

به نام خدا



دانشگاه تهران  
پردیس دانشکده‌های فنی  
دانشکده برق و کامپیوتر



درس کلان داده

پروژه پایانی

مرداد ماه ۱۳۹۹

## فهرست گزارش سوالات

۳	گام اول – دریافت اطلاعات و PreProcess
۳	قدم اول: خزش پیام رسان سروش
۳	قدم دوم: ذخیره اطلاعات در Kafka
۵	گام دوم – persistence
۱۷	گام سوم – ChannelHistory
۲۲	گام چهارم – statistics
۳۲	گام پنجم – Analytics
۳۶	ساخت داشبوردهای مدیریتی
۴۶	ساخت یک مدل پیش بینی کننده با اسپارک – دارای امتیاز
۴۹	پیوست – کار با ابزارها
۴۹	کار با Kafka
۵۱	کار با Cassandra
۵۲	کار با Clickhouse
۵۳	کار با Power BI
۵۵	کار با WSL
۵۵	تنظیمات ODBC
۵۷	مراجع

## گام اول – دریافت اطلاعات و PreProcess

ادرس فایل ریپوزیتوری گیت پروژه :

<https://github.com/fatemeh-saligheh/Immediate-system-for-instantaneous-analysis-of-internal-messenger-data>

### قدم اول: خزش پیام رسان سروش

ابتدا تعدادی از سایت هایی که کانال های سروش را معرفی کردند را پیدا کرده و سپس با استفاده از کد python به api این سایت ها متصل شده ایم. از صفحات مربوطه، آی دی های کانال ها را استخراج کرده و ذخیره کردیم. به تعداد بسیار خوبی کانال دست پیدا کردیم که از آن ها شروع به دریافت پیام کردیم. به این صورت که به api سروش وصل شده و پست جدید وارد شده در این کانال ها را دریافت کرده و آن ها را در kafka ذخیره کرده ایم. کدهای این قسمت در فایل step1/find\_channels.ipynb ذخیره شده است.

دریافت اطلاعات را از طریق یک حلقه همیشه true انجام داده و در مرحله بعد وارد کانال preprocess می شویم. در این قسمت هشتگ ها را در متن پیام پیدا نموده و همچنین خود پیام های دریافتی که در قالب json هستند دارای کلمات کلیدی می باشند و فیلد keyword این کلمات را مشخص می نماید. سپس به سراغ جستجوی کلمات معرفی شده در صورت مسئله می پردازیم و همه ی این موارد را به عنوان فیلد metadata به پیام اضافه می کنیم. در متن پیام url ها را جستجو نموده و در فیلدی به نام links می گذاریم سپس در فیلد file.extention را می بینیم، اگر شامل ادرس فایل jpg بود آن را به فیلد images اضافه می کنیم.

### قدم دوم: ذخیره اطلاعات در Kafka

پلت فرم کافکا قابل استفاده در لینوکس و ویندوز می باشد [1] و می توان در ترمینال و یا cmd دستورات آن را اجرا کرد. در پیوست نحوه ی اجرای کافکا در ویندوز آمده است. همچنین می توان با استفاده از کتابخانه pykafka نیز عملیات ایجاد topic و ذخیره سازی را انجام داد. نحوه نصب و کار با آن ها در پیوست آمده است.

برای اتصال به کافکا لازم است تا به پورت 9092 از localhost متصل شویم :

```
client = KafkaClient(hosts="127.0.0.1:9092")
topic = client.topics["final"]
```

برای نوشتن اطلاعات در کافکا لازم است تا یک producer تعریف کنیم تا پیام های دریافتی را به کافکا بدهد :

```
with topic.get_sync_producer() as producer:|  
    producer.produce(bytes(str(out['messages'][0]), encoding='utf-8'))
```

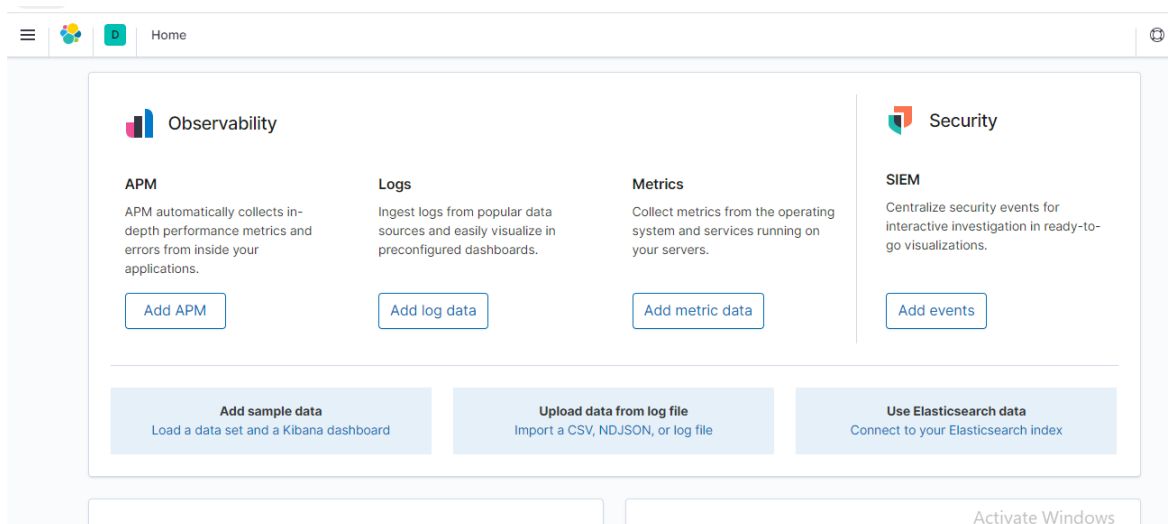
با هربار دریافت پیام جدید، برنامه آن را با producer ذخیره می کنیم. سپس می توان با استفاده از consumer پیام های ذخیره شده در هر topic را به دست آورد و آن را در فریم ورک های دیگری مانند الاستیک سرچ، کاساندرا و ... استفاده کرد. در واقع کافکا بستری توزیع شده است که برای انتقال داده ها از آن استفاده می شود.

## گام دوم – persistence

در این قسمت لازم است تا elasticsearch و kibana را اجرا نماییم . نکته ای که وجود دارد این است که برای کار با الاستیک ۸ لازم است تا نسخه ۱۱ به بعد جاوا را داشته باشیم . حال برای اجرای الاستیک سرچ می توان آن را از این آدرس [2] دانلود نموده و سپس فایل elasticsearch.bat را اجرا نماییم در صورتی که الاستیک سرچ بدون خطا اجرا شود در localhost:9200 قابل مشاهده است :

```
{
  "name" : "DESKTOP-IED3F77",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "k-TFHmpBS3CRZduGFKYA7A",
  "version" : {
    "number" : "7.8.0",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "757314695644ea9a1dc2fec26d1a43856725e65",
    "build_date" : "2020-06-14T19:35:50.234439Z",
    "build_snapshot" : false,
    "lucene_version" : "8.5.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

سپس kibana را از آدرس [3] دانلود نموده و سپس فایل kibana.bat را اجرا می کنیم که در این صورت در آدرس localhost:5601 قابل مشاهده است :



در این قسمت می توان از پایتون استفاده نموده و یک consumer برای کافکا تعریف نماییم تا اطلاعات را در الاستیک سرچ به صورت real time ذخیره نماید . برای کار با الاستیک سرچ از طریق پایتون لازم است تا کتابخانه elasticsearch را نصب نماییم بدین منظور از کامند زیر استفاده می نماییم :

```
pip install elasticsearch
```

حال یک consumer تعریف نموده تا به صورت real time اطلاعات را از کافکا بخواند در الاستیک سرچ قرار دهد :

```
client = KafkaClient(hosts="127.0.0.1:9092")
topic = client.topics["final"]
consumer = topic.get_simple_consumer()
```

چون در هنگام ارسال هر پیام به کافکا به صورت utf-8 کد کرده بودیم حال هر پیامی که دریافت می کنیم را انکود می نماییم :

```
msg = (message.value).decode("utf-8")
```

سپس چون فرمت پیام دریافت شده string است برای تبدیل آن به json از ماژول ast استفاده می نماییم :

```
msg = ast.literal_eval(msg)
```

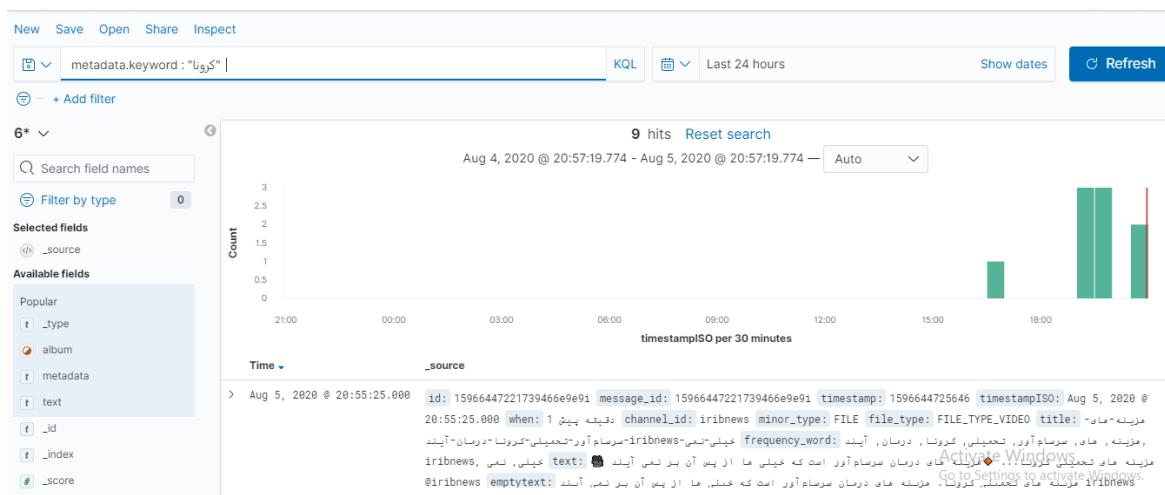
سپس به صورت زیر می توان پیام دریافتی را در الاستیک سرچ و در index مورد نظر ذخیره نمود .

```
es = Elasticsearch()
|
es.index(index=index, id=esid, body=msg)
```

حال پس از ذخیره اطلاعات نوبت به استفاده از kibana برای ایجاد داشبورد می رسد .

Kibana از چهار بخش Discover ، Visualize ، Dashboard و Settings تشکیل شده است .

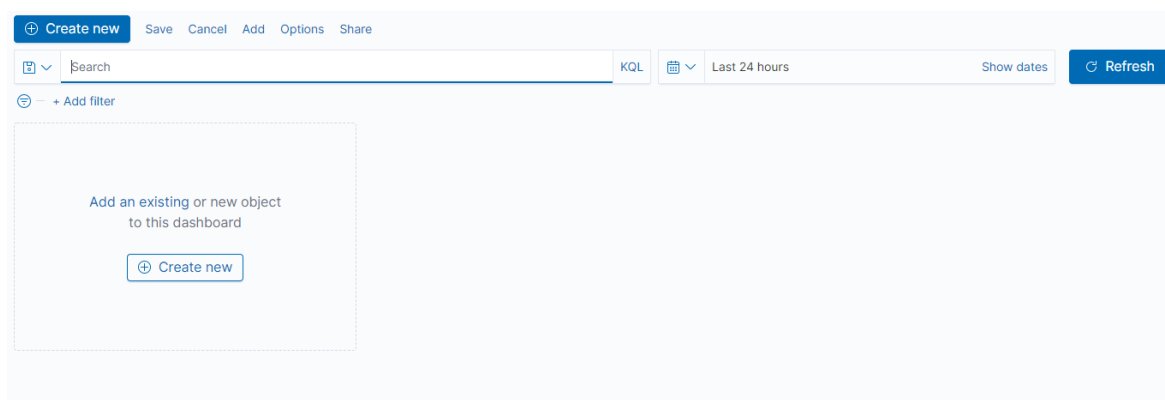
در بخش Discover تمام لاگ های دریافتی توسط الاستیک سرچ نشان داده می شود . ما می توانیم با استفاده از فیلتر و بخش کوئری پیام های مورد نظر را نشان داده و هم چنین می توانیم از طریق فیلتر زمانی آنها را فیلتر کنیم . برای نمونه می توان شکل زیر را مشاهده نمود :



در بخش visualize می توان visualization هایی را ایجاد تغییر و یا مشاهده نمود . چند نوع مختلف visualization از جمله نمودار دایره ای و ستونی نقشه و جدول و ... وجود دارد

در بخش Dashboard می توان چند visualization را به طور هم زمان نمایش داد . که ما در این پروژه از این قسمت استفاده می کنیم و در ادامه با بخش های مختلف آن آشنا می شویم .

برای ایجاد داشبورد در بخش Dashboards روی Create Dashboard کلیک میکنیم . صفحه زیر ایجاد می شود که می توان با استفاده از create new یک visualization جدید ایجاد نمود و یا می توان اگر یک search قبلا در بخش های دیگر انجام گرفته و ذخیره شده است را انتخاب نماییم تا در داشبورد نمایش داده شود .



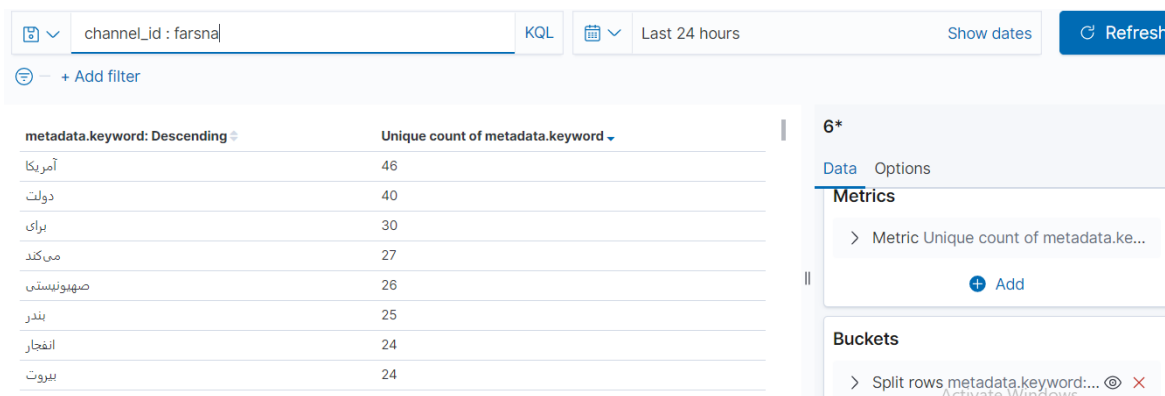
ابتدا می توان با استفاده از کد زیر می توان پیش پردازش های لازم را با استفاده از Persian analyzer انجام داد :

```
PUT /persian_example
{
  "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"
          ]
        }
      }
    }
  }
}
```

الف) ابر کلمات یک کانال یا خبرگزاری خاص در یک بازه زمانی

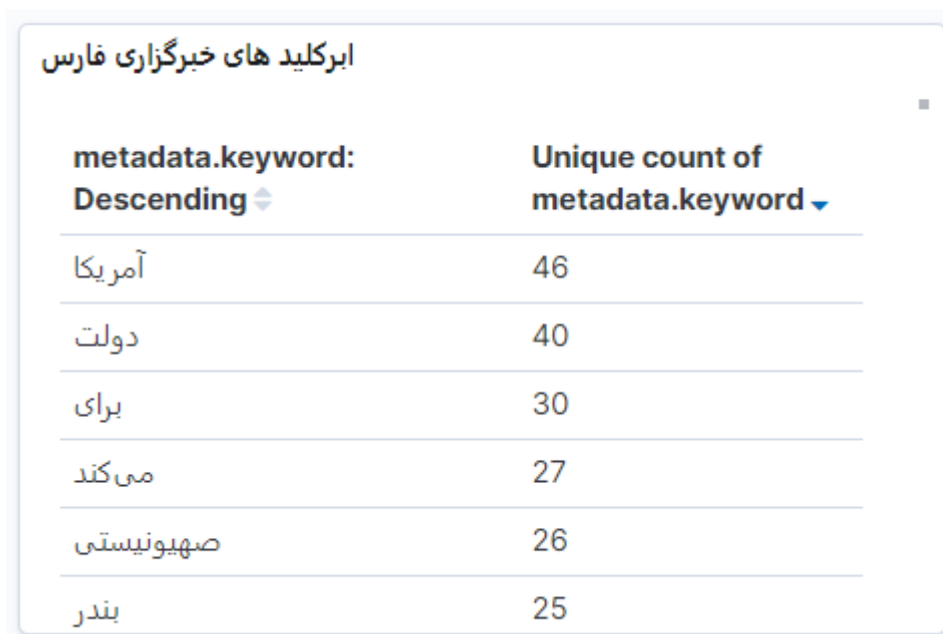
برای این بخش در بخش visualize یک table ایجاد می کنیم سپس در بخش سرچ کوئری را به این صورت در نظر می گیریم که پیام هایی که channel\_\_id آنها مثلا farsna است . سپس در سمت راست metric را unique count of metadata.keyword در نظر می گیریم یعنی به صورت uniq تعداد هر ابرکلمه را بشمار سپس در بخش bucket هم می گوئیم که فیلد metadata را نمایش بده . سپس در بخش فیلتر زمانی بازه زمانی را مشخص می کنیم مثلا یک روز گذشته .





حال می‌توان visualization ایجاد شده را ذخیره نمود و در داشبورد آن را استفاده نمود.

نتیجه در داشبورد به صورت زیر خواهد بود:



ب) متن ده پست اخیری که دریافت شده است

در بخش visualization یک table ایجاد می‌کنیم. هر پیام دارای یک مقدار timestamp است که می‌توان ترتیب دریافت اطلاعات را از آن طریق متوجه شد. در table ایجاد شده در بخش اول که metric را مشخص می‌کنیم، می‌گوییم که textها را بر اساس timestamp صورت نزولی مرتب کند سپس ۱۰ ردیف آخر را نشان دهد. در قسمت bucket هم می‌گوییم که row split انجام داده و timestamp را به صورت نزولی مرتب و ۱۰ تای اول را نشان دهد. بنابراین می‌توان ۱۰ پست آخر را مشاهده نمود.

timestampISO:
Descending ▾ Last 10 text ▾
Aug 6, 2020 @ 12:35:09.000 🗨️ رئیس جمهوری: شرکت های دولتی بعد از سرمایه گذاری و راه اندازی شرکت ها، کم کم باید کنار بروند ، فقط سهامداران @iribnews باشند و مدیریت این شرکت ها را واگذار کنند
Aug 6, 2020 @ 12:28:00.000 ⚽️ زمان قرعه کشی مسابقات جام حذفی ◀️ قرعه کشی مرحله نیمه نهایی جام حذفی (یادواره آزادسازی خرمشهر) فصل ۹۸-۹۹ ساعت ۱۵ روز سه شنبه ۲۱ مرداد، در سالن روابط عمومی سازمان لیگ فوتبال ایران و با حضور نمایندگان باشگاه های حاضر در این مرحله برگزار می شود. ◀️ تیم های تراکتور تبریز، پرسپولیس، نفت مسجدسلیمان و برنده مسابقه دو تیم زمان @iribnews، استقلال و سپاهان در این مرحله حضور دارند که براساس قرعه حریفان شان مشخص خواهد شد قرعه کشی مسابقات جام حذفی ◀️ قرعه کشی مرحله نیمه نهایی جام حذفی (یادواره آزادسازی خرمشهر) فصل ۹۸-۹۹ ساعت ۱۵ روز سه شنبه ۲۱ مرداد، در سالن روابط عمومی سازمان لیگ فوتبال ایران و با حضور نمایندگان باشگاه های حاضر در این مرحله برگزار می شود. ◀️ تیم های تراکتور تبریز، پرسپولیس، نفت مسجدسلیمان و برنده مسابقه دو تیم استقلال و @iribnews، سپاهان در این مرحله حضور دارند که براساس قرعه حریفان شان مشخص خواهد شد
Aug 6, 2020 @ 12:22:36.000 🗨️ رئیس جمهور : در بسیاری از زمینه ها به لطف شرکت های دانش بنیان به خودکفایی رسیده ایم و این یکی از راه های...

6\*

Data Options

Metrics

> Metric Last 10 text

+ Add

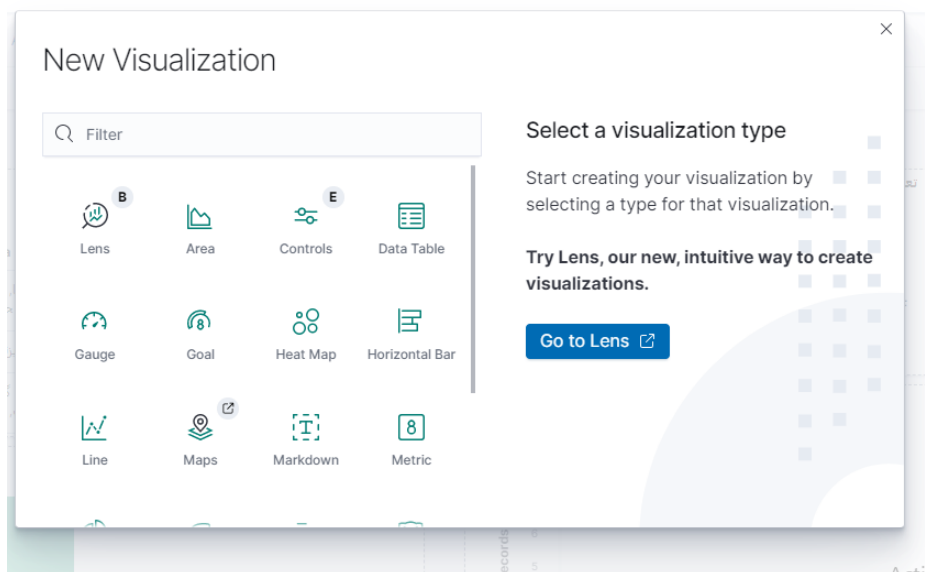
Buckets

> Split rows timestampISO: Des... Ⓞ ✕

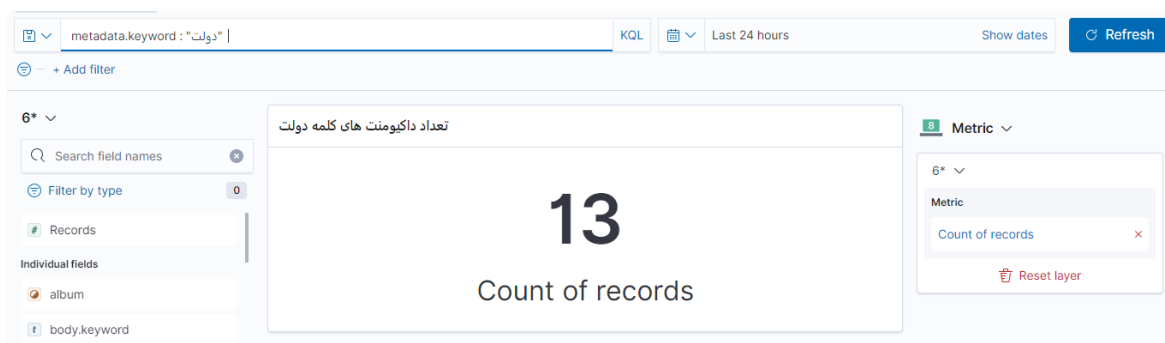
خروجی در داشبورد به صورت زیر است :

ده پست اخیر
timestampISO:
Descending ▾ Last 10 text ▾
Aug 6, 2020 @ 12:35:09.000 🗨️ رئیس جمهوری: شرکت های دولتی بعد از سرمایه گذاری و راه اندازی شرکت ها، کم کم باید کنار بروند ، فقط سهامداران باشند و مدیریت این شرکت ها را واگذار کنند @iribnews
Aug 6, 2020 @ 12:28:00.000 ⚽️ زمان قرعه کشی مسابقات جام حذفی ◀️ قرعه کشی مرحله نیمه نهایی جام حذفی (یادواره آزادسازی خرمشهر) فصل ۹۸-۹۹ ساعت ۱۵ روز سه شنبه ۲۱ مرداد، در سالن روابط عمومی سازمان لیگ فوتبال ایران و با حضور نمایندگان باشگاه های حاضر در این مرحله برگزار می شود. ◀️ تیم های تراکتور تبریز، پرسپولیس، نفت مسجدسلیمان و برنده مسابقه دو تیم استقلال و @iribnews، سپاهان در این مرحله حضور دارند که براساس قرعه حریفان شان مشخص خواهد شد

ج) یکی از ویژگی های کیبانا آن است که می توان در بخش create new از داشبورد از ابزار Lens استفاده نمود :



در این بخش از ابزار lens استفاده نموده ایم . لازم است تا metadata.keyword شامل کلمه مورد نظر باشد مثلا دولت یا کرونا یا ... . بازه زمانی با استفاده از فیلترزمان در بالا سمت راست مشخص خواهد شد .

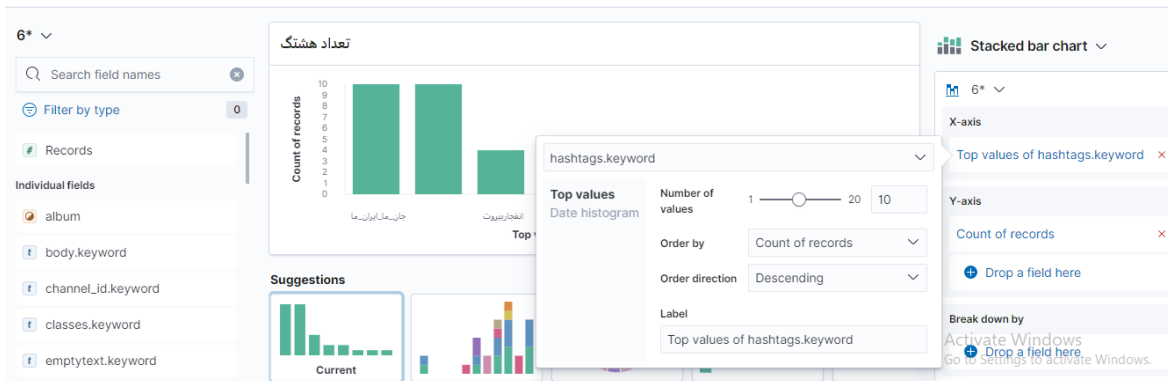


و نتیجه در داشبورد به صورت زیر خواهد بود :

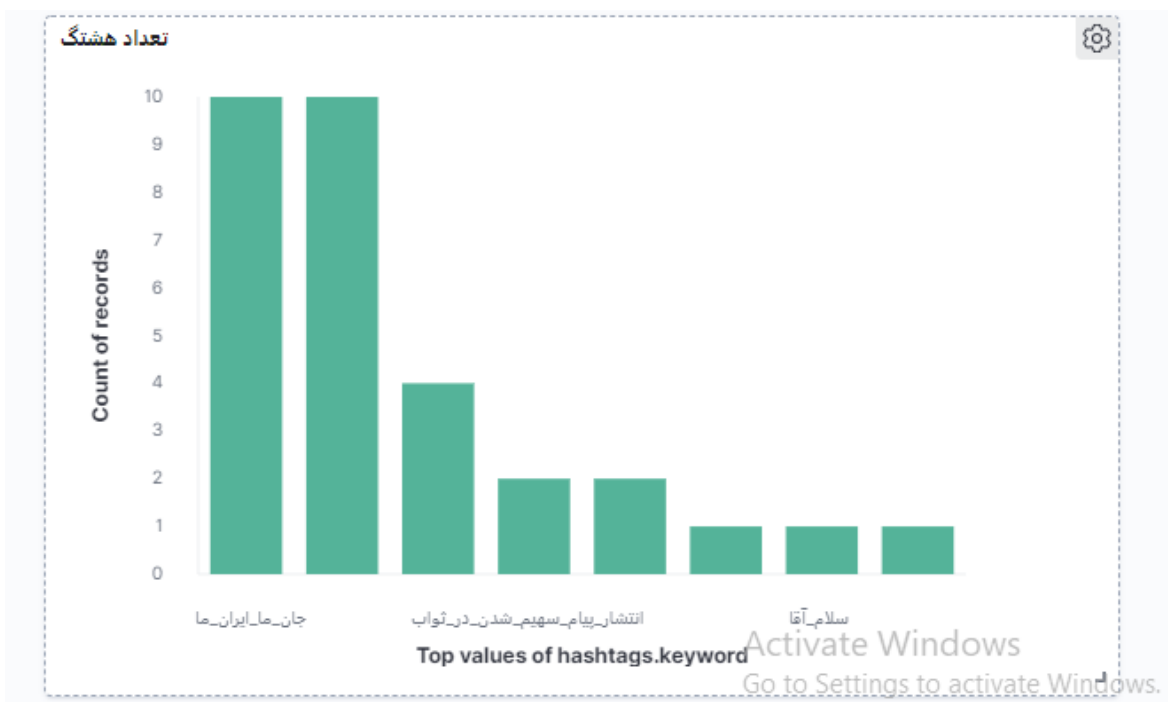


د) 10 هشتگ یا نماد بررسی بیشتر استفاده شده .

این بخش را با استفاده از Lens ایجاد میکنیم . به این صورت که یک نمودار ستونی را انتخاب می نماییم سپس محور x ها را hashtags.keyword در نظر می گیریم و به صورت نزولی بر اساس تعداد مرتب می کنیم و سپس ۱۰ مقدرا از آن را نمایش می دهیم

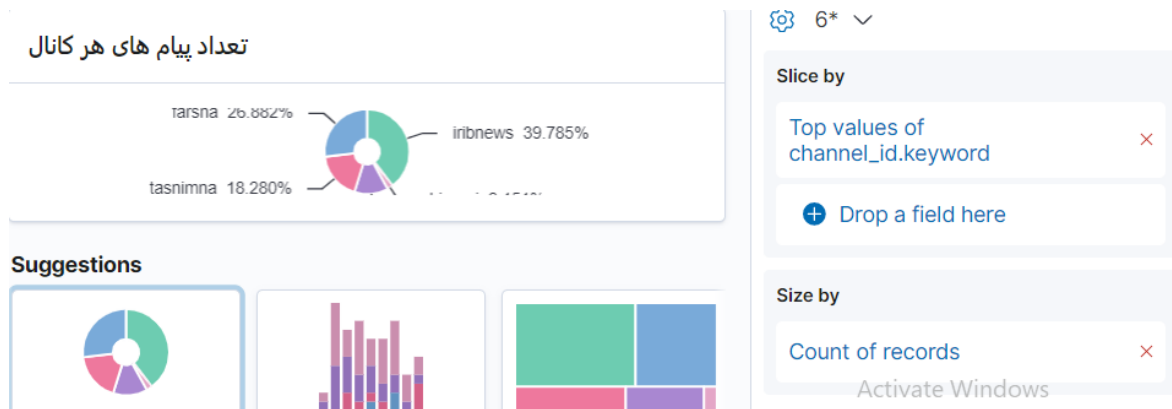


نتیجه در داشبورد به صورت زیر خواهد بود :

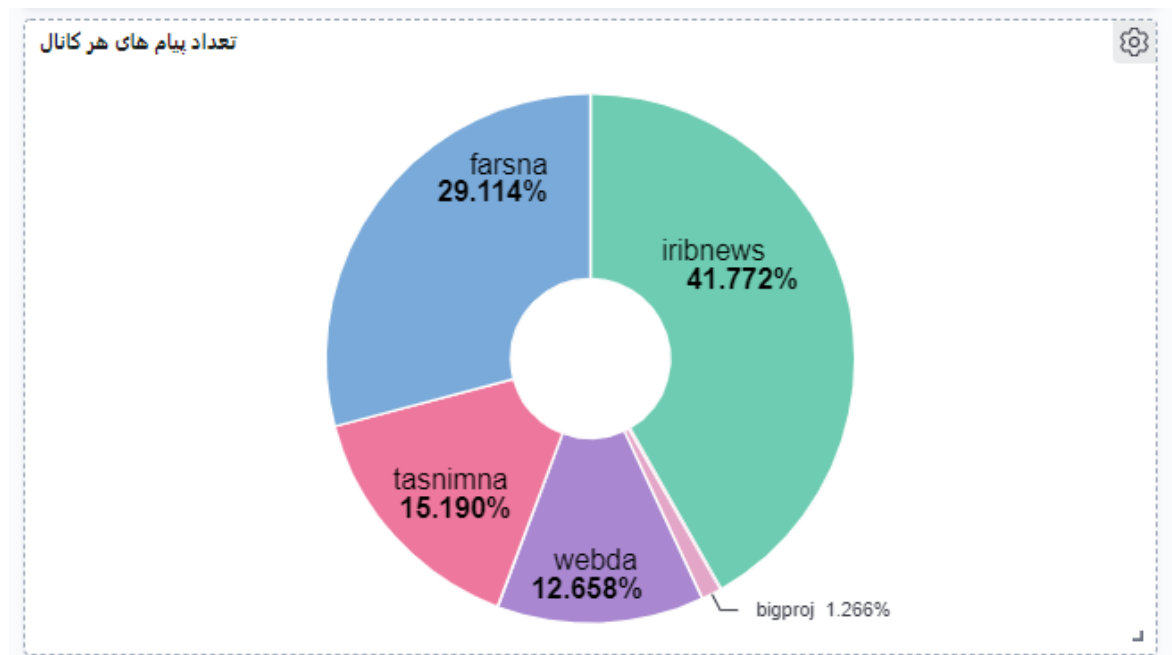


ه ) یک نمودار به انتخاب خودتان : درصد پیام های هر کانال

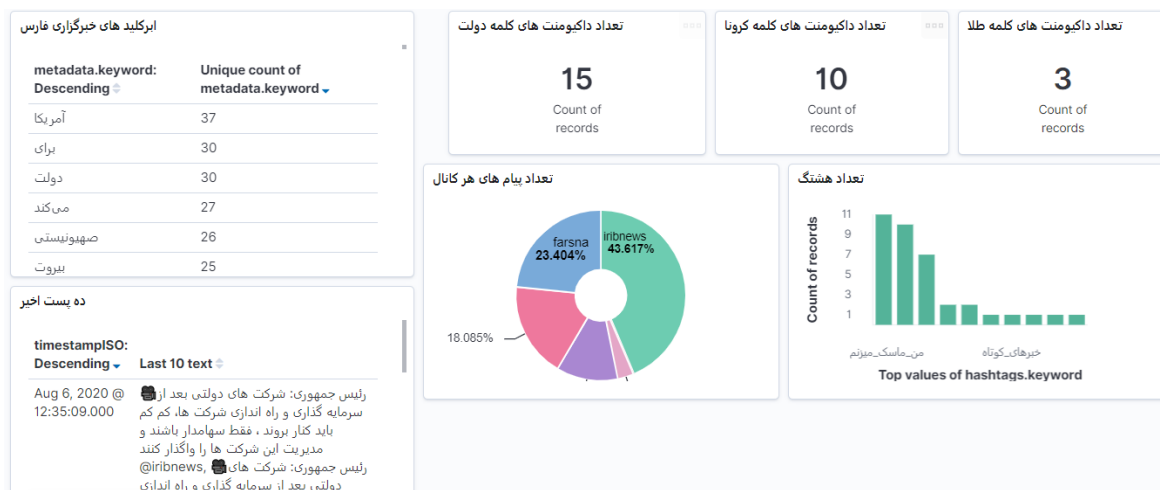
در این قسمت هم از lens استفاده نمودیم. گفته شده خبرگزاری های با بالاترین تعداد را نشان بده . با استفاده از فیلد channel\_id.keyword .



خروجی در داشبورد به صورت زیر است .



داشبورد ایجاد شده به صورت زیر خواهد بود :

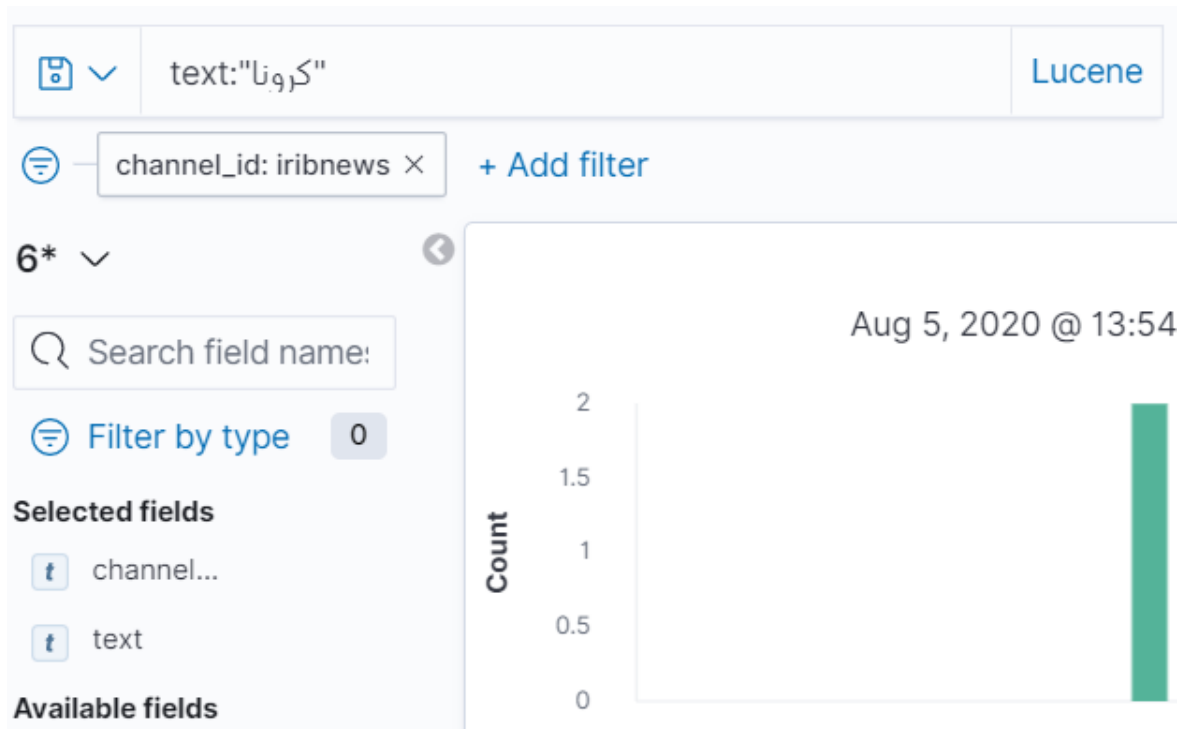


اگر به دنبال پست های یک کلمه خاص از یک خبرگزاری خاص هستیم

می توان از قسمت console و با استفاده از کوئری های DSL کوئری را به صورت زیر ایجاد نمود . must برای عملگر AND استفاده شده و از match برای آنکه در فیلد text کلمه کرونا را جستجو کنیم . سپس نتایج را با استفاده از filter فیلتر می کنیم یعنی می گوییم آن نتایجی که channel\_id آن ها iribnews است را فیلتر کن. از source هم استفاده شده است تا فقط فیلدهای timestampISO , channel\_id , text نمایش داده شود .

```
GET 6/_search
{
  "_source": ["channel_id", "text", "timestampISO"],
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "text": "کرونا"
          }
        }
      ],
      "filter": [
        {
          "term": {
            "channel_id": "iribnews"
          }
        }
      ]
    }
  },
  "size": 10000
}
```

هم چنین می توان کوئری خواسته شده را در داشبورد نیز نمایش داد . در بخش Discover به صورت شکل زیر نتایج را به دست می آوریم . از کوئری Lucene استفاده می کنیم و می گوئیم آن هایی را بده که در text کلمه کرونا وجود دارد سپس یک فیلتر ایجاد می کنیم که آن هایی را که channel\_id آن ها iribnews است را نگهدار.



در داشبورد هم به صورت زیر نمایش داده می شود .

پست های خبرگزاری صداسیما شامل کلمه کرونا

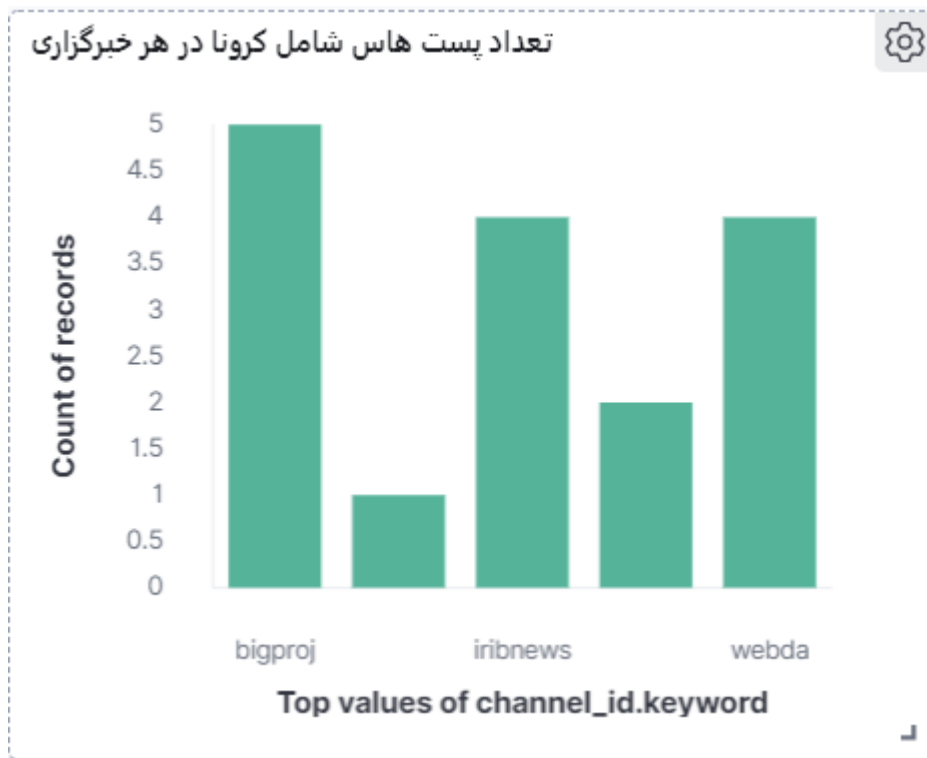
1-5 of 5

text	channel_id
> خبرهاي_کوتاه#	iribnews
<ul style="list-style-type: none"> <li>دستور رئیس جمهور به وزیر ارتباطات جهت اهداء اینترنت رایگان به خبرنگاران به مناسبت روز خبرنگار</li> <li>فرمانده ستاد مقابله با کرونا در تهران: متوفیان ۱۰ درصد نسبت به هفته گذشته کاهش داشته است</li> <li>وزارت اقتصاد در بخشنامه ای از ۱۷ بانک کشور خواست سهام عدالت را به عنوان وثیقه برای صدور کارت اعتباری</li> </ul>	
> خبرهاي_کوتاه#	iribnews
<ul style="list-style-type: none"> <li>دستور رئیس جمهور به وزیر ارتباطات جهت اهداء اینترنت رایگان به خبرنگاران به مناسبت روز خبرنگار</li> <li>فرمانده ستاد مقابله با کرونا در تهران: متوفیان ۱۰ درصد نسبت به هفته گذشته کاهش داشته است</li> <li>وزارت اقتصاد در بخشنامه ای از ۱۷ بانک کشور خواست سهام عدالت را به عنوان وثیقه برای صدور کارت اعتباری</li> </ul>	

تعداد پست های شامل یک کلمه خاص به ازای هر کانال

در این قسمت هم از lens استفاده می کنیم . محور x را channel\_id در نظر گرفته و محور عمودی را تعداد record ها و در قسمت سرچ هم از lucene استفاده نموده و می گوییم "کرونا" . text: بدین معنی که هر پستی که در متن آن کلمه کرونا آمده باشد .

در داشبورد نمودار به صورت زیر خواهد بود :





## گام سوم – ChannelHistory

در این گام سعی شده است که داده های ذخیره شده در کافکا را به Cassandra منتقل کنیم و پرس جوهای لازم را از آن جا استخراج کنیم. نحوه ی نصب آن در پیوست آمده است.

Cassandra یک پایگاه داده توزیع شده است که مقیاس پذیری بالایی دارد و برای مدیریت داده های ساخت یافته حجیم مورد استفاده قرار می گیرد. کاساندرا داده ها را به صورت مرتب ذخیره کرده و به علت نبود عملیات هایی مانند join، جست و جو و استخراج اطلاعات با سرعت بالایی انجام می گیرد. کاساندرا بیشتر مناسب برای ذخیره داده هایی هست که حالت سری زمانی دارد و کوئری براساس یک ترتیب زمانی خواسته می شود. در واقع یک ایندکس بر روی داده ها ایجاد می کند تا در هنگام کوئری زدن، ایندکس های مربوطه بازیابی می شود.[4]

برای انجام کوئری های این گام، ابتدا در پایتون با دستورات زیر جدول های مورد نظرم را ایجاد کردیم:

```
from cassandra.cluster import Cluster

cluster = Cluster()
session = cluster.connect()
session.execute("USE zh;")
table1 = """CREATE TABLE all_posts(date text , hour text, message_id text,
channel_id text
,PRIMARY KEY((date),hour, channel_id,message_id)
);"""
session.execute(table1)

table2 = """CREATE TABLE all_channels(channel_id text ,date text,hour text,
message_id text
,PRIMARY KEY((channel_id),date,hour,message_id)
);"""
session.execute(table2)

table3 = """CREATE TABLE all_hashtags(hashtag_name text,date text,hour text
, message_id text
,PRIMARY KEY((hashtag_name),date,hour,message_id)
);"""
session.execute(table3)
```

در ایجاد جدول ها بسته به کوئری هایی که نیاز داریم، لازم است که ستون ها و key هایش را متناسب با آن ها ایجاد کنیم. کلید های داخل پرانتز partition key هستند و بقیه آن ها clustering key هستند.

جدول های ایجاد شده به صورت زیر است:

```
cqlsh:zh> select * from all_posts;
```

```
date          | hour | channel_id | message_id
```

```
cqlsh:zh> select * from all_channels;
```

```
channel_id | date          | hour | message_id
```

```
cqlsh:zh> select * from all_hashtags;
```

```
hashtag_name | date          | hour | message_id
```

کدهای این قسمت در step3/create\_table.py ذخیره شده است.

حال باید این جداول را با مقادیری که از consumer کافکا دریافت می کنیم، پر کنیم.

برای این کار consumer کافکا را اجرا می کنیم:

```
from pykafka import KafkaClient
client = KafkaClient(hosts="127.0.0.1:9092")
topic = client.topics["test4"]
consumer = topic.get_simple_consumer()
```

و همین طور به کاساندرا متصل می شود و keyspace آن را وارد می کنیم:

```
from cassandra.cluster import Cluster

cluster = Cluster()
session = cluster.connect()
session.execute("USE zh;")
```

اینک از consumer پیام های ذخیره شده را دریافت کرده و براساس ساختار پیام های ذخیره شده، اطلاعات مورد نیاز برای پر کردن جداول را دریافت می کنیم و در هر دریافت، وارد جداول کرده و براساس ساختار جداول ها آن ها را پر می کنیم:

```
insert1 = """INSERT INTO all_posts(date, hour, message_id , channel_id) VALUES(%s,%s, %s, %s);"""
session.execute(insert1,(date,hour,message_id,channel_id))

insert2 = """INSERT INTO all_channels(channel_id ,date, hour, message_id) VALUES(%s,%s,%s,%s);"""
session.execute(insert2,(channel_id,date,hour, message_id))
```

```
x = hashtags.split(", ")
for h in x:
    if h != "":
        insert3 = """INSERT INTO all_hashtags(hashtag_name ,date, hour, message_id)
        VALUES(%s, %s,%s,%s);"""
        session.execute(insert3,(h[1:-1],date,hour,message_id))
```

کدهای این قسمت در فایل step3/consumer.py ذخیره شده است.

نمونه ای از پر شدن جدول ها:

```
cqlsh:zh> select * from all_posts;
```

date	hour	channel_id	message_id
2020-08-03	22	bigproj	15964922045871696auuti
2020-08-04	10	farsna	1596538644667d7faffyft
2020-08-04	10	tribnews	1596538733019e7500iHRk
2020-08-04	10	tasnimna	1596538645441d36b2x39y
2020-08-04	11	farsna	1596539195345d7faf2dxn
2020-08-04	11	farsna	1596540569815d7fafDMPV
2020-08-04	11	farsna	1596541628291d7fafFqf5
2020-08-04	11	farsna	1596542012347d7fafel8c
2020-08-04	11	tribnews	1596539409361e7500Q0Jn
2020-08-04	11	tribnews	1596540689581e7500YG9M
2020-08-04	11	tribnews	1596541337275e7500ck4I
2020-08-04	11	tribnews	1596541817748e75005Jlu
2020-08-04	11	tasnimna	1596539194063d36b2dhyb
2020-08-04	11	tasnimna	1596539353820d36b26n9f
2020-08-04	11	tasnimna	1596539510020d36b24r1l
2020-08-04	11	tasnimna	1596541331677d36b26hzt
2020-08-04	11	tasnimna	1596541989217d36b2m9ib
2020-08-04	11	tasnimna	1596542230897d36b2g2ry

```
cqlsh:zh> select * from all_channels;
```

channel_id	date	hour	message_id
tasnimna	2020-08-04	10	1596538645441d36b2x39y
tasnimna	2020-08-04	11	1596539194063d36b2dhyb
tasnimna	2020-08-04	11	1596539353820d36b26n9f
tasnimna	2020-08-04	11	1596539510020d36b24r1l
tasnimna	2020-08-04	11	1596541331677d36b26hzt
tasnimna	2020-08-04	11	1596541989217d36b2m9ib
tasnimna	2020-08-04	11	1596542230897d36b2g2ry
tasnimna	2020-08-04	12	1596542630688d36b2cagr
tasnimna	2020-08-04	12	1596542837524d36b2c81c
tasnimna	2020-08-04	12	1596544705979063e7ux3q
tasnimna	2020-08-04	12	1596544844569063e70ebd
tasnimna	2020-08-04	13	1596547326485063e7akt3
tasnimna	2020-08-04	13	1596548822987063e7svml
bigproj	2020-08-03	22	15964922045871696auuti
tribnews	2020-08-04	10	1596538733019e7500iHRk
tribnews	2020-08-04	11	1596539409361e7500Q0Jn

```
cqlsh:zh> select * from all_hashtags;
```

hashtag_name	date	hour	message_id
لوي	2020-08-04	10	1596538645441d36b2x39y
هنگی اویخ	2020-08-04	12	15965427369837f980loyh
هنگی اویخ	2020-08-04	11	1596541337275e7500ck4I

حال می توان کوئری های مورد نظر را از جدول ها استخراج کرد.

ابتدا به کاساندریا متصل می شویم:

```
cluster = Cluster()
session = cluster.connect()
session.execute("USE zh;")
```

از جمله کوئری هایی که از این جدول ها قابل استخراج هستند:

پست هایی که در یک ساعت اخیر اتفاق افتاده است:

```
query1 = """select message_id from all_posts where
date = ? and hour = ?;"""
st = session.prepare(query1)

now = datetime.now()
dt_string = now.strftime("%Y-%m-%d %H:%M:%S")
x = dt_string.split(' ')
messages = session.execute(st, [x[0],x[1][:2]])
print("all message_ids in last hour")
for m in messages:
    print(m)
```

پست هایی که در ۲۴ ساعت اخیر اتفاق افتاده است:

```
query2 = """select message_id from all_posts where
date = ?;"""
st2 = session.prepare(query2)

now = datetime.now()
dt_string = now.strftime("%Y-%m-%d %H:%M:%S")
x = dt_string.split(' ')
messages = session.execute(st2, [x[0]])
print("all message_ids in last 24-hours")
for m in messages:
    print(m)
```

پست هایی که در ۲۴ ساعت اخیر هشتگ "کرونا" دارند:

```
query3 = """select message_id from all_hashtags where
date = ? and hashtag_name = ?;"""
st3 = session.prepare(query3)

now = datetime.now()
dt_string = now.strftime("%Y-%m-%d %H:%M:%S")
```

```
x = dt_string.split(' ')
messages = session.execute(st3, [x[0], "کرونا"])
print("all message_ids in last 24-hours that have #کرونا")
for m in messages:
    print(m)
```

تمام پست هایی که در ۲۴ ساعت اخیر، کانال منتشر کرده است:

```
query4 = """select message_id from all_channels where
date = ? and channel_id = ?;"""
st4 = session.prepare(query4)

now = datetime.now()
dt_string = now.strftime("%Y-%m-%d %H:%M:%S")
x = dt_string.split(' ')
messages = session.execute(st4, [x[0], "farsna"])
print("all message_ids in last 24-hours from farsna")
for m in messages:
    print(m)
```

همان طور که می بینید کوئری هایی که مرتبط با ساختار جدول هستند و می توان جواب آن ها را از جداول پیدا کرد، قابل اجرا می باشند. حتی می توان بازه های زمانی، مد نظر را در آن ها به همین صورت مورد پرس و جو قرار داد.

کدهای این قسمت در فایل step3/query.py قرار دارد.

می توان بعضی از اطلاعات آماری را نیز از این جداول استخراج کرد. همانند زبان های SQL، می توان از دستورات COUNT برای پیدا کردن تعداد کوئری های مورد سوال، AVG میانگین، MAX یا MIN و همین طور بازه هایی که از primary key مورد سوال واقع شدند، قابلیت پاسخ گویی هستند. برای پرس و جو های خارج از ساختار این جداول، لازم است که جدول های جدیدی را بسته به نوع پرس و جو ایجاد و پر کنیم.

ردیس یک بانک اطلاعاتی in-memory است که از نوع key-value می باشد . این باک اطلاعاتی رایگان و open source است . ویژگی in-memory بودن باعث افزایش سرعت و بهبود کارایی در پاسخ دهی می باشد . از مزایای دیگر ردیس آن است که از data type های مختلفی پشتیبانی می کند .

### Data Types: [5]

#### ۱. Strings :

String ها مجموعه ای از بایت ها هستند که می توانند حداکثر ۵۱۲ مگابایت باشند . برای نمونه در شکل زیر یک کلید به نام name و که مقدار string آن tutorialspoint است و با استفاده از set می توان آن را ایجاد نمود و با استفاده از get و کلید می توان مقدار را دریافت نمود .

```
redis 127.0.0.1:6379> SET name "tutorialspoint"
OK
redis 127.0.0.1:6379> GET name
"tutorialspoint"
```

دیگر توابعی که برای strings می توان استفاده نمود به صورت زیر هستند :

APPEND, BITCOUNT, BITFIELD, BITOP, BITPOS, DECR, DECRBY, GET, GETBIT, GETRANGE, GETSET, INCR, INCRBY, INCRBYFLOAT, MGET, MSET, MSETNX, PSETEX, SET, SETBIT, SETEX, SETNX, SETRANGE, STRLEN

#### ۲. Sets

این نوع در واقع یک کالکشن از string ها است . با توجه به مثال زیر یک کالکشن با نام myset داریم و عضو های ۱ و ۲ و ۳ را با استفاده از SADD به آن اضافه می کنیم . سپس با استفاده از smember می توان مقادیر آن را مشاهده نمود .

```
> sadd myset 1 2 3
(integer) 3
> smembers myset
1. 3
2. 1
3. 2
```

از دیگر توابع مربوط به set ها :

SADD, SCARD, SDIFF, SDIFFSTORE, SINTER, SINTERSTORE, SISMEMBER, SMEMBERS, SMOVE, SPOP, SRANDMEMBER, SREM, SSCAN, SUNION, SUNIONSTORE

### ۳. Lists :

لیست ها در واقع مجموعه ای از string ها هستند که به ترتیب اضافه شدن در لیست مرتب شده اند و می توان به اول و آخر لیست یک عضو جدید اضافه نمود . در زیر مثالی را می بینیم . می توان از RPUSH برای اضافه نمودن عضو جدید به انتهای لیست و LPUSH برای اضافه نمودن عضو جدید به ابتدای لیست استفاده نمود . از LRANGE هم برای دریافت اعضای متعلق به یک بازه استفاده می شود .

```
> rpush mylist A
(integer) 1
> rpush mylist B
(integer) 2
> lpush mylist first
(integer) 3
> lrange mylist 0 -1
1) "first"
2) "A"
3) "B"
```

از دیگر توابع مربوط به لیست ها می توان به موارد زیر اشاره نمود :

BLPOP, BRPOP, BRPOLPUSH, LINDEX, LINSERT, LLEN, LPOP, LPUSH, LPUSHX, LRANGE,  
LREM, LSET, LTRIM, RPOP, RPOPLPUSH, RPUSH, RPUSHX

#### Sortes Sets.۴

شبیه sets است با این تفاوت که یک مقدار score دارد و به صورت مرتب شده براساس score قرار می گیرند و نمی توان عضو تکراری به آن اضافه نمود . برای نمونه مثال زیر آورده شده است . با استفاده از ZADD می توان عضو هایی به set اضافه نمود و با استفاده از ZRANGE می توان یک range از اعضا رانمایش داد .

```
redis> ZADD myzset 1 "one"
(integer) 1
redis> ZADD myzset 1 "uno"
(integer) 1
redis> ZADD myzset 2 "two" 3 "three"
(integer) 2
redis> ZRANGE myzset 0 -1 WITHSCORES
1) "one"
2) "1"
3) "uno"
4) "1"
5) "two"
6) "2"
7) "three"
8) "3"
...~
```

#### Hashes.۵

برای عمل mapping بین مقدار field و value است . برای نمونه در زیر می بینیم که یک کلید داریم به نام user:1000 که می توان نام خانوادگی و تاریخ تولد را برای آن set نمود . و با استفاده از hset , hget می توان اطلاعات را ذخیره و دریافت نمود . اگر چند فیلد باشد از hmset استفاده می نماییم .



```
> hmset user:1000 username antirez birthyear 1977 verified 1
OK
> hget user:1000 username
"antirez"
> hget user:1000 birthyear
"1977"
```

دیگر توابع مورد استفاده برای hash ها :

HDEL, HEXISTS, HGET, HGETALL, HINCRBY, HINCRBYFLOAT, HKEYS, HLEN, HMGET, HMSET, HSCAN, HSET, HSETNX, HSTRLEN, HVALS

حال با توجه به Data Type ها مختلف و توابع هر کدام می خواهیم اطلاعات مورد نیازمان را از پست های پیام رسان سروش به دست آوریم .

برای اجرای ردیس تنها لازم است تا از سایت <https://redis.io/> ردیس را دانلود نموده و سپس فایل redis-server.exe را اجرا نماییم . برای کار با ردیس هم می توان از redis-cli.exe استفاده نمود و هم از زبان ها مختلفی که از ردیس پشتیبانی می کنند . ما در این پروژه از زبان پایتون استفاده می کنیم . بنابراین لازم است تا کتابخانه redis را نصب کنیم . ردیس روی پورت ۶۳۷۹ localhost اجرا می شود . بنابراین به صورت زیر می توان از طریق پایتون به ردیس متصل شد :

```
r = redis.Redis(host='localhost', port=6379, db=0)
```

سپس به راحتی می توان از تمام توابع معرفی شده در بالا استفاده نمود .

برای دریافت اطلاعات مورد نیاز هر مسئله باید مشخص کنیم که از چه data type استفاده می کنیم و کلید ها را چگونه تعریف کنیم و سایر مواردی که مورد نیاز است که در هر بخش توضیح داده خواهد شد .

برای دریافت اطلاعات از کافکا و ذخیره در ردیس لازم است تا در مرحله اول یک consumer ایجاد کنیم و سپس به صورت real time هر پست جدید را خوانده و اطلاعات مورد نیاز را استخراج نماییم و در ردیس قرار دهیم

```
client = KafkaClient(hosts="127.0.0.1:9092")
topic = client.topics["final"]
consumer = topic.get_simple_consumer()
```

هر پستی که از طریق consumer دریافت می شود را دیکود نموده و با استفاده از ast به json تبدیل می کنیم .

```
msg = (message.value).decode("utf-8")
msg = ast.literal_eval(msg)
```

سپس به سراغ استخراج اطلاعات و قرار دادن آن ها در ردیس می رویم .

نکته ای که وجود دارد این است که با توجه به خواسته مسئله که بیان شده اطلاعات هر کلید تا یک هفته در ردیس باقی بماند و پس از آن به طور خودکار حذف شود ، لازم است تا از تابع expire از ردیس استفاده نماییم . به اینصورت عمل می کند که key و یک مقداری به ثانیه را دریافت نموده و پس از گذشتن آن میزان ثانیه کلید مورد نظر از ردیس حذف می شود . بنابراین ما هر بار که کلید جدیدی را ایجاد می کنیم از تابع expire برای حذف آن بعد از یک هفته استفاده می کنیم .

الف ( تعداد پست های ارسال شده توسط یک کانال در ۶ ساعت گذشته .

برای اینکه بتوانیم به خواسته سوال برسیم لازم است تا کلیدی تعریف کنیم که شامل نام کانال و تاریخ دریافت آن پیام از کانال باشد . برای تاریخ هم تنها به روز و ماه و ساعت دریافت پیام نیاز داریم .

هر بار که پیامی دریافت میشود می توان نام کانال را از فیلد channel\_id مشاهده نمود و به عنوان channel\_key در نظر می گیریم . سپس روز و ماه و ساعت دریافت این پیام را از datetime.now() بدست می آوریم و یک hash ایجاد می کنیم که کلید آن نام کانال به همراه تاریخ دریافت پیام توسط ردیس است و فیلد آن تاریخ دریافت پیام و مقدار آن برابر تعداد باری که در آن تاریخ آن کانال پیام داده است .

```
r.hincrby(channel_key+nowtime,nowtime,1)
r.expire(channel_key+nowtime,14515200)
```

تابع hincrby به این صورت عمل می کند که کلید و فیلد مورد نظر را به همراه یک عدد می گیرد و سپس آن عدد را به مقدار قبلی آن کلید و فیلد اضافه می کند . اگر آن کلید و فیلد در ردیس وجود نداشته

باشد آن ها را ایجاد می کند و مقدار قبلی را صفر در نظر می گیرد . به عنوان مثال اگر از خبرگزاری farsna در روز 15 مرداد ساعت 17 یک پیام دریافت کنیم کلید به صورت farsna051517 و فیلد به صورت 051517 خواهد بود . حال اگر برای اولین بار این پیام از farsna در این زمان دریافت شده باشد این کلید و فیلد را به دیتابیس اضافه نموده و مقدار قبلی را 0 در نظر می گیرد و یکی به آن اضافه می شود . اما اگر از قبل باشد مقدار قبلی را با یک جمع می کند . در خط بعدی از expire استفاده نموده تا پس از گذشت یک هفته (۱۲۵۱۵۲۰۰ ثانیه) آن کلید به صورت خودکار حذف شود .

برای نمایش پاسخ سوال خواسته شده به این صورت عمل می شود که برای نشان دادن تعداد پست های ارسال شده از یک کانال در ۶ ساعت قبل ، ماه و روز و ساعت زمان حال تا ۶ ساعت قبل را محاسبه می نماییم و به همراه نام خبرگزاری به ردیس می دهیم تا تعداد پیام در هر ساعت را به ما بدهد و این مقادیر را با هم جمع می کنیم تا تعداد پیام در ۶ ساعت قبل را بدست آوریم .

```
if(r.hexists(channel_key+last_time,last_time)):
    count += int(r.hget(channel_key+last_time,last_time))
```

همان طور که می بینیم ابتدا با استفاده از hexists چک می کنیم که آن خبرگزاری در آن تاریخ پیامی داشته یا نه . سپس اگر موجود بود مقدارش را با مقادیر سایر ساعات جمع می کنیم .  
Count تعداد پیام های یک خبرگزاری در ۶ ساعت قبل را نشان می دهد .

ب)تعداد کل پیام های دریافت شده در یک بازه زمانی مثلا روز گذشته

در این سوال تعداد پیام های دریافت شده در ۲۴ ساعت گذشته را نشان می دهیم . بدین منظور از hash ها استفاده می کنیم . کلید را همان طور که در شکل زیر مشاهده می کنید به صورت count به همراه روز و ماه و ساعت دریافت پیام در نظر می گیریم و فیلد مورد نظر را همان روز و ماه و ساعت . اگر آن کلید در ردیس موجود باشد با دیدن یک پیام در آن ساعت فقط یک مقدار به مقدار قبلی اضافه میشود در غیر این صورت آن کلید و فیلد به ردیس اضافه شده و مقدار یک می گیرد . با استفاده از expire نیز کلید مربوطه پس از یک هفته به طور خودکار حذف میشود .

```
r.hincrby('count'+nowtime,nowtime,1)
r.expire(channel_key+nowtime,14515200)
```

برای نشان دادن تعداد پیام ها در ۲۴ ساعت گذشته روز و ماه و ساعت را تا ۲۴ ساعت قبل به دست می آوریم و با کلمه count ترکیب نموده و کلید و فیلد مورد نظر را ابتدا با hexists چک می کنیم که موجود باشد سپس با استفاده از hget به ردیس می دهیم و تعداد پست در هر ساعت را به دست آورده و در نهایت این مقادیر را جمع می کنیم .

```
if(r.hexists('count'+last_time,last_time)):
    count += int(r.hget('count'+last_time,last_time))
```

پاسخ نهایی در count خواهد بود .

ج ) تعداد هشتگ های دریافت شده در یک ساعت گذشته ( به صورت منحصر به فرد)

برای به دست آوردن پاسخ این سوال پس از دریافت هر پیام ، فیلد hashtags را گرفته و هشتگ های موجود در آن را به ردیس اضافه می کنیم . برای این قسمت از hashها استفاده می کنیم . با توجه به شکل زیر کلید را برابر hashtags به همراه ماه و روز و ساعت قرار داده و فیلد ها را هم به صورت هشتگ موجود در پیام به همراه روز و ماه و ساعت و دقیقه قرار می دهیم . دلیل اینکه فیلد را به دقیقه وابسته می کنیم آن است که می خواهیم تعداد هر هشتگ در ۶۰ دقیقه گذشته را بدست آوریم .

```
r.hincrby('hashtags'+nowtime,hashtag+nowtimemin,1)
r.expire('hashtags'+nowtime,14515200)
```

زمانی که یک پست دریافت میشود به ازای هر هشتگ موجود در متن یک فیلد به کلید hashtags + nowtime اضافه می شود که این فیلد وابسته به روز و ماه و ساعت و دقیقه است .

برای پیدا کردن تعداد تکرار هر هشتگ در ۶۰ دقیقه گذشته ابتدا باید ببینیم که در ۶۰ دقیقه قبل چه هشتگ هایی دریافت شده است بنابراین می توان به صورت زیر فیلدهای مربوط به کلید ساعت حالا و یک ساعت قبل را به دست آورد .

```
hashtags = [i.decode("utf-8")[:-8] for i in r.hgetall('hashtags'+nowtime)] +
            [i.decode("utf-8")[:-8] for i in r.hgetall('hashtags'+last_time)]
```

بنابراین تا اینجا به دست آوردیم که در یک الی دو ساعت قبل چه هشتگ هایی داشتیم و پس از آن برای هر کدام از این هشتگ ها می خواهیم ببینیم که آیا در ۶۰ دقیقه قبل تکرار شده یا خیر؟ و اگر تکرار شده است چه تعداد بار تکرار شده است. بنابراین به صورت زیر عمل می کنیم

```
if(r.exists('hashtags'+last_time,hashtag+last_time_min) ):
    c += int(r.hget('hashtags'+last_time,hashtag+last_time_min))
if(r.exists('hashtags'+ nowtime,hashtag+last_time_min) ):
    c += int(r.hget('hashtags'+nowtime,hashtag+last_time_min))
```

برای بررسی حضور یک هشتگ در ۶۰ دقیقه قبل لازم است تا فیلد را شامل هشتگ مورد نظر به همراه روز و ماه و ساعت و دقیقه زمان حال و ۶۰ دقیقه قبل در نظر بگیریم و مقدار آن را که برابر تعداد تکرار آن هشتگ در آن دقیقه است را به دست آورده و همه را با هم جمع کنیم .

مقادیر به دست آمده برای هر هشتگ را در یک set ذخیره می کنیم .

(د) آخرین هشتگ های دریافت شده .

در این قسمت از لیست ها استفاده می کنیم به این صورت که هر پستی که به دست می آوریم ، هشتگ های موجود در آن را به ابتدا یک لیست با نام hashtagList با استفاده از تابع LPUSH اضافه می کنیم و پس از آن با استفاده از تابع LTRIM فقط عضوهای 0 تا 999 را نگه می داریم و بقیه حذف می شوند .

```
r.lpush('hashtagList',hashtag)
r.ltrim('hashtagList',0,999)
```

برای به دست آوردن ۱۰۰۰ هشتگ آخر دریافت شده نیز می توان از تابع LRANGE استفاده نمود :

```
r.lrange('hashtagList',0,999):
```

(ه) آخرین پس های دریافت شده

در این قسمت هم از لیست ها استفاده می کنیم به این صورت که هر بار پیامی دریافت شد متن موجود در فیلد text را در ابتدای لیست postList قرار می دهیم و سپس با استفاده از LTRIM تنها ۱۰۰ داده ابتدای لیست را نگه می داریم و بقیه را حذف می کنیم .

```
r.lpush('postList',msg['text'])
r.ltrim('postList',0,99)
```

برای نشان دادن ۱۰۰ پیام آخر نیز از تابع LRANGE استفاده می کنیم :

```
r.lrange('postList',0,99):
```

برای هر کدام از نمونه های بالا یک تابع ایجاد شده است . همه این توابع را در یک تابع به نام `result` فراخوانی می کنیم و پاسخ را به صورت `string` باز می گردانیم .

سپس با استفاده از یک app فلسک آن ها را نمایش می دهیم که به صورت زیر نوشته شده است :

```
app = Flask(__name__)

@app.route('/')
def resflask():
    return result()

if __name__ == '__main__':
    app.run()
```

نتیجه به صورت زیر است :

number of post from channel tasnimna in 6 hour ago is 0

total number of posts in 24 hours ago is 42

```
count of each hashtags in 60 min ago : {}
```

```
last 1000 hashtags :
```

[illegible]

last 100 post is :

0) @Farsna خسارات ناشی از موج شدید انفجار امروز بیروت

ن در جنگ رسانه‌های علیه ایران و واشنگتن تلویزیون: ◆ نهاد دولت برای نظارت بر رسانه‌هایی از جمله صدای آمریکا، رادیو آواز آسیا و صدای فردا، گزارشات داده صدور مجوز برای اتباع کشورهای مانند چین و ایران جهت کار مستقر در آمریکا، امنیت ملی آمریکا را به خطر انداخته است. ◆ اخیراً دولت آمریکا انتقادهای زیادی از این رسانه‌ها به عمل آورده و تلاش‌های متعددی از کسر بودجه گرفته تا تغییر مدیریت آن را در دستور کار قرار داده است.

fna.ir | ewyb9r @Farsna

2) \*مادری میثدال دارای 2 فرزند حاصل از ساکنین قسمت های جنوبی کشور که به دلیل چشم درد های مکرر به پزشک مراجعه می کند و متوجه می شود که به سرطان چشم مبتلا هستند و همسر بیمارین که کارگر \* و \* و صرف هزینه درمان چشم ایشان کرده است، طبق نظر پزشک باید 5 دوره شیمی درمانی و سپس جراحی شوند ولی دیگر توانایی تامین هزینه های آزمایشات و شیمی درمانی و جراحی را ندارند. این خانواده متشکل هستند و نای عقب افتاده رفته و دیگری پولی ندارند. (نظر داریار با پای خانواد متعال مبلغ 2 میلیون تومان برای کمک به تامین هزینه های عمل جراحی و سایر هزینه های مربوط به ایشان قرار داریم. <https://b2n.ir/000636>؛ پرداخت این مبلغ: <https://hakeim.ir/cash/5022224000303667>؛ \*274\*+296)

ن در جنگ رسانه‌های علیه ایران و واشنگتن تألیف: **◆ نهاد دولت برای نظارت بر رسانه‌های از جمله صدای آمریکا، رادیو آزاد آسیا و صدای فردا، گزارش‌های دستور مجوز برای اتباع کشورهای ثالث چین و ایران جهت کار 3) بیشتر در آمریکا، امنیت ملی آمریکا را به خطر انداخته است. ◆** اخیراً دولت آمریکا انتقادهای زیادی از این رسانه‌ها به عمل آورده و تلاش‌های متعددی از کسر بودجه گرفته تا تغییر مدیریت آن را در دستور کار قرار داده است. [fvwv9b@Farsna](mailto:fvwv9b@Farsna)

4)  **مادری میانساز** دارای 2 فرزند محصل از ساکنین قسمت های جنوبی کشور که به دلیل جرم درد های مکرر به پزشکی مراجعه می کند و متوجه می شود که به سرطان چشم مبتلا هستند و همسر بیمارشان که کارگران  **مادری میانساز** دارای 2 فرزند محصل از ساکنین قسمت های جنوبی کشور که به دلیل جرم درد های مکرر به پزشکی مراجعه می کند و متوجه می شود که به سرطان چشم مبتلا هستند و همسر بیمارشان که کارگران

که با هر بار رفرش نمودن صفحه توابع فراخوانی شده و داده ها به صورت real time نمایش داده می شود .



در این گام از دیتابیس clickhouse برای ذخیره اطلاعات به صورت ستونی استفاده کرده ایم. زیرا این الگوریتم با ذخیره داده ها به صورت ستونی، واکنشی سریع تری دارند و می توان بدون انجام عمل join کوئری ها را پاسخ دهیم. در این سوال یک جدول ایجاد کرده ایم و اطلاعات ذخیره شده برای هر پیام را از کافکا خوانده و وارد این جدول کرده ایم: [6]

جدول ساخته شده، سه مورد کلیدی دارد:

- نام جدول
- لیستی از ستون ها و نوع آن ها
- Table engine که مشخص می کند داده ها کجا ذخیره شوند ( در کجا نوشته شوند و از کجا خوانده شوند، کدام نوع کوئری ها فراهم شوند و چگونه؛ دسترسی به داده های اخیر؛ استفاده از ایندکس در صورت وجود؛ انواع table engines

- MergeTree
- ReplacingMergeTree
- SummingMergeTree
- AggregatingMergeTree
- CollapsingMergeTree
- VersionedCollapsingMergeTree
- GraphiteMergeTree

سری های mergeTree ها، برای وارد کردن حجم بسیاری از داده ها در جدول ها کاربرد دارد. داده ها، به سرعت و به صورت قسمت قسمتی وارد جدول ها نوشته می شوند. ویژگی های اصلی آن :

- Stores data sorted by primary key.
- Partitions can be used if the partitioning key is specified.
- Data replication support.
- Data sampling support.



نحوه ساخت جدول:

```
from clickhouse_driver import Client
client = Client(host='localhost')

client.execute('CREATE DATABASE IF NOT EXISTS clickhouse')
client.execute("""CREATE TABLE clickhouse.all_posts(
    channel_id String,eventdate date
    ,eventtime DateTime,message_id String ,hashtags Array(String)
    ,metadata Array(String)
)
ENGINE = MergeTree
PARTITION BY eventdate ORDER BY(eventdate,channel_id)
""")
```

می توان برای تحلیل هشتگ های به کار رفته در پیام ها، جدول جداگانه ای برایشان ایجاد کنیم:

```
from clickhouse_driver import Client
client = Client(host='localhost')
client.execute("""CREATE TABLE clickhouse.hashtags(
    hashtag String,
    channel_id String,eventdate date
    ,eventtime DateTime,message_id String
)
ENGINE = MergeTree
PARTITION BY hashtag ORDER BY(hashtag,eventdate,channel_id)
""")
```

کد این قسمت در فایل step5/create\_table.py ذخیره شده است.

```
--
In [10]: 1 client.execute('show tables from clickhouse')
Out[10]: [('all_posts',)]
```

(لازم به ذکر است که استفاده از کلیک هوس در ویندوز از طریق دستوراتی که در پیوست در قسمت کار با wsl گفته شده است، امکان پذیر شده است).

نحوه خواندن داده ها از کافکا و ذخیره آن ها در جدول:

```
#clickhouse
from clickhouse_driver import Client
client = Client(host='localhost')
```

```

from datetime import datetime
#print("message_id" , "timestampISO" , "channel_id", "hashtags")
for message in consumer:
    if message is not None:
        #print(message.offset, (message.value).decode("utf-8"))
        msg = (message.value).decode("utf-8")
        channel_id = ''
        message_id = ""
        timestampISO = ""
        time_ = ""
        date = ""
        metadata = ""
        hashtags = ""
        date_time_obj = datetime.now()
        if msg.find('message_id')!= -1:
            msg = msg[msg.find('message_id')+14:]
            message_id = msg[:msg.find(',')-1]
        if msg.find('timestampISO')!= -1:
            msg = msg[msg.find('timestampISO')+16:]
            timestampISO = msg[:msg.find(',')-1]
            date = msg[2:msg.find('T')]
            time_ = msg[msg.find('T')+1:msg.find('+')]
            date_time_obj = datetime.strptime(date+" "+time_, '%y-%m-%d %H:%M:%S')
        if msg.find('channel_id')!= -1:
            msg = msg[msg.find('channel_id')+14:]
            channel_id = msg[:msg.find(',')-1]
        if msg.find('metadata')!= -1:
            msg = msg[msg.find('metadata')+12:]
            metadata = msg[:msg.find('],')]
        if msg.find('hashtags')!= -1:
            msg = msg[msg.find('hashtags')+12:]
            hashtags = msg[:msg.find('],')]
        insert1 = """INSERT INTO clickhouse.all_posts(
            channel_id,eventtime,message_id ,
            hashtags
            ,metadata
        ) VALUES"""
        x = hashtags.split(", ")
        y = metadata.split(", ")
        client.execute(insert1,[{'channel_id':channel_id,'eventtime':date_time_obj,'message_id':message_id,'metadata':y,'hashtags':x}])

```

کد این قسمت در فایل `step5/consumer.py` ذخیره شده است.

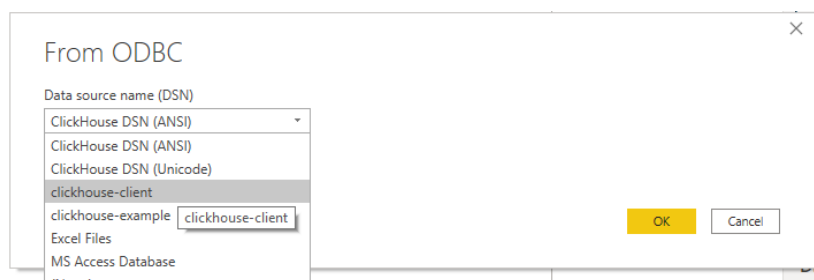
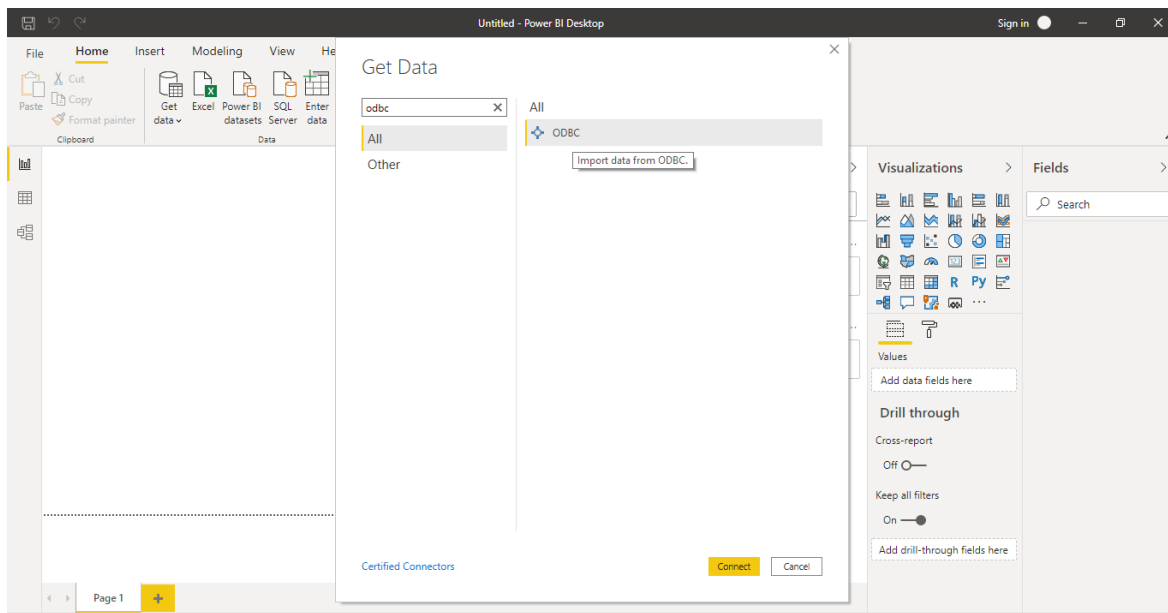
نمونه ای از اطلاعات ذخیره شده :

```
In [3]: 1 client.execute('select * from clickhouse.all_posts')
```

```
Out[3]: [('bigproj',  
          datetime.datetime(2020, 8, 6, 10, 58, 2),  
          '15967114784041696ag1ay',  
          ['',  
            ["'ايندېنځست'", "'تېکنین'", "'نیټوی'", "'این'"],  
            ('tasnimna',  
              datetime.datetime(2020, 8, 7, 19, 53, 5),  
              '1596829759540d36b2w24j',  
              ['',  
                ["'فهرمائی'",  
                  '"ایران"',  
                  '"حسینی"',  
                  "'TasnimNa'",  
                  '"لحه"',  
                  '"بردن"',  
                  '"آیگ"',  
                  '"نوسط"',  
                  '"پالو"',  
                  '"جام"' ]]),  
            ('farsna',  
              datetime.datetime(2020, 8, 7, 19, 27, 49),  
              '1596828467244179f9zi183',  
              ['',  
                ["'بیروت'",  
                  '"\u200cهای پروکل'",  
                  '"ایران"',  
                  '"کتکور"',  
                  "..."]])
```

## ساخت داشبوردهای مدیریتی

با انجام عملیات زیر، پایگاه داده ای که ایجاد کرده بودیم، با نام clickhouse را به پاور بی آی متصل کردیم:



با وارد کردن default : user به کلیک هوس متصل شده است.

نمونه ای از نمایش جدول ایجاد شده در کلیک هوس با استفاده از Power Bi:

The screenshot shows the ClickHouse Navigator application. On the left, a tree view under 'clickhouse [1]' shows the 'all\_posts' table selected. The right pane displays the 'all\_posts' table with the following data:

channel_id	eventtime	hashtags	message_id	meta
bigproj	8/6/2020 10:58:02 AM	[""]	15967114784041696ag1ay	این
tasnimna	8/7/2020 7:53:05 PM	[""]	1596829759540d36b2w24J	بی
farsna	8/7/2020 7:27:49 PM	[""]	1596828467244179f9zi83	کل
iribnews	8/7/2020 6:49:26 PM	[""]	15968261643749466e4hQg	جی

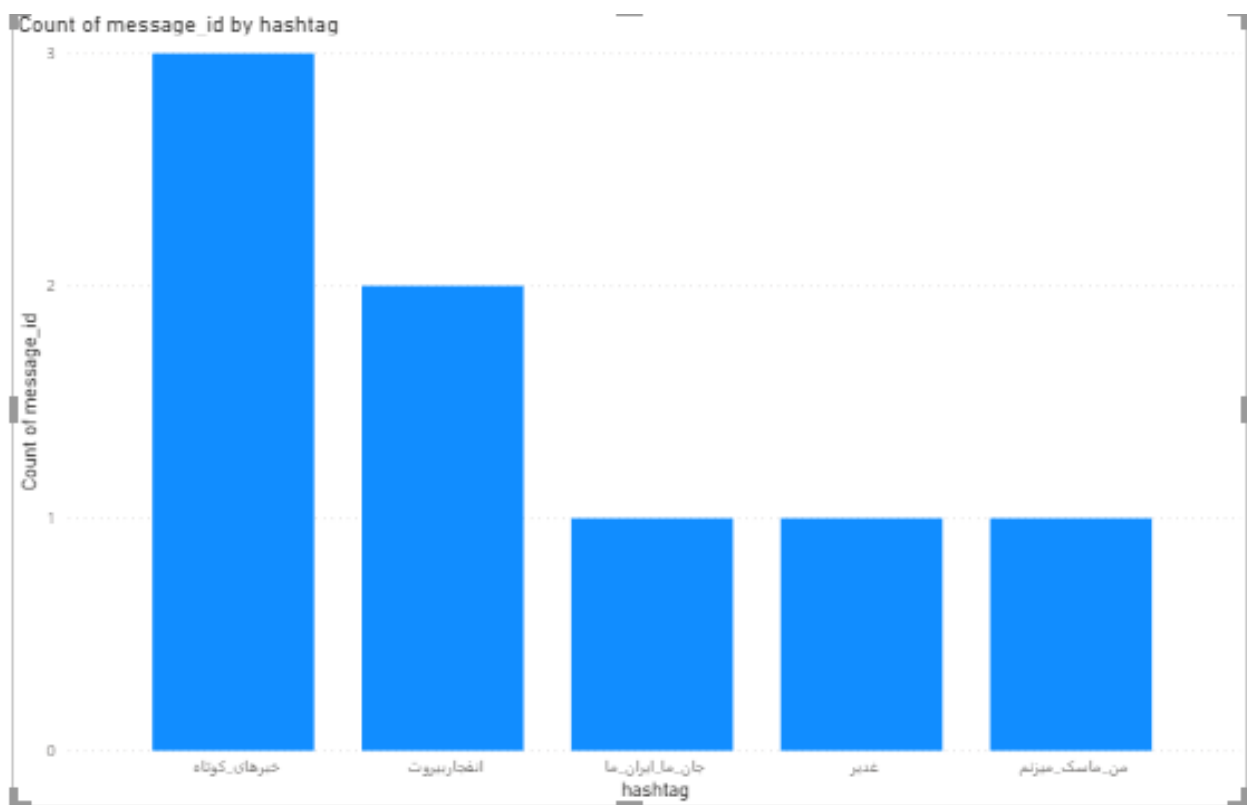
( لازم به ذکر است که برای اتصال به کلیک هوس در تمام مراحل باید آن را به دستور `sudo service clickhouse-server start` در ابونتوی wsl اجرا کرده باشیم)

به علت پایین بودن میزان رم کامپیوتر شخصی، توانایی استفاده از داده های بسیار زیاد وجود نداشت، به همین دلیل از تعداد کمی از کانال ها و پیام هایشان استفاده کرده ایم.

حال می توان با استفاده از اعمال دستورات مختلف، نمودار های مورد نظر را ایجاد کرد:

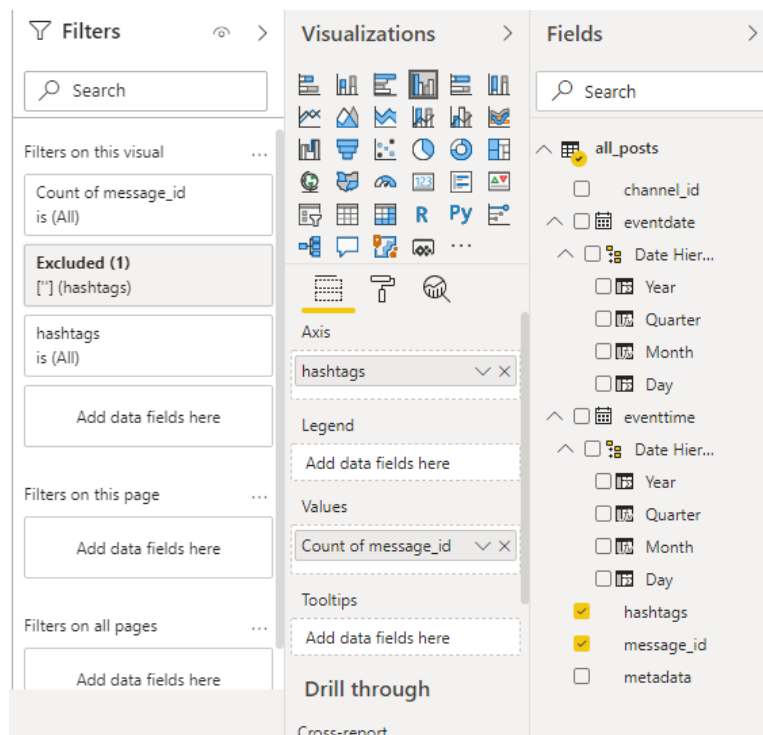
*گزارش مرتبط با هشتک ها:*

- تعداد هر کدام از هشتک های به کار رفته در کل پیام ها :



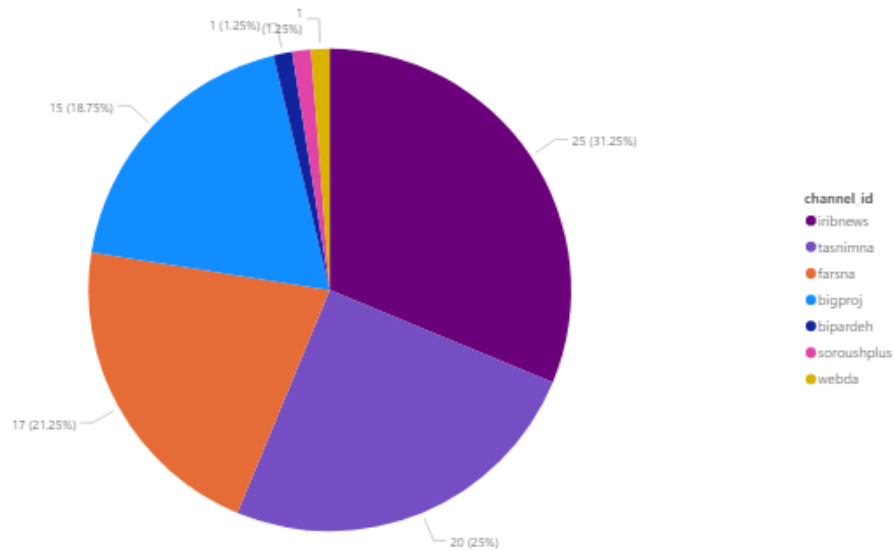
همان طور که دیده می شود، در میان هشتگ های دریافت شده از پیام های کانال ها، #خبرهای\_کوتاه بیشترین تعداد بار در پیام ها تکرار شده است.

تنظیمات :

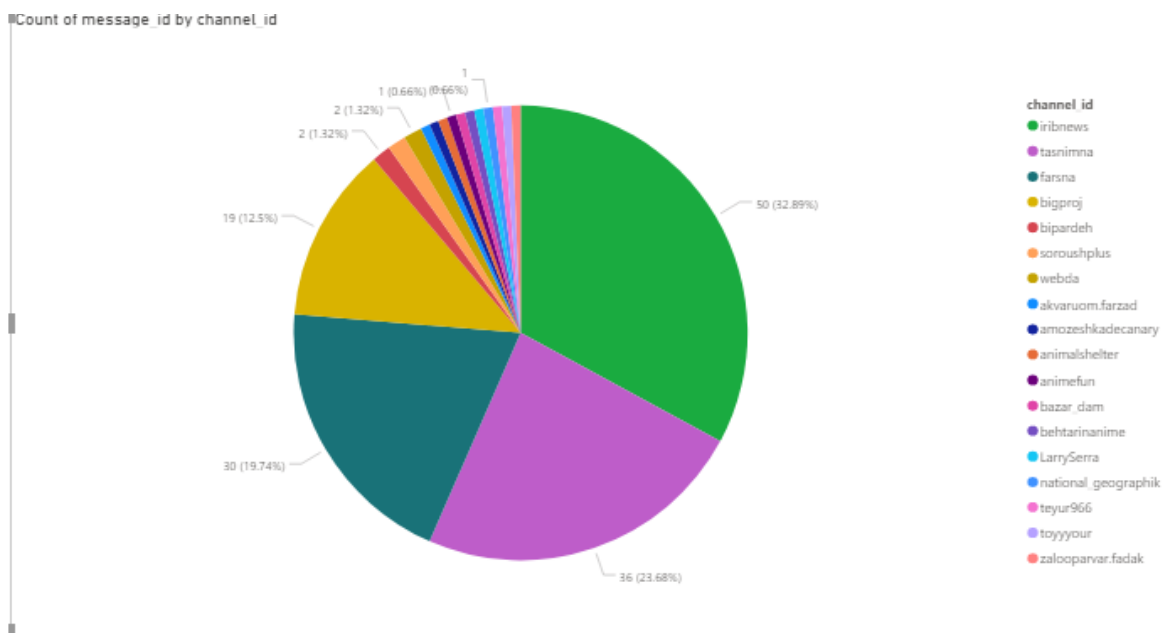


### گزارش مربوط به کانال ها:

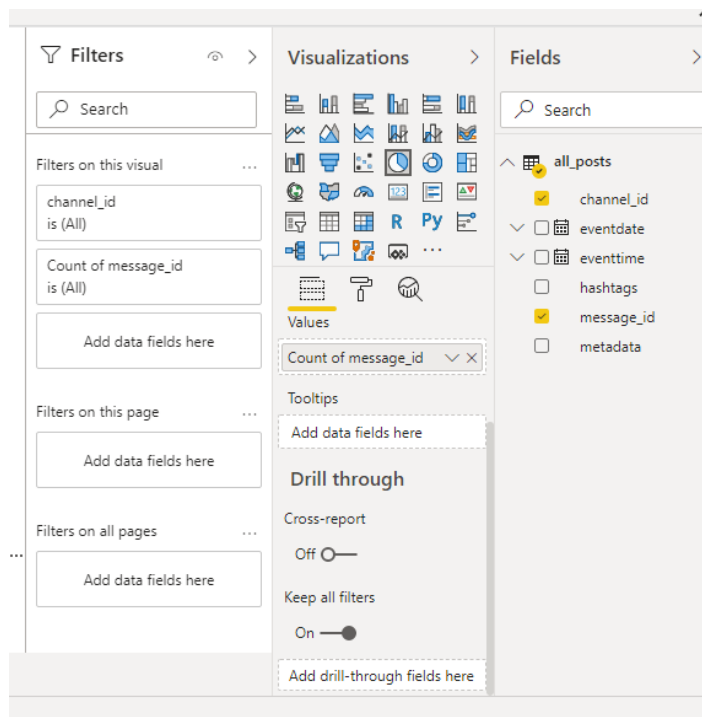
- تعداد پیام هایی که در هر کانال قرار داده شده است:



در نمودار بعدی با داده های به روز شده و از تعداد کانال های بیشتری استفاده شده است.

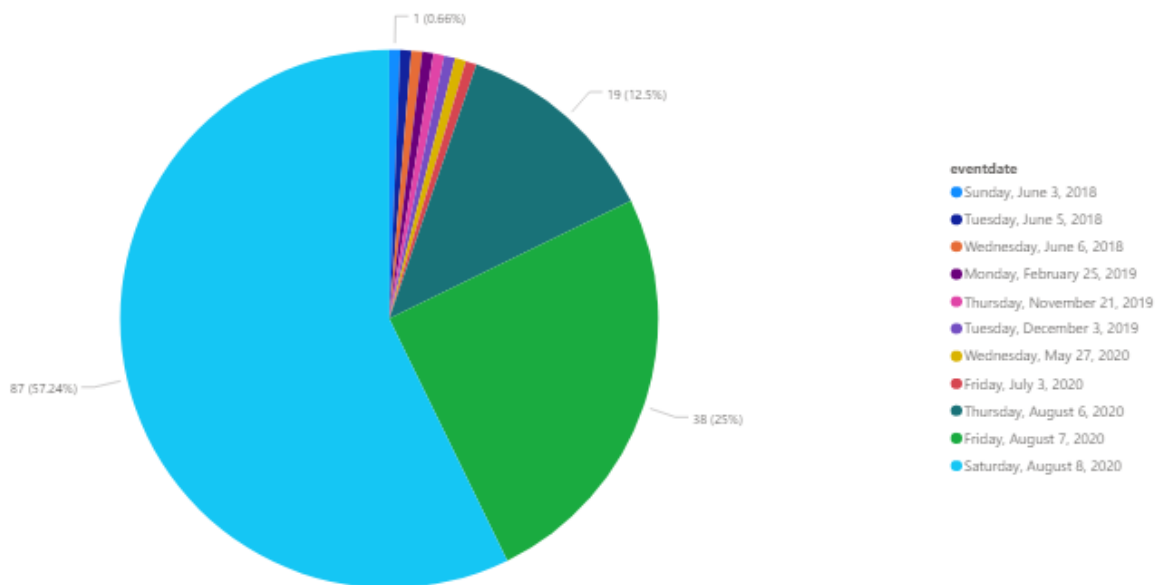


تنظیمات:



- در هر روز، هر کانال چه تعداد پیام گذاشته اند:

Count of message\_id by eventdate

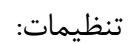


تنظیمات:



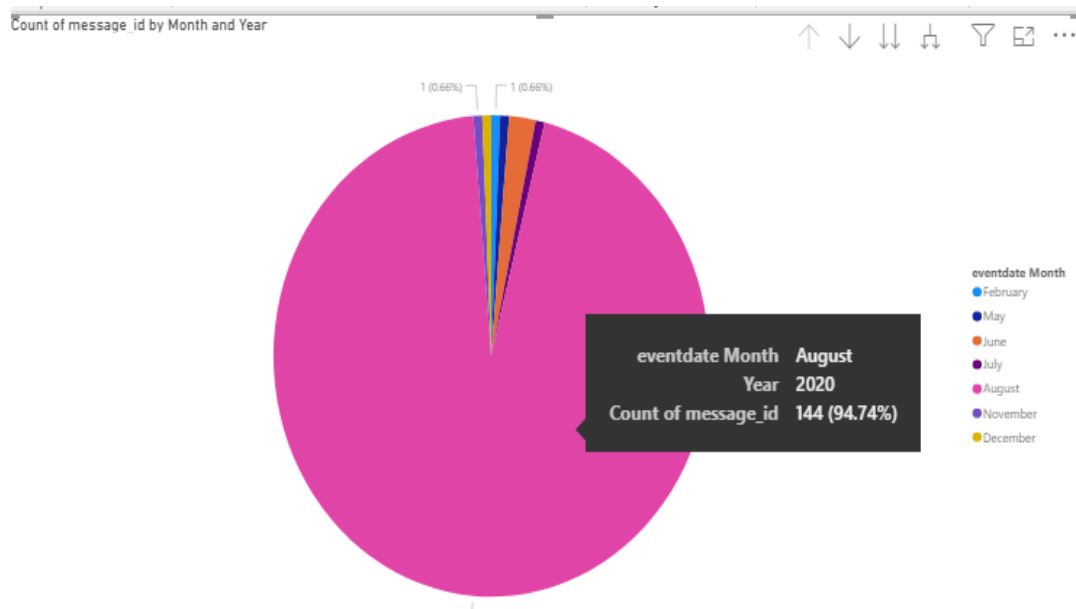


- تعداد همه ی پیام ها به ازای هر روز:

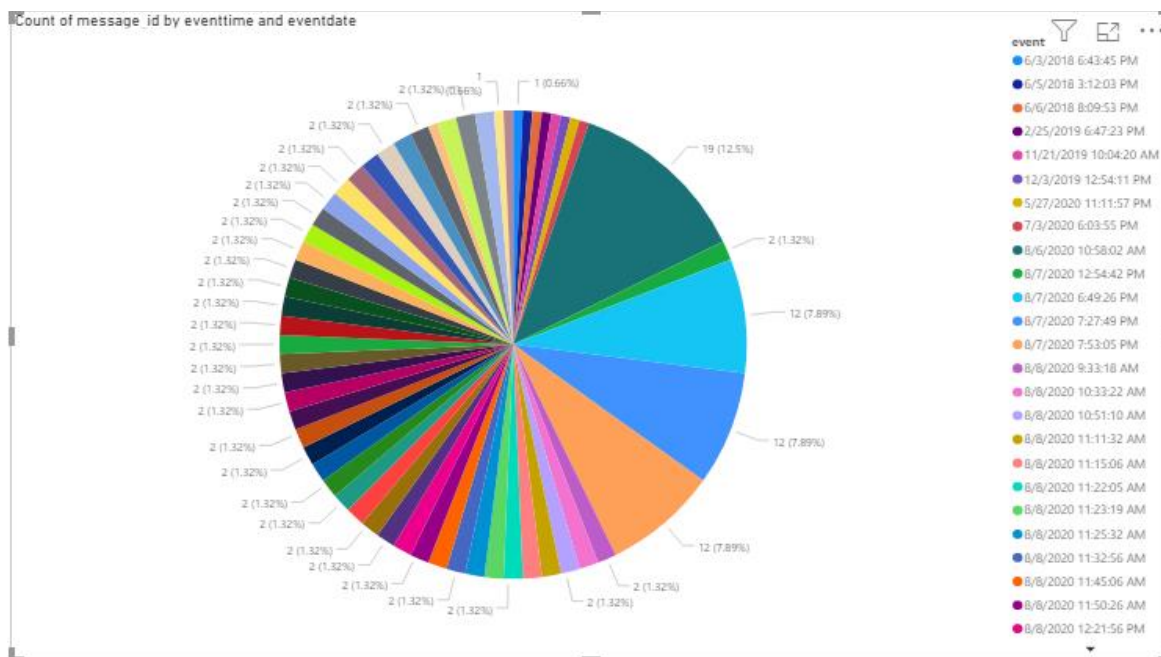


۴۲

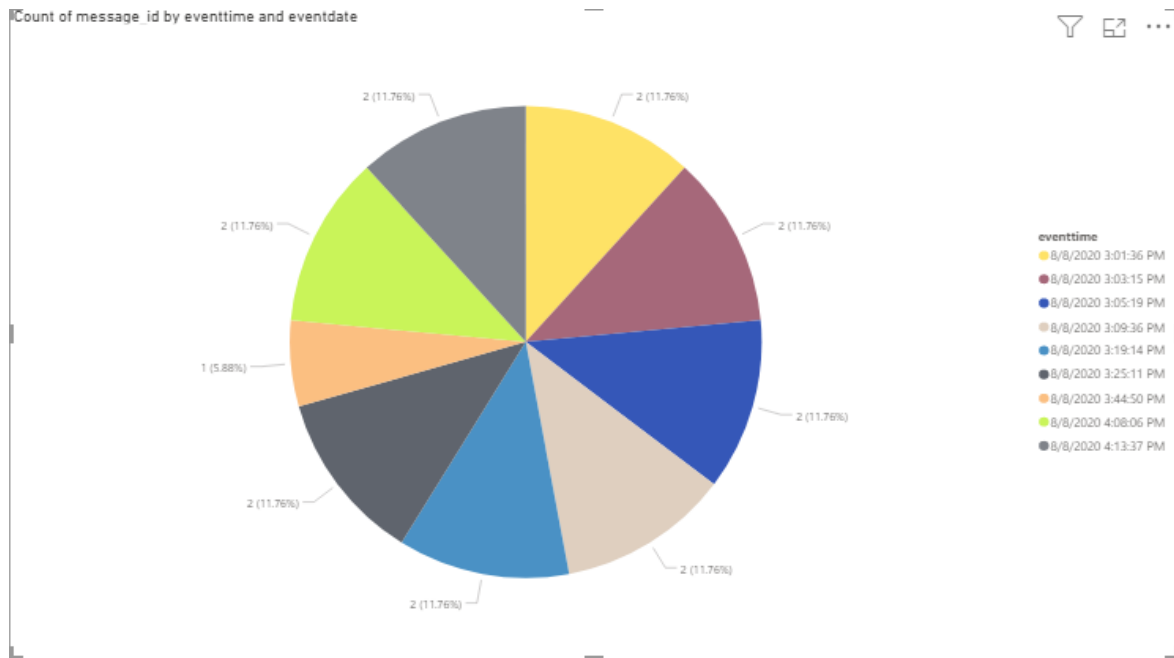
- تعداد همه ی پیام ها در هر ماه:



- تعداد پیام‌ها در هر روز و هر ساعت:



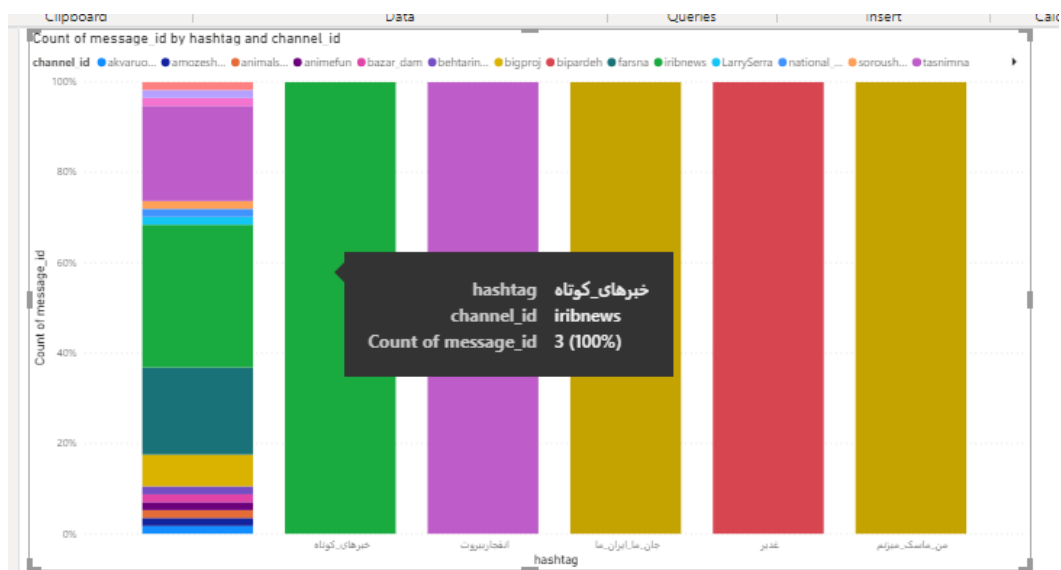
- تعداد پیام ها در یک ساعت گذشته ( تاریخ ۲۰۲۰/۸/۸ بین ساعت ۳ تا ۴):

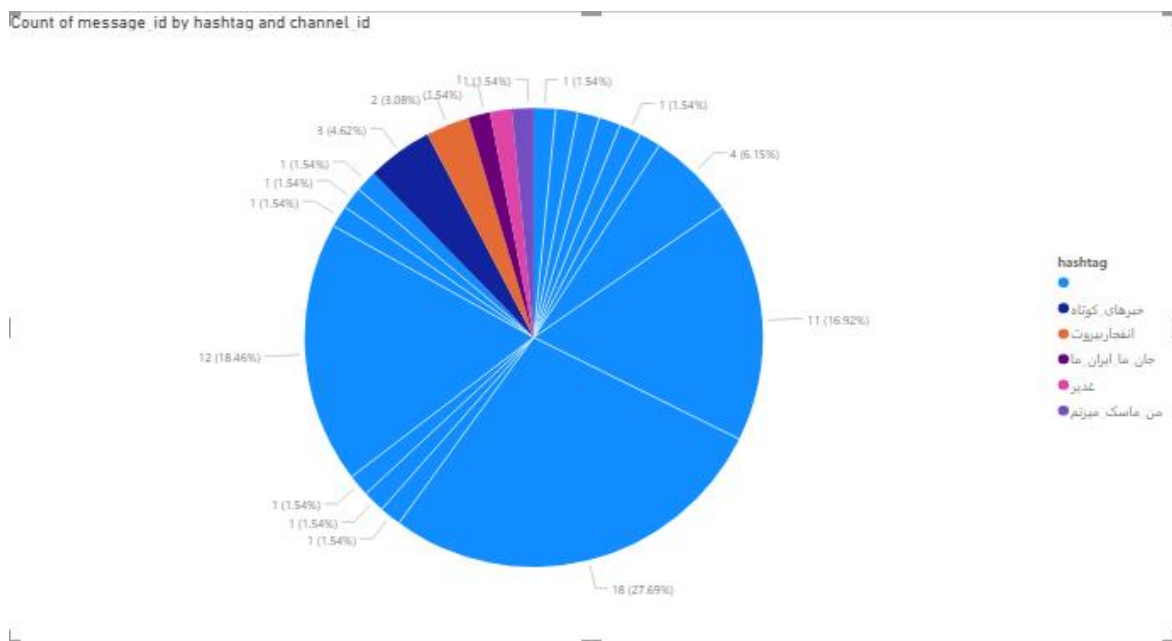


### گزارش های مرتبط با کانال و هشتگ:

- هر کانال چه تعداد از هشتگ ها را در پیام های خود داشته اند:

دو نمودار زیر نشان می دهند که هر کدام از هشتگ ها در کدام یک از کانال ها آمده است و چه تعداد تکرار شده است.





تنظیمات:

Filters

Search

Filters on this visual

channel\_id is (All)

Count of message\_id is (All)

Excluded (1)

[""] (hashtags)

hashtags is (All)

Add data fields here

Filters on this page

Add data fields here

Visualizations

Axis

hashtags

Legend

channel\_id

Values

Count of message\_id

Tooltips

Add data fields here

Drill through

Cross-report

Fields

Search

all\_posts

channel\_id

eventdate

eventtime

hashtags

message\_id

metadata

hashtags

channel\_id

eventdate

eventtime

hashtag

message\_id

## ساخت یک مدل پیش بینی کننده با اسپارک – دارای امتیاز

ابتدا لازم است تا کاساندرا و pyspark را نصب نموده و سپس آن ها را به هم وصل نماییم . برای نصب کاساندرا در ویندوز ، کاساندرا نسخه 11.3.7 را دانلود نموده و فایل Cassandra.bat را در فولدر bin اجرا می کنیم . سپس pyspark را با استفاده از دستور زیر اجرا می کنیم :

```
!python -m pip install pyspark==2.4.6|
```

سپس برای اتصال کاساندرا و pyspark فایل spark-cassandra-connector-2.4.0-s\_2.11.jar را دانلود می کنیم و از طریق sparkConf ، sparkContext را به کاساندرا وصل می نماییم:

```
conf = SparkConf()\
    .setAppName("Python Script for getting the table 'favorite foods')\
    .setMaster("local[*]") \
    .set('spark.jars', 'spark-cassandra-connector-2.4.0-s_2.11.jar')
```

```
sc = SparkContext(conf=conf)
```

سپس یک sqlContext ایجاد می کنیم و داده ها را از کاساندرا می خوانیم :

```
sqlContext = SQLContext(sc)
```

### • پیش بینی زمان ارسال پست بعدی

برای این مدل لازم است تا داده های مربوط به جدول all\_channels را از keyspace با نام final4 بخوانیم :

```
df = sqlContext.read.format("org.apache.spark.sql.cassandra")\
    .options(table="all_channels", keyspace="final4").load()
```

حال در این قسمت فیلدهای مربوط به کانال خبرگزاری مورد نظر را جدا نموده و سپس براساس date آن ها را groupby می کنیم و سپس می شماریم که هر date چند بار تکرار شده است . در واقع می خواهیم ببینیم در یک date مشخص چه تعداد پیام از آن کانال دریافت شده است .

```
from pyspark.sql import functions as F
train_data = df.where(F.col('channel_id') == 'iribnews').\
    groupby(F.col('date').alias('ds')).agg(F.count(F.lit(1)).alias("y")).sort('ds')
```

برای پیش بینی این مدل لازم است تا از مدل های time series استفاده نماییم . یک مدل به نام prophet وجود دارد که time series عمل می کند و به عنوان ورودی یک دیتافریم شامل دو ستون

دریافت می کند . یک ستون ds نام دارد که درواقع date time ها را نشان می دهد و می تواند براساس زمان های مختلف یعنی سال و ماه و روز و ساعت و دقیقه دریافت کند و براساس زمان های مختلف پیش بینی نماید . فرمت داده های date time وردی به صورت YYYY-MM-DD HH:MM:SS است که به ما داده ها را به این فرمت ذخیره نموده ایم و چون می خواهیم براساس دقیقه پیش بینی نماییم ثانیه ها را صفر در نظر می گیریم . ستون دیگر y نام دارد که در واقع مقداری است که می خواهیم پیش بینی کنیم که ما در این سوال مقدار آن را تعداد پست ها در آن date time در نظر میگیریم .

از آنجایی که دیتافریم ما شامل فقط date time هایی است که اتفاق افتاده و مقادیر y بزرگتر مساوی یک دارند . لازم است تا date time هایی که پستی در آن ها ارسال نشده یعنی  $y=0$  است را به صورت رندوم (چون دقایقی که پیامی ارسال نشده بسار بیشتر از دقایقی است که ارسال شده و باید دیتاست balance داشته باشیم ) به دیتافریم اضافه کنیم تا مدل بتواند مقادیر را به درستی تشخیص دهد . prophet یک مدل با دقت بالا است و می تواند با دقت بالا برای ما پیش بینی نماید .

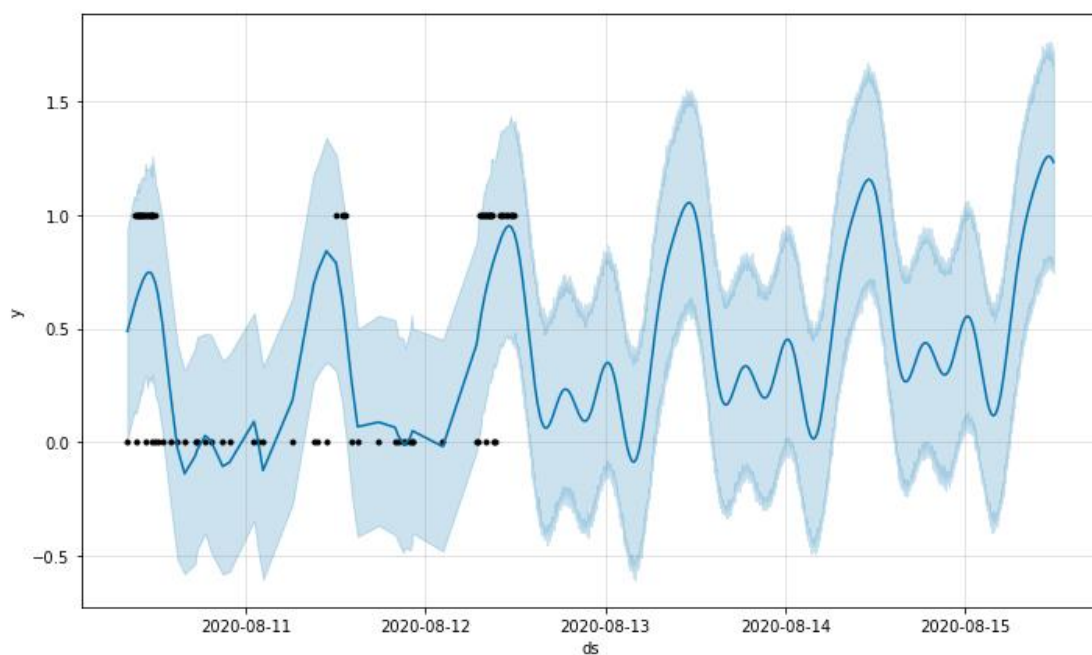
حال لازم است تا مدل را ایجاد نموده و دیتا فریم را به آن بدهیم .

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

سپس با استفاده از make\_future\_dataframe می توانیم بازه زمانی را که می خواهیم پیش بینی کنیم را به مدل بدهیم و با استفاده از predict آن را پیش بینی کنیم . زمانی که ما در نظر گرفتیم سه روز آینده است .

```
future = m.make_future_dataframe(periods=3*24*60, freq='min', include_history =False)
fcst = m.predict(future)
```

سپس می توانیم مقادیر پیش بینی شده را رسم کنیم . این نمودار شامل date time های گذشته و مقادیر پیش بین شده برای آینده و هم چنین مقدار upper bound و lower bound را نیز شامل می شود .



نقاطی که با مشکی مشخص شده اند نقاط گذشته را نشان می دهند . برای اینکه نشان دهیم چند دقیقه آینده پیام دریافت می شود می توانیم یک threshold در نظر بگیریم و سپس آن با توجه به مقادیر پیش بینی شده آن دقیقه را حساب کنیم .



## پیوست – کار با ابزارها

### کار با Kafka

ابتدا از سایت <https://kafka.apache.org> آخرین نسخه ی آن را دانلود و unzip کرده و در داخل فولدر آن دستورات زیر را به ترتیب در cmd ویندوز اجرا کرده ایم ( برای اجرای دستورات زیر در لینوکس ، کافی است به جای bin\windows\bin فقط bin\ را قرار دهیم و همین فایل های bat. را تبدیل به فایل های sh. کنیم):

اجرای [7] zookeeper:

```
bin\windows\zookeeper-server-start.bat config/zookeeper.properties
```

```
[2020-07-28 14:22:48,557] INFO Using checkIntervalMs=60000 maxPerMinute=10000 (org.apache.zookeeper.server.ContainerManager)
[2020-07-28 14:23:18,818] INFO Creating new log file: log.1 (org.apache.zookeeper.server.persistence.FileTxnLog)
```

اجرای سرور کافکا:

```
bin\windows\kafka-server-start.bat config/server.properties
```

```
[2020-07-28 14:23:23,421] INFO Kafka version: 2.5.0 (org.apache.kafka.common.utils.AppInfoParser)
[2020-07-28 14:23:23,434] INFO Kafka commitId: 66563e712b0b9f84 (org.apache.kafka.common.utils.AppInfoParser)
[2020-07-28 14:23:23,436] INFO Kafka startTimeMs: 1595930003364 (org.apache.kafka.common.utils.AppInfoParser)
[2020-07-28 14:23:23,459] INFO [KafkaServer id=0] started (kafka.server.KafkaServer)
```

ایجاد topic:

```
bin\windows\kafka-topics.bat --create --bootstrap-server
localhost:9092 --replication-factor 1 --partitions 1 --topic topic_name
```

ارسال پیام به صورت دستی به کافکا: dt:

```
bin\windows\kafka-console-producer.bat --bootstrap-server localhost:9092
--topic topic_name
```

```
C:\kafka_2.12-2.5.0>bin\windows\kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic test
Created topic test.

C:\kafka_2.12-2.5.0> bin\windows\kafka-topics.sh --list --bootstrap-server localhost:9092
'bin\windows\kafka-topics.sh' is not recognized as an internal or external command,
operable program or batch file.

C:\kafka_2.12-2.5.0> bin\windows\kafka-topics.bat --list --bootstrap-server localhost:9092
test

C:\kafka_2.12-2.5.0>bin\kafka-console-producer.bat --bootstrap-server localhost:9092 --topic test
'bin\kafka-console-producer.bat' is not recognized as an internal or external command,
operable program or batch file.

C:\kafka_2.12-2.5.0>bin\windows\kafka-console-producer.bat --bootstrap-server localhost:9092 --topic test
'bin' is not recognized as an internal or external command,
operable program or batch file.

C:\kafka_2.12-2.5.0>bin\windows\kafka-console-producer.bat --bootstrap-server localhost:9092 --topic test
>hello test!
>
```

دریافت پیام از کافکا به وسیله consumer:

```
bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092
--topic test --from-beginning
```

```
C:\Users\Admin>cd "C:\kafka_2.12-2.5.0"
C:\kafka_2.12-2.5.0>bin\windows\kafka-console-consumer.bat --bootstrap-server localhost:9092 --topic test --from-beginning
hello test!
```

می توان کافکا را بر روی پایتون نصب کرد و دستورات آن را در قالب کدهای پایتون اجرا کرد.[8]

نصب کتابخانه pykafka:

```
pip install pykafka
```

چند نمونه از دستورات لازم کافکا در پایتون (کتابخانه pykafka)

اتصال به kafka:

```
client = KafkaClient(hosts="127.0.0.1:9092")
```

استفاده از topic:

```
topic = client.topics["topuc_name"]
```

ذخیره پیام در کافکا:

```
with topic.get_sync_producer() as producer :
```

```
producer.produce('message')
```

استفاده از consumer:

```
consumer = topic.get_simple_consumer()
```

## کار با Cassandra

می توان کاساندرا را بر روی ابونتو نصب کرد و از دستورات ترمینال برای اجرای آن استفاده کرد: [9]

برای نصب آن نیاز به نصب پایتون و جاوا روی ابونتو داریم.

برای نصب کاساندرا دستور زیر را وارد می کنیم:

(این دستور نسخه ی ۳,۶ کاساندرا را نصب می کند)

```
echo "deb http://www.apache.org/dist/cassandra/debian 36x main" | sudo  
tee -a /etc/apt/sources.list.d/cassandra.sources.list
```

repository keys را اضافه می کنیم:

```
curl https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -
```

حال باید repository ها را آپدیت کنیم:

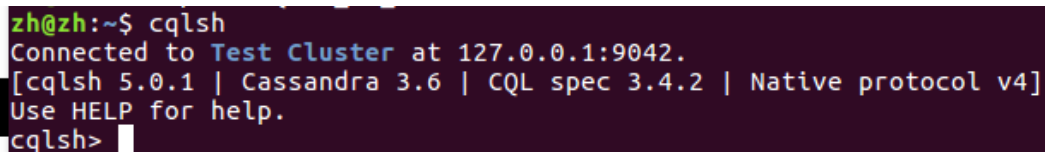
```
sudo apt-get update
```

با دستور زیر نصب انجام می گیرد:

```
sudo apt-get install Cassandra
```

حال با دستور زیر کاساندرا اجرا می شود و می توان دستورات مربوط به پایگاه داده را وارد کرد:

```
cqlsh
```



```
zh@zh:~$ cqlsh  
Connected to Test Cluster at 127.0.0.1:9042.  
[cqlsh 5.0.1 | Cassandra 3.6 | CQL spec 3.4.2 | Native protocol v4]  
Use HELP for help.  
cqlsh> █
```

اگر اجرای کاساندرا با این ارور مواجه شد، لازم است اقدامات زیر را انجام دهیم:

Connection error: ('Unable to connect to any servers', {'127.0.0.1': TypeError('ref() does not take keyword arguments',)}),

```
sudo apt-get install python-pip
```

```
pip install cassandra-driver
```

```
export CQLSH_NO_BUNDLED=true
cqlsh
```

حال بدون خطا اجرا می شود.

ایجاد یک keyspace برای کاساندر:

```
cqlsh> CREATE KEYSPACE IF NOT EXISTS zh WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 3 };
cqlsh>
```

دستورات مربوط به استفاده از کاساندر، ساخت جداول، کوئری ها و غیره را می توان با استفاده از کدهای پایتون نیز اجرا کرد. برای اینکار لازم است مراحل زیر طی شود: [10]

```
pip install cassandra-driver
```

و اتصال به کاساندر در قالب کدپایتون:

```
from cassandra.cluster import Cluster

cluster = Cluster()

session = cluster.connect()
```

وارد کردن کوئری در تابع زیر به صورت استرینگ انجام می گیرد:

```
rows = session.execute('SELECT name, age, email FROM users')
for user_row in rows:
```

## کار با Clickhouse

ابتدا کلیک هوس را بر روی لینوکس نصب کرده ایم. [11] [12]

برای این کار باید چک کنیم که cpu، SSE4.2 را ساپورت می کند :

```
grep -q sse4_2 /proc/cpuinfo && echo "SSE 4.2 supported" || echo "SSE 4.2 not supported"
```

با دستورات زیر پکیج های مورد نیاز را نصب می کنیم:

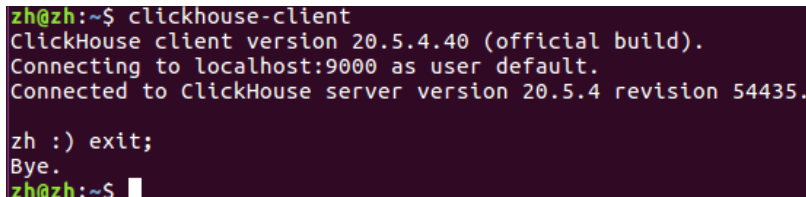
```
sudo apt-get install apt-transport-https ca-certificates dirmngr
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
E0C56BD4

echo "deb https://repo.clickhouse.tech/deb/stable/ main/" | sudo tee \
/etc/apt/sources.list.d/clickhouse.list
sudo apt-get update
```

حال دستور نصب کلیک هوس و اجرای آن:

```
sudo apt-get install -y clickhouse-server clickhouse-client
```

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

A terminal window with a dark background and light-colored text. The text shows the execution of the 'clickhouse-client' command, which connects to the local ClickHouse server. The output includes the client version (20.5.4.40), the connection details (localhost:9000 as user default), and the server version (20.5.4 revision 54435). The user then enters 'exit;' and the prompt returns to the shell.

```
zh@zh:~$ clickhouse-client  
ClickHouse client version 20.5.4.40 (official build).  
Connecting to localhost:9000 as user default.  
Connected to ClickHouse server version 20.5.4 revision 54435.  
  
zh :) exit;  
Bye.  
zh@zh:~$
```

نصب درایور پایتون:

```
pip install clickhouse-driver
```

## کار با Power BI

Power BI را می توان در تحلیل و مصورسازی اطلاعات، پیاده سازی پروژه های هوش تجاری هوش کسب و کار و همچنین در ایجاد گزارشات به صورت تحت وب و موبایل استفاده کرد.

در ارتباط با Power BI می توان گفت که این ابزار از سه کامپوننت اصلی تشکیل شده است بدین معنی که برای تحلیل داده اکثر کار ما با این سه کامپوننت خواهد بود. امروزه اجزای بیشتری به این سرویس اضافه شده است اما اجزای کلیدی و اصلی این سرویس این سه مورد هستند:

### ۱. Power Query

کامپوننت ETL که از آن برای واکنشی داده ها از منابع اطلاعاتی مختلف و تغییر، تبدیل و پالایش آن ها استفاده می شود. همچنین تغییر داده ها، افزودن ستون و... همراه با رابط کاربری آسان و بدون کدنویسی انجام می گردد اما برای اعمال تغییرات پیچیده تر در داده ها می بایست از زبان فرمول نویسی به نام "M" استفاده کرد.

### ۲. Power Pivot

دیتابیس تحلیلی و OLAP Engine که داده ها به صورت فشرده در آن ذخیره می شوند. در این کامپوننت مدل سازی داده و ایجاد انواع محاسبات توسط زبان DAX صورت می گیرد. این محاسبات عبارتند از: مقادیر محاسباتی، ستون های محاسباتی و جداول محاسباتی (Calculated Measure, Calculated Column, Calculated Table). در ادامه باید گفت که سرعت کوئری گرفتن از

مدل به خاطر استفاده از تکنولوژی xVelocity In-Memory analytics engine بسیار بالا است. برای اطلاعات بیشتر می توانید به این مقاله مراجعه کنید

### ۳. Power View

کامپوننت Visualization برای مصورسازی اطلاعات است که با استفاده از نمودار های متنوع آن، داشبورد ها و گزارش ها طراحی می گردد. در واقع این کامپوننت قسمتی است که کاربر نهایی گزارش ها را در آن مشاهده می کند. در این قیمت بصورت پیشفرض چارت های متنوع و قدرتمندی وجود دارد که حتی اگر آن ها نیز جوابگوی نیاز های شما نباشند، می توانید از کاستوم ویژوال ها استفاده کنید که به صورت یک افزونه قابل اضافه شدن به داشبورد ها هستند

اجزای دیگری که در Power BI وجود دارد:

Quick Insight: به کمک این کامپوننت Power BI ب شما می گوید که کدام قسمت از داده های شما نیاز به بررسی دارند (بدیهی است که با استفاده از هوش مصنوعی این کار را انجام می دهد)

Data Flow: این کامپوننت دقیقاً کارایی همان Power Query را دارد که می توان از آن به صورت کاملاً مجزا استفاده کرد.

Q&A: با استفاده از کامپوننت Power Q&A می توان در رابطه با اطلاعات وارد شده به زبان های مختلف سوال پرسید که Power BI به صورت نموداری پاسخ سوالات را نمایش می دهد.

نصب Power BI بر روی ویندوز امکان پذیر است اما پایگاه داده کلیک هوس، روی لینوکس قابل نصب است، بنابراین لازم است یک سری تنظیمات برای اتصال این دو انجام گیرد. راه های مختلفی برای این کار وجود دارد؛ می توان از حافظه های ابری استفاده کرد و کلیک هوس را روی آن ها وصل کرده و آدرس آی پی آن را وارد ODBC ویندوز کنیم ( لازم است که ابتدا ODBC را برای ویندوز نصب کنیم) سپس در Power BI، با انتخاب گزینه ی ODBC، مشخصات آن را وارد می کنیم. اینک پاور بی آی به کلیک هوس موجود در فضای ابری، متصل شده است. [13]

روش دیگر این است که از windows subsystem for linux(wsl) استفاده کنیم و در داخل ویندوز از طریق زیر لینوکس را نصب کنیم و آدرس آی پی آن را وارد ODBC ویندوز کنیم ( لازم است که ابتدا ODBC را برای ویندوز نصب کنیم) سپس در Power BI، با انتخاب گزینه ی ODBC، مشخصات آن را وارد می کنیم. اینک پاور بی آی به کلیک هوس ران شده در wsl، متصل شده است.

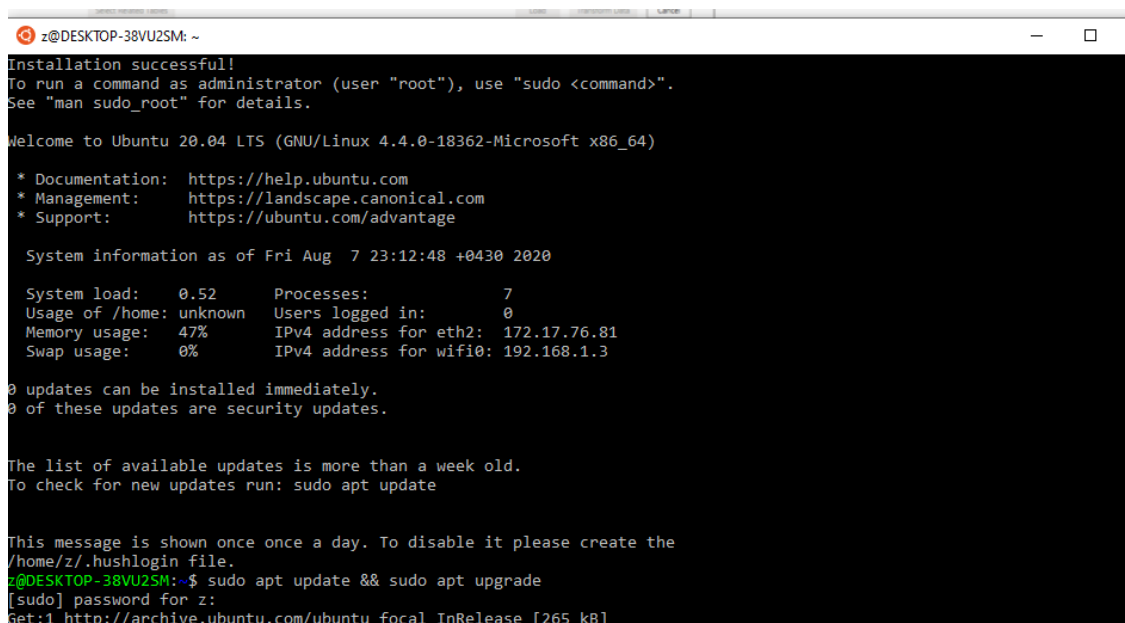
## کار با WSL

در داخل power shell ویندوز دستور زیر را وارد کرده ایم تا امکان نصب لینوکس در ویندوز را فعال کند: [14]

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

سپس ویندوز را دی استارت می کنیم.

حال از Microsoft store نسخه ی ابونتوی آن را دانلود کرده و آن را اجرا می کنیم:



```
z@DESKTOP-38VU2SM: ~  
Installation successful!  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
Welcome to Ubuntu 20.04 LTS (GNU/Linux 4.4.0-18362-Microsoft x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
System information as of Fri Aug  7 23:12:48 +0430 2020  
  
System load:  0.52      Processes:            7  
Usage of /home: unknown  Users logged in:      0  
Memory usage: 47%      IPv4 address for eth2: 172.17.76.81  
Swap usage:   0%       IPv4 address for wifi0: 192.168.1.3  
  
0 updates can be installed immediately.  
0 of these updates are security updates.  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
  
This message is shown once once a day. To disable it please create the  
/home/z/.hushlogin file.  
z@DESKTOP-38VU2SM:~$ sudo apt update && sudo apt upgrade  
[sudo] password for z:  
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
```

حال با دستوراتی که در clickhouse ذکر شده، در این لینوکس وارد کرده و آن را نصب کردیم.

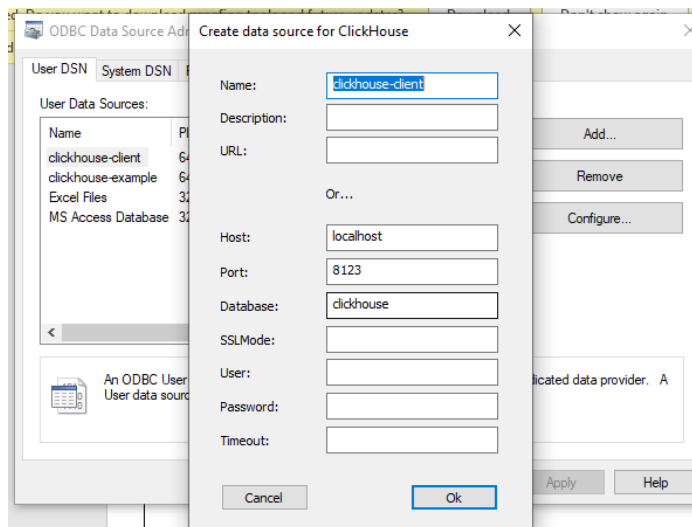
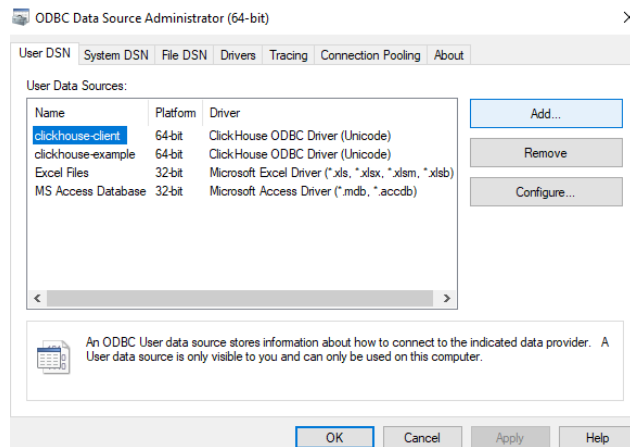
بعد از نصب، آدرس localhost:9000 در اختیار کلیک هوس قرار گرفته است که با اجرای آن در این محیط، می توان در ویندوز از آن استفاده کرد.

## تنظیمات ODBC

از لینک زیر ODBC متناسب با نسخه ی ویندوز نصب کرده ایم:

<https://github.com/ClickHouse/clickHouse-odbc/releases>

سپس پایگاه داده مربوطه را به آن اضافه کردیم:





- [1] [Online]. Available: <https://kafka.apache.org/quickstart>.
- [2] [Online]. Available: <https://www.elastic.co/downloads/elasticsearch>.
- [3] [Online]. Available: <https://www.elastic.co/downloads/kibana>.
- [4] [Online]. Available: <https://www.tutorialspoint.com/cassandra/index.htm>.
- [5] [Online]. Available: <https://redis.io/topics/data-types-intro>.
- [6] [Online]. Available: <https://clickhouse-driver.readthedocs.io/en/latest/index.html>.
- [7] [Online]. Available: <https://zookeeper.apache.org/>.
- [8] [Online]. Available: <https://pykafka.readthedocs.io/en/2.0.4/>.
- [9] [Online]. Available: <https://www.liquidweb.com/kb/install-cassandra-ubuntu-16-04-lts/>.
- [10] [Online]. Available: <https://docs.datastax.com/en/developer/python-driver/3.24/>.
- [11] [Online]. Available: <https://altinity.com/blog/clickhouse-and-python-getting-to-know-the-clickhouse-driver-client>.
- [12] [Online]. Available: <https://clickhouse.tech/docs/en/getting-started/install/>.
- [13] [Online]. Available: <https://medium.com/@annu.nmit/connecting-to-clickhouse-database-server-from-power-bi-864e950d83e2>.
- [14] [Online]. Available: <https://docs.microsoft.com/en-us/windows/wsl/install-win10>.