرونا فقط در یک پیمانه تن... ...	خبر_های_نایخ_مه...	[0.7939037465759257, 0.002311444250366713, 0.0...	0.0	0.0
2	پیامرس اکرده (صلی الله علیہ وآله وآلہ واصحیحت روز) #...	خبر_های_نایخ_مه...	[0.4370969567325774, 0.003564240843062007, 0.0...	0.0	0.0
3	پیامرس اکرده (صلی الله علیہ وآله وآلہ واصحیحت روز) #...	حیثیت_روز...	[0.4370969567325774, 0.003564240843062007, 0.0...	22.0	0.0
4	مانجزای چال بازیگر مرد که همسرش سر به سرشن می...	خبر_های_نایخ_مه...	[0.25006200963056624, 0.01191034627024749, 0.0...	0.0	0.0
5	اینفوگرافیا خطر و قوع سیل و طوفان در کدام ای...	اینفوگرافی...	[0.014491868640361734, 0.013280069667171468, 0...	19.0	7.0
6	و...@TehranNews_ir اینجا به ساگهایی منتظر مرگ شیر مانند...	Lover_saipa	[0.013695369241624265, 0.01608518508042878, 0...	33.0	85.0
7	مشایی رفته کمک از مل و جان، دگنهن پیاپا...	فضل_نظری	[0.013678418108633128, 0.014257064735530194, 0...	92.0	11.0
8	ملیٹ_عین_بایه#...	ملیٹ_عین_بایه#...	[0.012988227437713993, 0.015090123630850192, 0...	105.0	11.0
9	اولین فیلم‌های بویل مسی با تهمی اینگریکتر...	کاریکاتور	[0.012575890046145633, 0.016603600500793913, 0...	122.0	11.0
10	امکان ... خراج بزرگ ای راه#...	امکان...	[0.01209477509958082, 0.01350221502487951, 0.0...	16.0	12.0
11	بررسی_نمایی_قطعه_در_طب_اسلامی#...	بررسی_نمایی_قطعه_در_طب_اسلامی#...	[0.011600096103237785, 0.012671272035947072, 0...	50.0	96.0
12	لرستان ...@asrar...	لرستان	[0.011380069259861384, 0.011938259552357404, 0...	26.0	26.0
13	لکتریکی#...	لکتریکی#...	[0.0112999903072581, 0.013215401975195515, 0.0...	79.0	79.0
14	مسکن ... ازام مسکن ۵۵ هیلین توکانی برای بازدیدشگان...	مسکن...	[0.010935318082006361, 0.012076872735743257, 0...	107.0	107.0
15	تست_شمسيت_شدني#...	تست_شمسيت_شدني#...	[0.010603491994056757, 0.008163797182675498, 0...	56.0	56.0
16	خلاصه_بازی اینستاگرام#...	خلاصه_بازی	[0.010547025241245884, 0.01106386654187269, 0...	74.0	74.0
17	پارسیده ... شهر و دنیا#...	پارسیده	[0.010424291739581254, 0.012132276431342398, 0...	11.0	11.0
18	حافظه_تاریخی ... پرستیز برای کسانی که میگردند#...	حافظه_تاریخی	[0.010210168091406931, 0.01488397536683387, 0...	66.0	124.0
19	لوبیا_المکت ... ساختهای مذهبی برای کار و خ...	لوبیا_المکت	[0.009974391406587962, 0.00987890682495949, 0...	99.0	99.0
20	کلیک_گردن ... سلام امیدوارم#...	کلیک_گردن	[0.009229711873635256, 0.013932595258605742, 0...	125.0	125.0
21	رد_مجد_هر ... این اهدام پانچاهی سلاح جنگی در حرب کنور...	سیاسی	[0.008992976968302637, 0.006528166724495138, 0...	10.0	10.0
22	از_رد_بیختن ... آخرین قیمت روز ارزهای نیجیتبال بر اساس پیشتر...	از_رد_بیختن	[0.008855665764858445, 0.010746972415080977, 0...	14.0	14.0
23	رد_مجد_هر ... داشتمد این رای تراست راه حل #...	رد_مجد_هر	[0.008647025685118543, 0.005693750674948045, 0...	24.0	24.0
24	رای_گران ... جایزه میلیونی داشن_آمنان و گذشتگان...	رای_گران	[0.008290598871666532, 0.012655986975403764, 0...	9.0	9.0
25	هر_های_یک_لایگان_سلمت ... ادامه روز پرگاری هشت...	هر_های_یک_لایگان_سلمت	[0.0075773341952104145, 0.010777884732435466, ...	28.0	18.0
26	فاکس_پیور ... بزرل و وانگی بی در تاکت در زیراهه بر جام گفت و گ...	فاکس_پیور	[0.007539940380984098, 0.019758980168505372, 0...	2.0	2.0
27	ارسل_پیشرات_و_پیشرهات_هونه ... کار زیاد و پسندیده بیکی از هشترین خبرین من در...	ارسل_پیشرات_و_پیشرهات_هونه	[0.007516000097692273, 0.008713481185016046, 0...	41.0	39.0
28	خبر_زند...	...	[0.00744755965992496, 0.008597209968278523, ...	23.0	23.0
29	فاکس_پیور ... صعود استقلال، فولاد و گل گهر به پیوه بیانی ح...	فاکس_پیور	[0.007289538219132843, 0.01948340619545737, 0...	2.0	2.0
30	ورزشی_موال_سرامیک#...	ورزشی_موال_سرامیک#...	[0.006481757992863962, 0.007308604709129509, 0...	29.0	30.0
31	پت_فرزوراد ... بالاترین سو ردماید#...	پت_فرزوراد	[0.005742213827439547, 0.006570602901831404, 0...	5.0	5.0

همچنین ارزیابی این مدل به ازای معیار f1 به شرح زیر می‌باشد:

```
: from pyspark.ml.evaluation import MulticlassClassificationEvaluator  
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction",labelCol="label",metricName="f1")  
evaluator.evaluate(predictions)  
0.5918699186991871
```

countVectorize با استفاده از ویژگی Logistic regression -2

برای این بخش نیز مطابق کد زیر مدل را آموزش می‌دهیم

```
: lr = LogisticRegression(featuresCol = 'features',maxIter=20, regParam=0.3, elasticNetParam=0)  
lrModel = lr.fit(train)  
  
: predictions = lrModel.transform(test)
```

ارزیابی آن به ازای معیار f1 به شرح زیر است:

```
: from pyspark.ml.evaluation import MulticlassClassificationEvaluator  
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction",labelCol="label",metricName="f1")  
evaluator.evaluate(predictions)  
0.5021367521367521
```

Naïve bayse -3

برای این بخش نیز مطابق کد زیر مدل را آموزش می‌دهیم

```
: from pyspark.ml.classification import NaiveBayes  
nb = NaiveBayes(smoothing=1)  
model = nb.fit(train)  
predictions = model.transform(test)
```

ارزیابی آن به ازای معیار f1 به شرح زیر می‌باشد:

```
: from pyspark.ml.evaluation import MulticlassClassificationEvaluator  
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction",labelCol="label",metricName="f1")  
evaluator.evaluate(predictions)  
0.2727272727272726
```

همانطور که در بالا نیز آورده شده بهترین مدل اولی می‌باشد که از TF-IDF به همراه logistic regression استفاده شده است و دارای  $f1 = 59$  می‌باشد.