

# به نام خدا



دانشگاه تهران  
پردیس دانشکده‌های فنی  
دانشکده برق و کامپیوتر



## درس تحلیل ها و سیستم های داده های حجیم

پروژه شماره یک  
فاز دوم

نام و نام خانوادگی :

سحر رجبی

شماره دانشجویی :

۸۱۰۱۹۹۱۶۵

خرداد ۱۴۰۱

فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را بهروز کنید.)

- 1 سوال
- 2 سوال
- 3 سوال
- 4 سوال

## سوال ۱ - دریافت داده و پیش‌پردازش

### قیمت لحظه‌ای ارزهای دیجیتال

برای دریافت لحظه‌ای قیمت ارزهای دیجیتال، از `curl`-point که برای آپدیت استفاده می‌کنند، استفاده می‌کنیم. به این صورت که با استفاده از دستور `curl` قیمت‌ها را برای ارزهای موجود در صفحه دریافت می‌کنیم که در فرمت `json` خروجی می‌دهد. سپس با استفاده از `pipe` (عملگر `|`) این خروجی را به ابزار `jq` می‌دهیم تا بتوانیم اطلاعات لازم را از `json` استخراج کنیم. برای این کار دستور زیر را اجرا کردہایم که در ادامه توضیح خواهیم داد.

```
jq -r '.data | map([.title_fa,.title_en,.symbol,.p,.p_irr,.dp,.datetime] | join("\t")) | join("\n")'
```

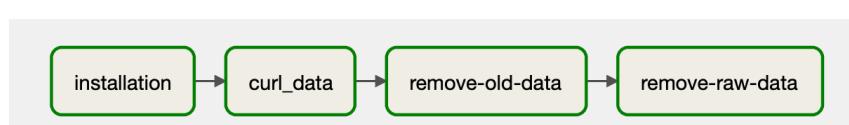
ما `key` که برابر با `data` است در `json` اصلی را می‌گیریم (اطلاعات مد نظر ما در این فیلد قرار دارد که به صورتی لیستی از `json`‌هاست) و در هر کدام از آن‌ها، مقادیر خواسته شده را که در بالا هم قابل مشاهده است در یک لیست قرار می‌دهیم و با استفاده از `map`، آن‌ها را `join` می‌کنیم. نتیجه‌ی این فرآیند را به تابع `map` می‌دهیم تا این نتیجه که برای هر خط حاصل می‌شود را با استفاده از `new-line` جدا کنیم.

فرمت خروجی این دستور به صورت یک `tsv` است که ما آن را در یک فایل موقت، به صورت لوکال ذخیره می‌کنیم.

سپس باید داده‌هایی که برای بیش از یک ساعت گذشته هستند را از بین اطلاعات رمزارزها حذف کنیم. به این منظور با استفاده از کتابخانه `jdatetime` که مشابه کتابخانه `astandar` استاندارد `datetime`، اما برای تاریخ جلالی است؛ زمان امروز را به دست می‌آوریم و سپس با استفاده از کتابخانه `pandas`، فایل ذخیره شده در قسمت `instance` را باز می‌کنیم. سپس با باز کردن استرینگ `time`، زمان گزارش شده در هر خط از داده را به یک `pd.datetime` تبدیل می‌کنیم و اگر با زمان حال حاضر بیش از یک ساعت اختلاف داشته باشد در ستون `old` که برای هر داده اضافه می‌کنیم مقدار `true` قرار می‌گیرد و برعکس. در نهایت رکوردهایی که ستون `old` آن‌ها است را از `dataframe` حذف می‌کنیم و بعد از حذف ستون `old`، داده را مجددا در یک فایل `tsv` که نام آن با زمان مشخص می‌شود ذخیره می‌کنیم. کد این بخش در فایل `crypto_remove_old_rows.py` قرار دارد.

بعد از انجام این کار، دیتایی که در بخش اول و قبل از حذف داده‌های قدیمی، ذخیره کرده بودیم را هم حذف می‌کنیم.

تمامی توالی توضیحات بالا، هر یک ساعت یک بار اجرا می‌شوند و در هر ساعت یک فایل `tsv` با زمان مشخص ذخیر می‌شود. `DAG` مورد استفاده برای این بخش به این صورت است:



(بخش installation به این دلیل اضافه شده که برای نصب پکیج‌های مورد استفاده در کدها، امکان اینکه داکرفایل را تغییر بدهیم وجود نداشت. چرا که به علت تحریم‌ها و سرعت بسیار پایین دانلود در این چند روز زمان زیادی مصرف می‌شد. راه درست این بود که نصب‌ها را در DockerFile اضافه کنیم و هر بار این بخش اجرا نشود).

بخش curl\_data بخش گرفتن داده از سایت است، تاکم remove\_old\_data داده‌های مربوط به بیش از یک ساعت قبل را حذف می‌کند و remove\_raw\_data هم داده‌های پردازش نشده را حذف می‌کند.

کد این DAG در فایل crypto\_dag.py قرار دارد.

کار دیگری که باید صورت بگیرد این است که داده‌های مربوط به هر ۲۴ ساعت با هم ادغام شوند و پس از ادغام در hdfs قرار بگیرند. همچنین فایل داده‌های ادغام شده هم پاک شود. برای این کار، DAG بین بخش news و بخش crypto مشترک است که در قسمت بعدی توضیح داده می‌شود.

### خبر

در این بخش هم باید به صفحه‌ی اخبار در لینک داده شده می‌رفتیم و اطلاعات خواسته شده برای هر خبر را استخراج می‌کردیم.

برای دریافت داده‌ها، از دستور wget به صورت recursive استفاده کردیم که عمق آن برابر ۱ باشد و تنها اخباری را قبول کند که فرمت url آن‌ها به صورت [news/][7-digits code] باشد و آن‌ها را ذخیره کند. برای انجام این کار از دستور wget به صورت زیر استفاده کردیم.

```
 wget --tries 1 -user-agent "BigDataClass" --recursive -q --level 1 --accept-regex '/news/[0-9]+/\w.*' https://www.tgju.org/news/
```

بعد از دریافت داده‌ها، آن‌ها در فolder www.tgju.org/news ذخیره می‌شوند و برای هر خبر یک فolder با عنوان کد خبر ایجاد می‌شود که در آن صفحه‌ی کرال شده قرار دارد.

برای استخراج اطلاعات هر صفحه، هر یک از فایل‌های کرال شده را باز می‌کنیم و با استفاده از BeautifulSoup فیلدهای مذکور را به دست می‌آوریم و آن‌ها را در یک فایل csv با نام زمان ذخیره‌سازی ذخیره می‌کنیم. کدهای این بخش در فایل tgju\_parser.py قرار دارد. بعد از فایل‌های کرال شده را کاملاً پاک می‌کنیم.

در این قسمت، DAG استفاده شده به صورت زیر است:



که همان دلیل توضیح داده شده در بخش قبل اضافه شده است، همان دستور crawl-data همان است، همان که همان دلیل توضیح داده شده در بخش قبل اضافه شده است، همان extract-from-webpages است، همان که همان دستور wget است، همان که فیلدهای خواسته شده را استخراج

می‌کند و تسک آخر، برای پاک کردن داده‌های کرال شده است. کد DAG این بخش، در فایل `news_dag.py` قرار دارد.

یک بخش مهم که هم برای قسمت رمزارز و هم برای اخبار باقی‌مانده است، ادغام داده‌های ۲۴ ساعت اخیر است.

در این بخش از یک DAG استفاده کردہ‌ایم که هر ۲۴ ساعت اجرا می‌شود و داده‌های روز قبل را با یکدیگر ادغام می‌کند و در `hdfs` قرار می‌دهد.

یک تسک شروع داریم که کتابخانه‌های مورد نیاز را نصب می‌کند (به علتی که برای `installation` توضیح داده شد). و بعد از آن دو شاخه در درخت DAG به وجود می‌آید که یکی برای داده‌های رمزارز است و دیگری برای اخبار.

برای بخش رمزارز، ابتدا کد نوشته شده در فایل `crypto_merge.py` را اجرا می‌کنیم که با به‌دست آوردن تاریخ روز گذشته، تمامی فایل‌های `tsv` که نام آن‌ها شامل تاریخ روز گذشته است را باز می‌کند و داده‌های آن را در یک فایل که این بار تنها شامل تاریخ است (و فاقد ساعت است) ذخیره می‌کند. همچنین این کد، در نهایت لیست فایل‌هایی که در روز گذشته ایجاد شده‌بودند را هم بر می‌گرداند و ما خروجی این کد را با استفاده از `pipe` (عملگر `|`) به دستور `rm` می‌دهیم تا فایل‌هایی که ادغام شده‌اند را حذف کند.

دقیقاً همین کد، و با همین منطق و توالی، برای فایل‌های `csv` ایجاد شده برای اخبار روز قبل هم اجرا می‌شود.

در شاخه‌ی مربوط به اخبار، یک تسک اضافه وجود دارد که داده‌هایی که چند بار کرال شده باشند را حذف می‌کند و تنها یک نسخه از آن‌ها را نگهداری می‌کند. این کار با اجرای فایل `news_remove_redundancy` انجام می‌شود که در آن آخرین فایل `csv` ساخته شده را دریافت می‌کنیم (این فایل در مرحله‌ی قبل ساخته شده است) و با استفاده از کتابخانه `pandas` آن را می‌خوانیم و داده‌های تکراری آن را حذف می‌کنیم. سپس مجدداً در فایلی به همان نام قبلی سیو می‌کنیم.

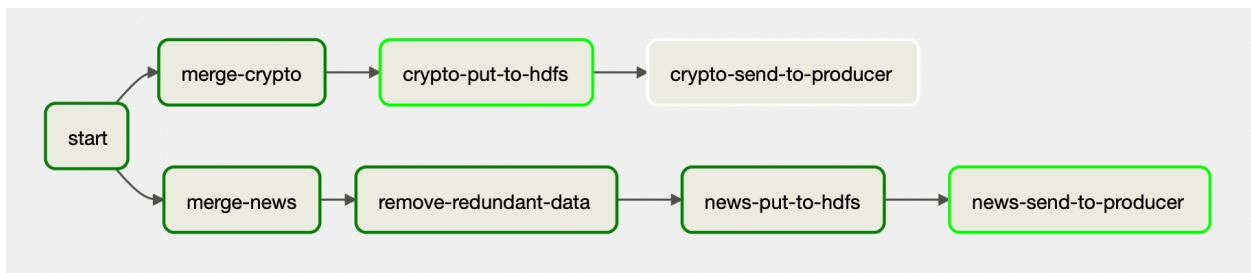
بعد از این مراحل، هر دو شاخه‌ی این DAG آخرین فایل `csv` یا `tsv` ساخته شده را در `hdfs` قرار می‌دهند. این کار به این صورت انجام می‌شود که ابتدا فولدرهایی در `hdfs` با عنوان `[year]` و `[month]` ساخته می‌شوند (که `month` در داخل فولدر `year` قرار دارد) سپس فایل مذکور در این پوشه قرار می‌گیرد. در تمامی مراحل، از آنجایی که ممکن است کاری در قبل انجام شده باشد و بخواهیم مجدداً یک فولدر در `hdfs` بسازیم و یا داده را کپی کنیم (این کار می‌تواند به علت `fail` کردن `data pipeline` اتفاق بیفتد) خروجی را با `true`، `or` کردیم تا بتوانیم مراحل بعدی `pipeline` را انجام دهیم.

بعد از ذخیره‌ی فایل‌ها در `hdfs`، در هر دو شاخه، یک نمونه از کد `produce_from_hdfs` را اجرا می‌کنیم که با دریافت دو ورودی، شامل نوع داده (`news` یا `crypto`) و آخرین فایلی که در این دسته ایجاد شده، کد را اجرا می‌کند. در این کد، یک `KafkaProducer` از کتابخانه‌ی `KafkaProducer` ایجاد می‌کنیم که هر خط از داده

را، به بایت تبدیل می‌کند (با انکدینگ UTF-8) و داده‌های رمزارز را روی کانال crypto و همچنین داده‌های اخبار را روی تاپیک news قرار می‌دهد.

کد DAG این بخش در فایل merging\_dag.py قرار دارد.

در زیر، DAG مربوط به این بخش که بین رمزارز و اخبار مشترک است را مشاهده می‌کنید.



شاخه‌ی بالا مربوط به رمزارز است و شاخه‌ی پایین مربوط به اخبار، که همان‌طور که توضیح داده شد مرحله‌ی remove-redundant-data را اضافه دارد.

## سوال ۲- استفاده از صفت توزیع شده‌ی kafka

### کانال (Preprocess)

همان‌طور که در سوال ۱ توضیح داده شد، در انتهای بخش ادغام ۲۴ ساعته‌ی داده‌ها، یک producer-Kafka قرار می‌داد و داده‌ها اخبار را بر روی تاپیک news و DAG به اتمام می‌رسید.

اما ما یک کد تحت عنوان consumer داریم، که در آن یک preprocessor بر روی این دو تاپیک گوش می‌ایستد. در حلقه‌ی این کد، اگر پیام روی تاپیک crypto دریافت شده باشد، preprocess ما از نوع پیش‌پردازش زبانی و زبان فارسی نخواهد بود. چرا که داده‌ها اصلاً متنی ندارند. تنها کاری که در این بخش انجام می‌شود این است که داده را به فرم json در می‌آوریم و همچنین time را در آن از شمسی، به میلادی تبدیل کنیم. سپس دو producer این مقادیر را بر روی دو کانال قرار می‌دهند و تاپیک آن‌ها هم crypto-statistics و persistence می‌باشد.

برای زمانی که داده روی تاپیک news دریافت می‌شود، ما باید از Persian analyzer در الستیک استفاده کنیم. به این منظور ابتدا یک PUT request به elastic می‌زنیم که در آن Persian analyzer را index می‌کنیم تا بتوانیم داده‌ها را به آن داده و پردازش شده‌ی آن‌ها را دریافت کنیم. Request به صورت زیر است:

```
res = requests.put(
    'http://es-container:9200/persian_analyzer',
    json={
        "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"
                        ]
                    }
                }
            }
        }
    )
)
```

سپس برای هر پیام، بخش `text`, `title` و `summary` که شامل متن فارسی هستند را، با یک POST request که به ایندکس `persian_analyzer` که قبلاً تعریف شده می‌رود و در آن به بخش `_analyze`، می‌فرستیم که در آن مقدار `text` برابر با داده‌ای است که می‌خواهیم پردازش کنیم و `analyzer` برابر با نام آنالیزور که در این کیس تعريف شده است قرار می‌دهیم. این آنالیزور مشکلات مربوط به نیمفاصله و فاصله برای زبان فارسی، حذف stopwordها و همچنین stemming را انجام می‌دهد و نتیجه را در غالب یک json به ما بر می‌گرداند که به صورت آرایه‌ای از token تشکیل شده است و اطلاعات آن توکن را به همراه خود token برگردانده است. ما در نهایت این token‌ها را مجدداً با فاصله join می‌کنیم و به عنوان فیلدی از داده استفاده می‌کنیم. ریکوئستی که به استیک در این بخش زده می‌شود به صورت زیر است:

بعد از دریافت داده‌ای پردازش شده، برای news هم یک json می‌سازیم که هر فیلد را در آن قرار می‌دهیم

```
data_res = requests.post(
    'http://es-container:9200/persian_analyzer/_analyze',
    json={
        "text": data,
        "analyzer": "rebuilt_persian"
    },
)
```

و همچنین زمان آن را هم به میلادی تبدیل می‌کنیم. سپس این داده‌ها را برای دو تاپیک news-persistence و news-statistics منتشر می‌کنیم تا در کانال‌های persistence و statistics به آن‌ها دسترسی داشته باشیم. در نتیجه در این بخش داده‌های قرار گرفته در کانال preprocess را دریافت کردیم، و بر روی کانال‌های statistics و persistence قرار دادیم.

## کانال (Persistence)

در بخش قبل گفته شد که داده‌های crypto و news با تاپیک‌های persistence و crypto-persistence بر روی کانال preprocess قرار می‌گیرند. در فایل persistence.py ما یک consumer داریم که بر روی این دو تاپیک گوش می‌کند و داده‌ای که دریافت می‌کند را به استیک می‌فرستد. به این صورت که با توجه به تاپیک، ایندکس را مشخص می‌کند و سپس داده را به عنوان یک رکورد جدید در آن ایندکس ثبت می‌کند. ریکوئست ارسالی به استیک به این صورت است:

```
headers = {'Content-type': 'application/json'}
data_res = requests.post(
    f'http://es-container:9200/{msg_topic}/_doc',
    msg_value,
    headers=headers
)
```

## کانال (Statistics)

در قسمت Preprocess گفته شد که داده‌ی پیش‌پردازش شده با تاپیک‌های news- crypto-statistics و statistics بر روی کانال presto قرار می‌گیرند و حال باید با وصل شدن به کافکا، داده را از این دو تاپیک به دست بیاورد و کوئری‌های خواسته شده را پاسخ دهد.

## سوال ۳ - kibana و elastic-search

توضیحات مربوط به این بخش در سوال ۲ داده شده اما برای راحتی در اینجا هم اضافه می شود.

همان طور که در سوال ۱ توضیح داده شد، در انتهای بخش ادغام ۲۴ ساعته‌ی داده‌ها، یک producer-Kafka همان‌طور که در سوال ۱ توضیح داده شد؛ در انتهای بخش ادغام ۲۴ ساعته‌ی داده‌ها، یک producer-Kafka قرار می‌داد و داده‌ها اخبار را بر روی تاپیک news و DAG به اتمام داده‌های رمزارز را بر روی تاپیک crypto قرار می‌داد و داده‌ها اخبار را بر روی تاپیک news و DAG به اتمام می‌رسید.

اما ما یک کد تحت عنوان consumer داریم، که در آن یک consumer بر روی این دو تاپیک گوش می‌ایستد. در حلقه‌ی این کد، اگر پیام روی تاپیک crypto دریافت شده باشد، preprocess ما از نوع پیش‌پردازش زبانی و زبان فارسی نخواهد بود. چرا که داده‌ها اصلاً متني ندارند. تنها کاری که در این بخش انجام می‌شود این است که داده را به فرم json در می‌آوریم و همچنین time را در آن از شمسی، به میلادی تبدیل crypto. سپس دو producer این مقادیر را بر روی دو کانال قرار می‌دهند و تاپیک آن‌ها هم crypto-statistics و persistence می‌باشد.

برای زمانی که داده روی تاپیک news دریافت می‌شود، ما باید از Persian analyzer در الستیک استفاده کنیم. به این منظور ابتدا یک PUT request به elastic می‌زنیم که در آن Persian analyzer را index می‌کنیم تا بتوانیم داده‌ها را به آن داده و پردازش شده‌ی آن‌ها را دریافت کنیم. Request به صورت زیر است:

```
res = requests.put(
    'http://es-container:9200/persian_analyzer',
    json={
        "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_filters": ["zero_width_spaces"],
                        "filter": [
                            "lowercase",
                            "decimal_digit",
                            "arabic_normalization",
                            "persian_normalization",
                            "persian_stop"
                        ]
                    }
                }
            }
        }
    )
)
```

سپس برای هر پیام، بخش `text`, `title` و `summary` که شامل متن فارسی هستند را، با یک POST request که به ایندکس `persian_analyzer` که قبلاً تعریف شده می‌رود و در آن به بخش `_analyze`، می‌فرستیم که در آن مقدار `text` برابر با داده‌ای است که می‌خواهیم پردازش کنیم و `analyzer` برابر با نام آنالیزور که در این کیس تعريف شده است قرار می‌دهیم. این آنالیزور مشکلات مربوط به نیمفاصله و فاصله برای زبان فارسی، حذف stopwordها و همچنین stemming را انجام می‌دهد و نتیجه را در غالب یک json به ما بر می‌گرداند که به صورت آرایه‌ای از token تشکیل شده است و اطلاعات آن توکن را به همراه خود token برگردانده است. ما در نهایت این token‌ها را مجدداً با فاصله join می‌کنیم و به عنوان فیلدی از داده استفاده می‌کنیم. ریکوئستی که به استیک در این بخش زده می‌شود به صورت زیر است:

```
data_res = requests.post(
    'http://es-container:9200/persian_analyzer/_analyze',
    json={
        "text": data,
        "analyzer": "rebuilt_persian"
    },
)
```

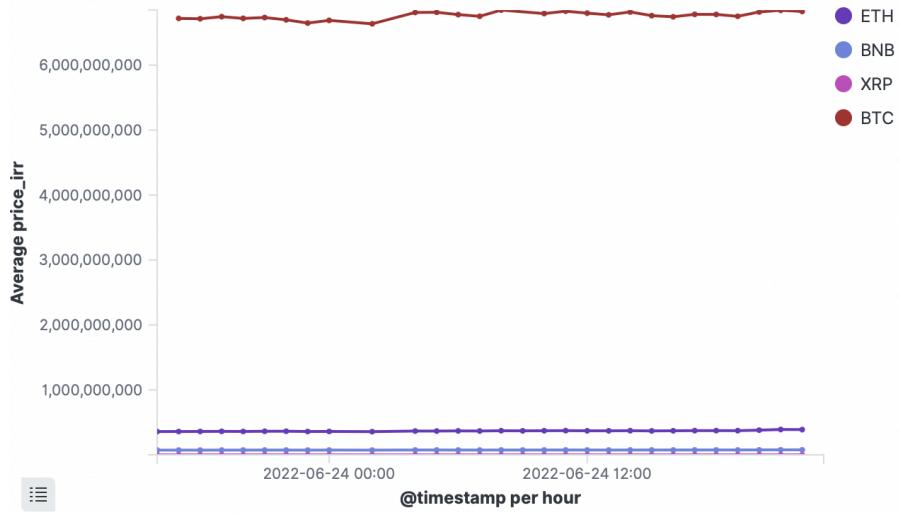
در بخش قبل گفته شد که داده‌های crypto و news با تاپیک‌های news- crypto-persistence بر روی کانال persistence قرار می‌گیرند. در فایل consumer.py ما یک consumer که بر روی این دو تاپیک گوش می‌کند و داده‌ای که دریافت می‌کند را به استیک می‌فرستد. به این صورت که با توجه به تاپیک، ایندکس را مشخص می‌کند و سپس داده را به عنوان یک رکورد جدید در آن ایندکس ثبت می‌کند. ریکوئست ارسالی به استیک به این صورت است:

```
headers = {'Content-type': 'application/json'}
data_res = requests.post(
    f'http://es-container:9200/{msg_topic}/_doc',
    msg_value,
    headers=headers
)
```

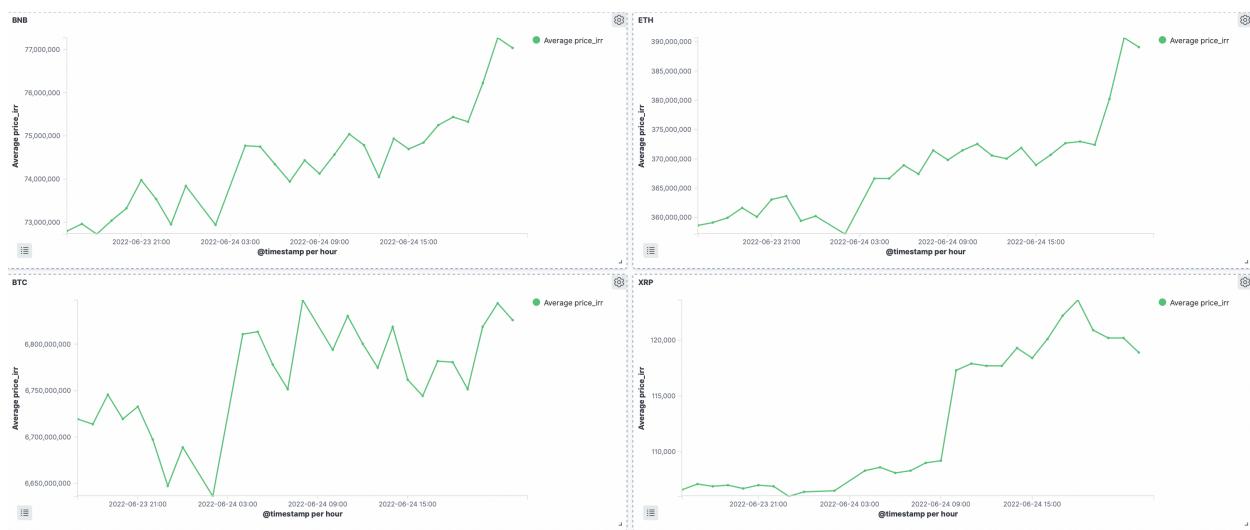
سپس با استفاده از kibanna که به داده‌های elastic-search دسترسی دارد، موارد خواسته شده را در دشبورد ایجاد می‌کنیم.

- در بخش اول خواسته شده که قیمت‌های ۴ ارز را نمایش بدهیم، و بعد بررسی کنیم که آیا تغییرات ارزها بر یکدیگر اثری دارند یا خیر.

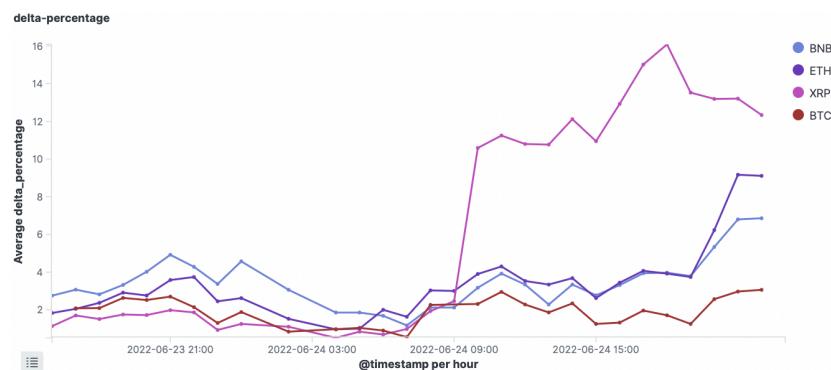
برای نمایش قیمت تمامی ارزها در یک پلات، برای ۴ ارز با نمادهای BNB، BTC، ETH و XRP پلات زیر به دست آمد که همانطور که مشخص است به علت تفاوت بسیار زیاد قیمت بیت‌کوین و بقیه، این پلات اطلاعات زیادی در اختیار ما نمی‌گذارد.



به همین علت، نمودار را ابتدا برای هر رمزارز به صورت مجزا رسم کردیم، که ۴ پلات زیر نتیجه‌ی آن‌هاست:



و در نگاه اول به نظر می‌رسد که تغییرات قیمت ارزها، از شیوه‌های کلی یکسانی پیروی کرده است. برای بررسی دقیق‌تر، این بار نمودار درصد تغییرات هر یک را در یک پلات رسم کردیم که حاصل شکل زیر است:



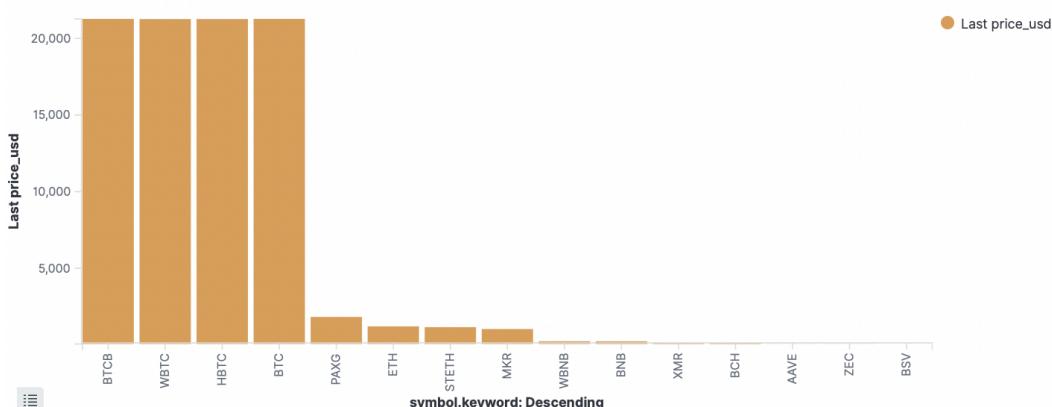
و باز هم همانطور که مشخص است، سه ارز BTC و ETH و BNB صعود و نزول های تقریباً یکسان و حتی با شبی مشابه دارند و ارز XRP با اینکه گاهها با شبی متفاوتی فاصله گرفته است اما در مجموع کمی هماهنگی دارد.

در نتیجه با بررسی ویژوال داده ها، به نظر می رسد که قیمت ارزها بر روی یکدیگر اثر می گذارند. البته این اثر گذاری بین دسته های مختلف، متفاوت است.

برای رسم پلات های قیمت رمزارزها، ابتدا در پنل ایجاد شده با استفاده از قسمت filters داده ها را محدود به ارز دیجیتال مورد نظر کردیم، سپس محور عمودی را برابر با فیلد price\_irr و محور افقی را از جنس date با دقت یک ساعته انتخاب کردیم تا داده ها را رسم کند.

برای ۴ نمودار در یک صفحه هم با انتخاب محور عمودی مورد نظر، محور افقی را مشابه قبل ایجاد می کنیم و فقط در قسمت filter باید لیستی از موارد قابل قبول را وارد کنیم.

- نمودار به دست آمده در این قسمت، به صورت زیر است:



برای رسم این نمودار، با انتخاب نمودار میله ای، بر روی محور عمودی aggregation را برابر top-hit قرار دادیم تا بتوانیم فیلد price\_usd را انتخاب کنیم در حالی که aggregation داخلی آن بر روی زمان به صورت نزولی مرتب شده است و تنها یکی از آن ها را انتخاب می کنیم. به این صورت آخرین قیمت از دسته هایی که در پایین مشخص می شود را انتخاب می کنیم.

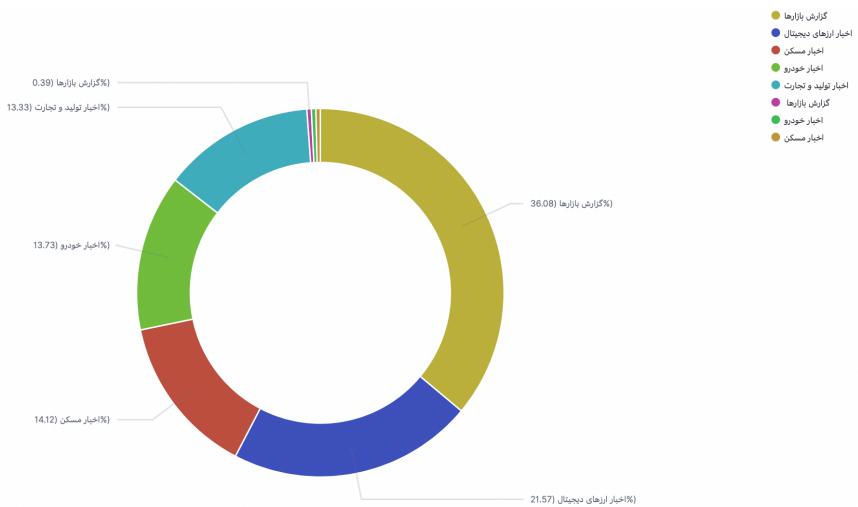
محور افقی هم در قسمت buckets بر روی symbol گروه شده. برای نمایش ۱۵ تای اول، orderby را در حالت custom قرار می دهیم و بر روی ماکسیمم مقادیر price\_usd مرتب می کنیم. بعد از این تنظیمات، بر حسب قیمت به صورت نزولی مرتب می شوند که ما ۱۵ تای اول را برای نمایش انتخاب می کنیم.

- در بخش سوم، ابتدا خواسته شده که با نمودار دایره ای تعداد اخبار در هر دسته را گزارش کنیم.

برای این کار در قسمت buckets، بر روی category.keywords گروه می کنیم و بر اساس تعداد هر دسته مرتب می کنیم. تصویر زیر نمایانگر نتیجه ای این تنظیمات است:

text.keyword: Descending	time
دکس المان 12912.59 فنی 100 شاخص_های_بورس_اروپا 7020.45 100	Jun 24, 2022 @ 15:10:00.000
... شاخص_های_بورس_امريكا 11232.19 نزدك 30677.36 داوجونز	Jun 24, 2022 @ 13:15:00.000
شاخص_های_بورس_اسياي 3183.31 داوجونز آسيا 261717.25 225 ...	Jun 24, 2022 @ 07:50:00.000
شاخص_های_بورس_امريكا 11081.68 نزدك 30359.59 داوجونز ...	Jun 24, 2022 @ 02:15:00.000
شاخص_های_بورس_اسياي 3177.34 داوجونز آسيا 261717.25 225 ...	Jun 23, 2022 @ 21:50:00.000

توجه کنید که در دسته‌های بالا، دو گزارش بازارها وجود دارد که احتمالاً برای وجود کاراکتر اضافی‌ای مانند



نیم‌فاصله است. در قسمت بعدی مشکلی که با آن مواجه می‌شدیم این بود که هر دوی پر تکرارترین و کم تکرارترین دسته گزارش بازار بودند! به همین دلیل برای کم تکرارترین دو رکورد را نمایش داده‌ایم تا به کم تکرارترین واقعی دست پیدا کنیم. همچنین گفته شده که تکرار کلمات در هفته‌ی اخیر بررسی شود که به علت اینکه تنها دیتای ۲ روز را موجود داشتیم، در همین دو روز نمایش دادیم. اما برای محدود کردن زمان، در بالای پنل، می‌توانیم زمان مورد نظر خود را تعیین کنیم. برای نمایش این دو دسته در dashboard از table استفاده کردیم.

least-common	most-common
category.keyword: Descending	category.keyword: Ascending
گزارش_بازارها	گزارش_بازارها
92	1
	اخبار_خودرو
	1

همانطور که مشخص است، گزارش بازار بیشترین تکرار و اخبار خودرو کمترین تکرار را دارد.

- برای نمایش متن ۵ خبر اخیر، اگر بخواهیم در یک جدول در dashboard این کار را بکنیم باید برای متريک اول aggregation top hit را قرار دهیم تا بر روی timestamp داده‌ها را مرتب کنیم و در قسمت

بگذارید که در قسمی از custom metrics می‌توانیم timestamp را از بین های مختلف برای یک text فیلد بررسی کنیم.

خروجی این تنظیمات، جدول زیر است:

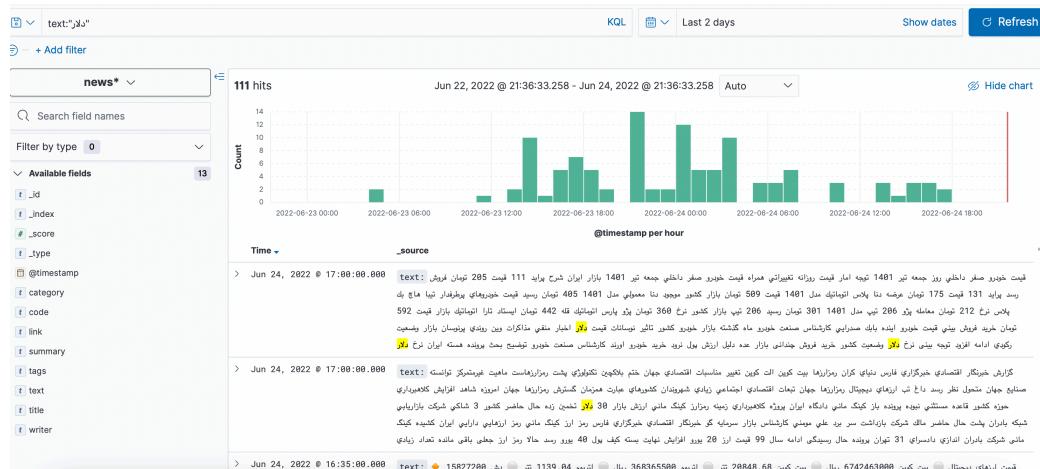
- برای نمایش ابر کلمات، باید می‌توانستیم به کلمات فیلد text دسترسی داشته باشیم. در حالت عادی این امکان نبود. به همین منظور، با یک put request مپینگ مربوط به این فیلد را آپدیت کردیم تا مقدار fielddata را برای آن برابر با true قرار دهد. ریکوئست زده شده به صورت زیر است:

```
PUT ▾ http://localhost:9200/news/_mapping
JSON ▾ Auth ▾ Query Header ① Docs
1 ↴ { "properties": {
2   "text": {
3     "type": "text",
4     "fieldData": true
5   }
6 }
7 }
8 }
```

بعد از انجام این کار، به سادگی با انتخاب cloud tag و قرار دادن text در buckets، ابر کلمات را ایجاد می‌کند. البته از آنجایی که ۱۵ روز داده در دست نداشتیم، برای همان دو روزی که در اختیار داشتیم رسم شد اما باز هم می‌توانستیم در قسمت بالای پل، معین کنیم که از چند روز قبل از امروز کلمات را نمایش دهد. ابر کلمات رسم شده را در زیر می‌بینید:



- برای نوشتن کوئیری‌ای که اخباری که کلمه‌ی اتریوم در متن خبر آن‌ها به کار رفته باشد، می‌توانیم در قسمت **discover**, "اتریوم" را تایپ کنیم و نتیجه در زیر قابل مشاهده است. (البته به علت اینکه در لیست اخبار ما این کلمه وجود نداشت، به جای آن به جستجوی کلمه‌ی دلار پرداختیم).



## سوال ۴ - بررسی آماری به کمک presto

همانطور که اشاره شد، داده‌ها از کanal news-statistics و crypto-statistics با تاپیک preprocess از کanal Kafka.properties را در پوششی statistics وارد می‌شوند. برای اتصال کافکا به presto باید فایل catalog را در پوششی Kafka.properties قرار می‌دادیم که محتویات آن به صورت زیر است:

```
connector.name=kafka
kafka.nodes=broker:9092
kafka.table-names=crypto_statistics,news_statistics
kafka.hide-internal-columns=false
```

همچنین خود presto هم با استفاده از image ای که در پروژه‌ی سوم داده شده بود بالا آمد.

برای آنکه بتوانیم روی فیلدهای داخل message \_producer (که در واقع پیامی است که از سمت Kafka برای تاپیک منتشر می‌شود) کوئری بزنیم، در کنار پوششی etc\_coordinator که پوششی catalogs هم داخل آن قرار دارد؛ یک فولدر با نام Kafka می‌سازیم و دو فایل، یکی برای داده‌های crypto و دیگری برای داده‌های news می‌سازیم و در آن هر فیلد را با نوع آن و .. تعریف می‌کنیم. برای مثال، فایل زیر مربوط به داده‌های crypto است.

```
{
  "tableName": "crypto_statistics",
  "schemaName": "default",
  "topicName": "crypto_statistics",
  "message": {
    "dataFormat": "json",
    "fields": [
      {
        "name": "title_fa",
        "mapping": "title_fa",
        "type": "VARCHAR"
      },
      {
        "name": "title_en",
        "mapping": "title_en",
        "type": "VARCHAR"
      },
      {
        "name": "symbol",
        "mapping": "symbol",
        "type": "VARCHAR"
      },
      {
        "name": "price_usd",
        "mapping": "price_usd",
        "type": "DOUBLE"
      },
      {
        "name": "price_irr",
        "mapping": "price_irr",
        "type": "DOUBLE"
      },
      {
        "name": "delta_percentage",
        "mapping": "delta_percentage",
        "type": "DOUBLE"
      },
      {
        "name": "@timestamp",
        "mapping": "@timestamp",
        "type": "TIMESTAMP",
        "dataFormat": "iso8601"
      }
    ]
  }
}
```

اضافه کردن این اطلاعات پیچیدگی خاصی ندارد. اما در مورد فیلد `timestamp`، برای اینکه `presto` بتواند با `string` که از تابع `datetime` در پایتون و به فرمت `iso` ایجاد شده است؛ مشابه `timestamp` کار کند، نوع آن را `timestamp` قرار می‌دهیم و فرمت آن را `iso8601` مشخص می‌کنیم تا بتوانیم کوئری‌های مربوط به زمان را بر روی آن اجرا کنیم.

مشابه همین کار را برای داده‌های `news` هم انجام می‌دهیم و بعد به سراغ کوئری‌ها می‌رویم.

#### - کوئری اول:

برای این کوئری که ما در حد ماه و حتی هفته داده در اختیار نداشتیم، اما از آنجایی که مشخص کردن زمان در هر دو حالت شکل یکسانی دارد، برای یک روز مشخص کوئری را اجرا کرده‌ایم و قابل تعمیم به هر بازه‌ی زمانی دیگری هم هست.

در اینجا داده‌ها را بر روی فیلد `category` گروه کرده‌ایم و در قسمت `where` شرط را برابر با این قرار دادیم که فیلد `timestamp` بین دو مشخص شده باشد. و در نهایت هم آن‌ها را بر اساس تعداد هر گروه مرتب کرده‌ایم. کوئری به شکل زیر است:

```
select category, count(*) as count_ from kafka."default".news_statistics where
"@timestamp" between timestamp '2022-06-23 00:00' and timestamp '2022-06-23 23:00'
group by category order by count_ desc ;
```

و نتیجه‌ی آن هم در عکس زیر مشخص است:

category	count_
اخبار ارزهای دیجیتال	52
اخبار خودرو	40
اخبار مسکن	36
گزارش بازارها	34
اخبار تولید و تجارت	25
گزارش بازارها	1

برای ماه گذشته و یا هفته‌ی گذشته هم با عوض کردن مقادیر داده‌شده، بدون کاسته شدن از عمومیت راه حل می‌توان به جواب رسید.

#### - کوئری دوم:

ابتدا برای کیس روزهای هفته راه حل را توضیح می‌دهیم و بعد برای روز ماه هم قابل تعمیم است.

برای به دست آوردن اهمیت روزهای هفته، باید ببینیم که به طور متوسط در هر روز از هفته چند خبر منتشر می‌شود. برای این کار، روز هفته، سال و روز سال را مشخص می‌کنیم تا داده‌ها را بر اساس آن‌ها گروه کنیم. چرا که اگر تنها بر اساس روز هفته گروه کنیم، امکان اینکه میانگین انتشار خبر در هر روز را به دست بیاوریم نداریم. برای به دست آوردن این ویژگی‌ها هم از تابع `format_datetime` استفاده می‌کنیم و سپس تعداد خبر منتشر شده در هر «روز» را (که در داده اسم روز هفته‌ی آن را هم داریم) استخراج می‌کنیم. بعد از

آن که این نتیجه را به عنوان یک جدول موقت در نظر گرفتیم، دوباره داده‌ها را بر روی تنها روز هفته گروه می‌کنیم و میانگین را هم به راحتی به دست می‌آوریم. کوئری زیر این کارها را انجام می‌دهد.

```
select week_day, (sum(count_) / count(*)) as avg from
(
select count(*) as count_, week_day, year_, day_of_year from
(
select format_datetime("@timestamp", 'E') as week_day,
format_datetime("@timestamp", 'Y') as year_,
format_datetime("@timestamp", 'D') as day_of_year
from kafka."default".news_statistics
)
group by (week_day, year_, day_of_year)
) group by week_day order by avg desc;
```

نتیجه‌ی این کوئری هم به این شکل است:

week_day	avg
Thu	190
Fri	144
Wed	123
Sat	108
Tue	45
Mon	39

دقیقاً همین کوئری را می‌توانیم برای به دست آوردن اهمیت روزهای ماه انجام بدهیم و باید تنها به جای روز هفته، روز ماه را جایگزین کنیم. نتیجه‌ی کوئری برای روزهای ماه هم به این صورت است:

month_day	avg
24	259
22	215
23	190
11	108
21	45
20	39
16	32
10	30

- کوئری سوم:

برای انجام این کوئری، ابتدا با استفاده از `with clause` یک جدول موقت با نام `t` ایجاد می‌کنیم که مقدار چارک اول را با استفاده از تابع `approx_percentile` بر روی جدولی که برای هر رکورد، تعداد تکرار کلمه‌ی مورد نظر در متن را گزارش کرده است، به دست می‌آورد. همچنین تعداد کلمات را هم با استفاده از محاسبه‌ی `cardinality` تابع `regexp_extract_all` محاسبه می‌کند.

در ادامه، در یک سری timestamp یک جدول موقت تعریف می‌کنیم که برای هر رکورد، `nested-query` تعداد تکرار کلمه‌ی مورد نظر و چارک اول را با استفاده از `join` با `t` به جدول اضافه کردیم) و سپس سطرهایی که تعداد تکرار آن‌ها از چارک اول بیشتر است را فیلتر می‌کنیم و عدد هفته‌ی سال را از فیلد `group by` آن با استفاده از تابع `format_datetime` استخراج می‌کنیم. در نهایت بر روی عدد هفته timestamp می‌کنیم و تعداد را برای هر هفته، محاسبه می‌کنیم. کوئری به این صورت است:

```
with t as (
  select approx_percentile(count_, 0.25) as first_quartile from (
    select cardinality(regexp_extract_all(text, 'ارز')) as count_ from kafka."default".news_statistics
  )
  select week, count(*) as count_ from(
    select format_datetime("@timestamp", 'w') as week from
    (
      select "@timestamp", cardinality(regexp_extract_all(text, 'ارز')) as count_, first_quartile as quartile
      from kafka."default".news_statistics join t on true
    )
    where count_ > quartile
  )
  group by week;
```

و نتایج برای کلمات مختلف، در زیر گزارش شده است.

week	count_	RBC_col2
23	44	ارز
25	547	ارز
11	11	ارز

week	count_	RBC_col2
25	473	دلار
23	24	دلار
11	11	دلار

week	count_	RBC_col2
25	438	طللا
23	54	طللا
11	11	طللا

اما ظاهرا کلمات ارز دیجیتال و بیت کوین در متن‌ها وجود نداشتند. همچنین می‌توانیم دقیقا همان کوئری را با جابجایی هفته‌ی سال با روز سال، به تفکیک برای هر روز محاسبه کنیم.

نتایج برای روزها هم در زیر آورده شده است:

day_	count_	RBC_col2
162	19	دلار
161	5	دلار
172	45	دلار
173	172	دلار
174	84	دلار
171	39	دلار
75	11	دلار
175	133	دلار

day_	count_	RBC_col2
173	203	ارز
75	11	ارز
171	39	ارز
161	25	ارز
172	45	ارز
162	19	ارز
175	141	ارز
174	119	ارز

day_	count_	RBC_col2
171	39	طلا
162	42	طلا
173	201	طلا
172	45	طلا
161	12	طلا
174	93	طلا
175	60	طلا
75	11	طلا

#### - کوئری چهارم:

برای این کوئری به سادگی با گروه کردن بر روی نماد رمزارزها می‌توانیم تمامی موارد خواسته شده را استخراج کنیم. تنها برای مشخص کردن بازه‌ی زمانی، باید مجدداً مانند کوئری اول، شرط اینکه `filed` @timestamp بین دو مقدار مشخص شده باشد را چک کنیم. و باز هم از آن جایی که ما داده‌ی کافی برای چک کردن روزهای خواسته شده نداشتیم، بدون کاسته شدن از عمومیت راه حل، این کوئری را برای یک روز اجرا کردیم. کوئری مذکور در زیر آورده شده است.

```
select symbol, min(price_irr) as min_price, max(price_irr) as max_price, avg(price_irr) as mean_price
from kafka."default".crypto_statistics
where "@timestamp" between timestamp '2022-06-23 00:00' and timestamp '2022-06-23 23:00'
group by symbol;
```

و بخشی از نتیجه‌ی این کوئری هم به این صورت است:

symbol	min_price	max_price	mean_price
BUSD	325,000	326,400	325,714.2857142857
DOT	2,472,200	2,503,500	2,487,685.714285714
LTC	17,773,200	18,143,000	17,920,385.714285713
XCN	28,500	28,700	28,585.7142857143
MIOTA	94,700	100,000	97,600
NEO	2,985,000	3,012,100	2,999,671.4285714286
TUSD	325,100	326,200	325,520