Loading Musicbrainz in Elasticsearch

I stumpled upon the INSERT INTO LOGSTASH SELECT DATA FROM DATABASE article and decided to play around.

Installing PostgreSQL and loading the database and the data into Elasticsearch was just as described; the manual for MBSlave is very good.

Refer to the https://musicbrainz.org/doc/MusicBrainz_Database for more

I decided to use elkserver3 and a new logstash on that one, to avoid messing elkserver1 up.

Is possible to have a different Logstash config on each server in the cluster. Also - I decided to name the Index "musicbrainz-%{+YYYY.MM.dd}" to have some control over the load and a possible cleanup afterwards.

My Input file:

10-musicbrainz.conf

```
input {
    jdbc {
        jdbc_driver_library => "/etc/logstash/postgresql-9.4.1212.jre6.jar"
        jdbc_driver_class => "org.postgresql.Driver"
        jdbc_connection_string => "jdbc:postgresql://localhost:5432/musicbrainz?
user=musicbrainz&password=*******"
        jdbc_user => "musicbrainz"
        statement_filepath => "/etc/logstash/query.sql"
        schedule => "0 15 * * *"
    }
}
```

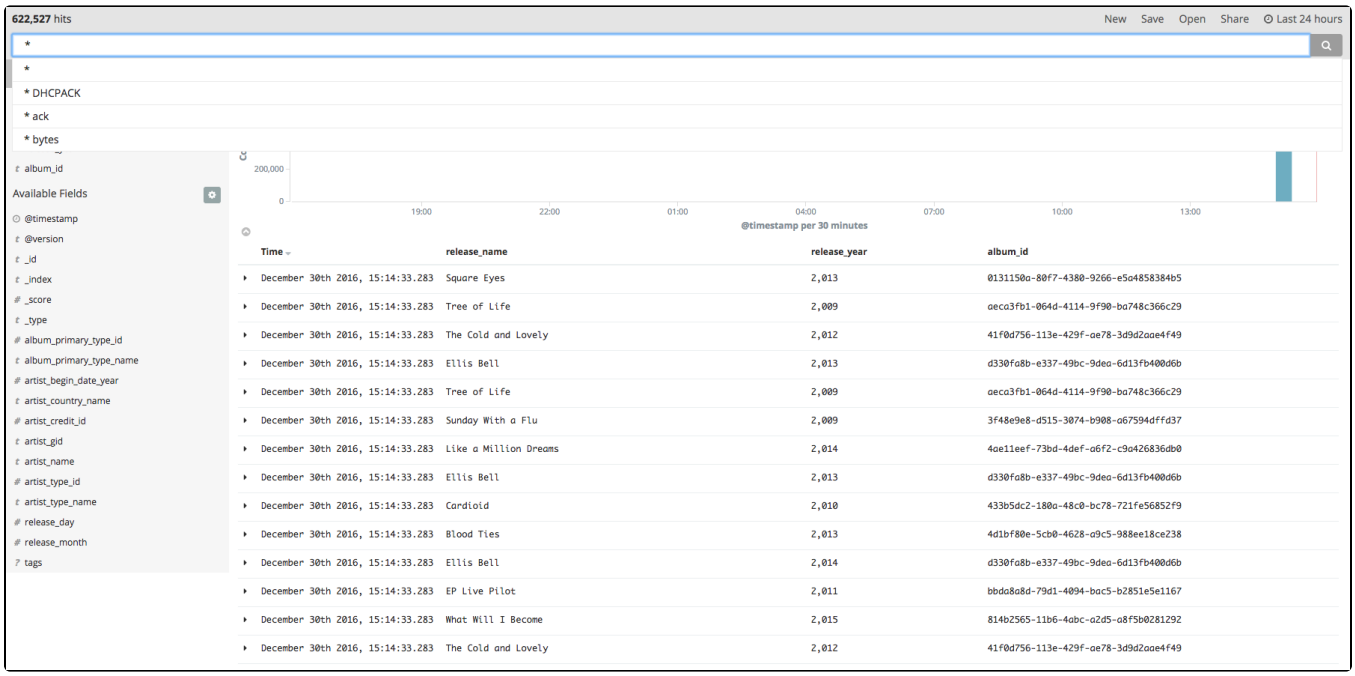
Notice the Schedule - the Query runs one time each day at 15:00 - but the data is static, so its not nessesary....But the schedule makes sure I know when the Query is runned one time (only)

My Output file:

```
output {

  elasticsearch
  {
    hosts => "localhost:9200"
    sniffing => false
    manage_template => false
    index => "musicbrainz-%{+YYYY.MM.dd}"
  }
}
```

The load gave this in Kibana:



And verifying the row count in PostgreSQL with:

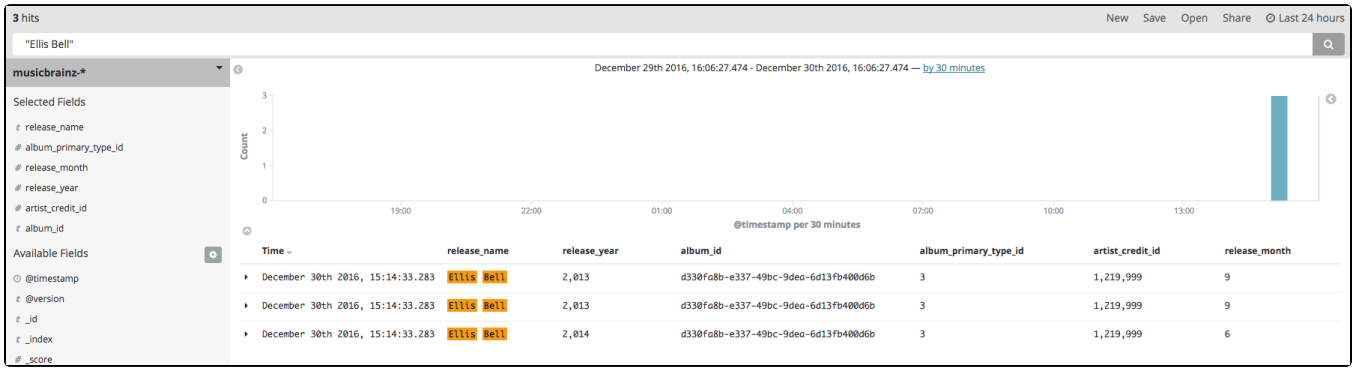
```
SELECT count(*) AS Dummy FROM (
SELECT
    release_group.gid AS album_id,
    release_group.type AS album_primary_type_id,
    release_group_primary_type.name AS album_primary_type_name,
    release.name AS release_name,
    artist.name AS artist_name,
    artist.gid AS artist_gid,
    artist_credit.id AS artist_credit_id,
    artist.type AS artist_type_id,
    artist_type.name AS artist_type_name,
    artist.begin_date_year artist_begin_date_year,
    area.name AS artist_country_name,
    release_country.date_year AS release_year,
    release_country.date_month AS release_month,
    release_country.date_day AS release_day
FROM
    musicbrainz.artist
INNER JOIN musicbrainz.artist_credit_name
    ON artist_credit_name.artist = artist.id
INNER JOIN musicbrainz.artist_credit
    ON artist_credit.id = artist_credit_name.artist_credit
INNER JOIN musicbrainz.release_group
    ON release_group.artist_credit = artist_credit.id
INNER JOIN musicbrainz.release
    ON release.release_group = release_group.id
INNER JOIN musicbrainz.release_country
    ON release.id = release_country.release
INNER JOIN musicbrainz.artist_type
    ON artist.type = artist_type.id
INNER JOIN musicbrainz.area
    ON artist.area = area.id
INNER JOIN musicbrainz.release_group_primary_type
    ON release_group_primary_type.id = release_group.type
WHERE
    ((release_country.date_year IS NOT NULL) AND
    (release_country.date_month IS NOT NULL) AND
    (release_country.date_day IS NOT NULL))
    ) As Dummy2
```

Gave:

```
 dummy
--------
 622527
(1 row)
```

Success - same row count 🙂

I do notice that some rows seems to be the same:



Or not? The "album_id" is the same, but in one row the "release_year" differs from the two others....

Running the SQL

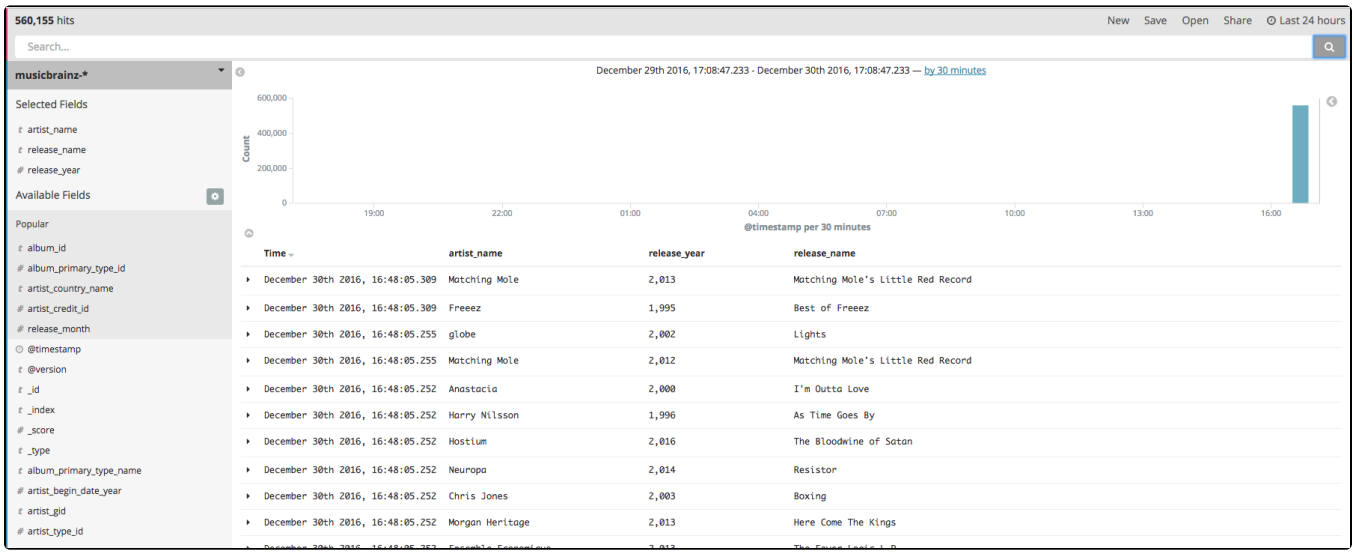
```
SELECT Distinct * FROM (
SELECT
    release_group.gid AS album_id,
    release_group.type AS album_primary_type_id,
    release_group_primary_type.name AS album_primary_type_name,
    release.name AS release_name,
    artist.name AS artist_name,
    artist.gid AS artist_gid,
    artist_credit.id AS artist_credit_id,
    artist.type AS artist_type_id,
    artist_type.name AS artist_type_name,
    artist.begin_date_year artist_begin_date_year,
    area.name AS artist_country_name,
    release_country.date_year AS release_year,
    release_country.date_month AS release_month,
    release_country.date_day AS release_day
FROM
    musicbrainz.artist
INNER JOIN musicbrainz.artist_credit_name
    ON artist_credit_name.artist = artist.id
INNER JOIN musicbrainz.artist_credit
    ON artist_credit.id = artist_credit_name.artist_credit
INNER JOIN musicbrainz.release_group
    ON release_group.artist_credit = artist_credit.id
INNER JOIN musicbrainz.release
    ON release.release_group = release_group.id
INNER JOIN musicbrainz.release_country
    ON release.id = release_country.release
INNER JOIN musicbrainz.artist_type
    ON artist.type = artist_type.id
INNER JOIN musicbrainz.area
    ON artist.area = area.id
INNER JOIN musicbrainz.release_group_primary_type
    ON release_group_primary_type.id = release_group.type
WHERE
    ((release_country.date_year IS NOT NULL) AND
    (release_country.date_month IS NOT NULL) AND
    (release_country.date_day IS NOT NULL))
    ) As Dummy2
```

Gave

```
560155 rows
```

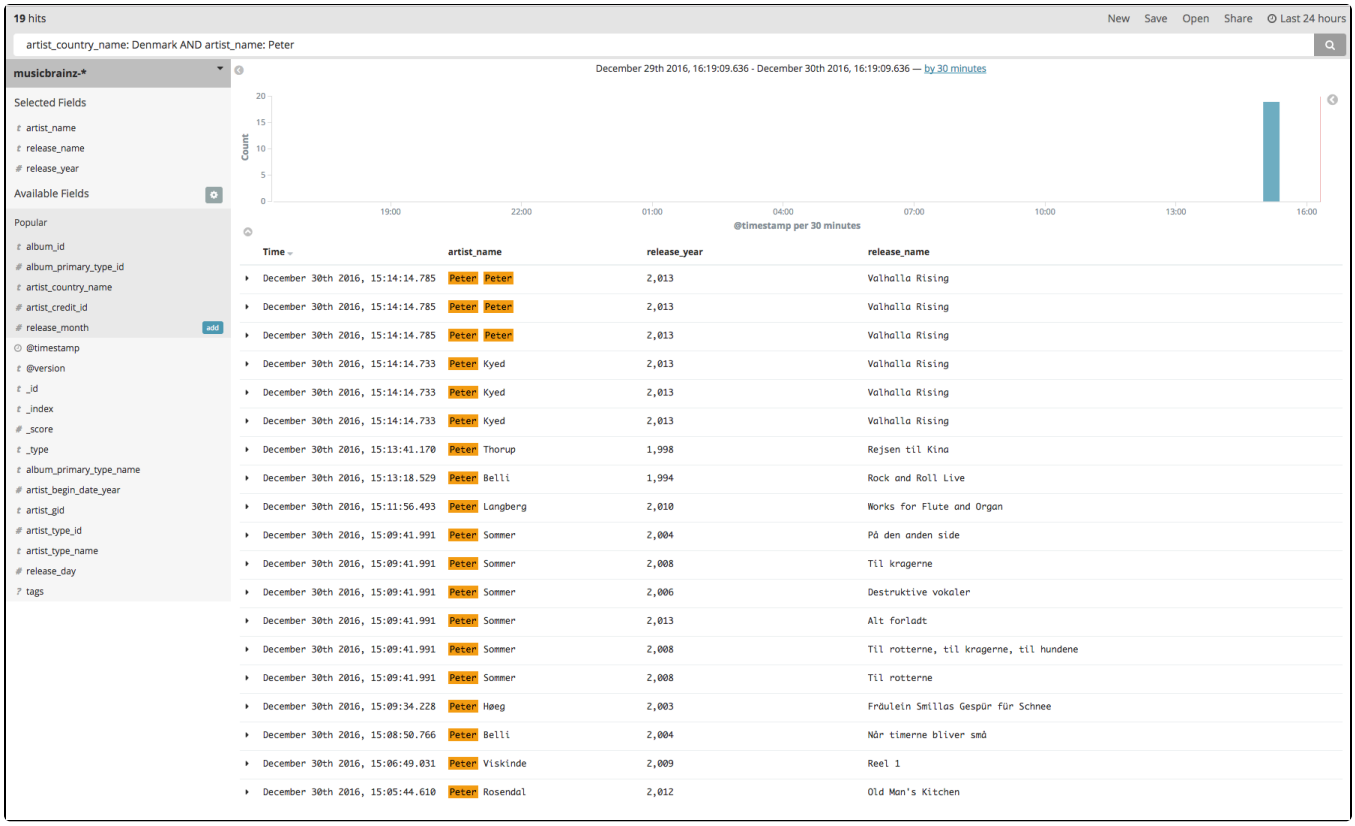
So, there is a possible redundancy in the SQL provided from https://www.elastic.co/blog/logstash-jdbc-input-plugin

Reloading the data (after deleting the Index) gives:

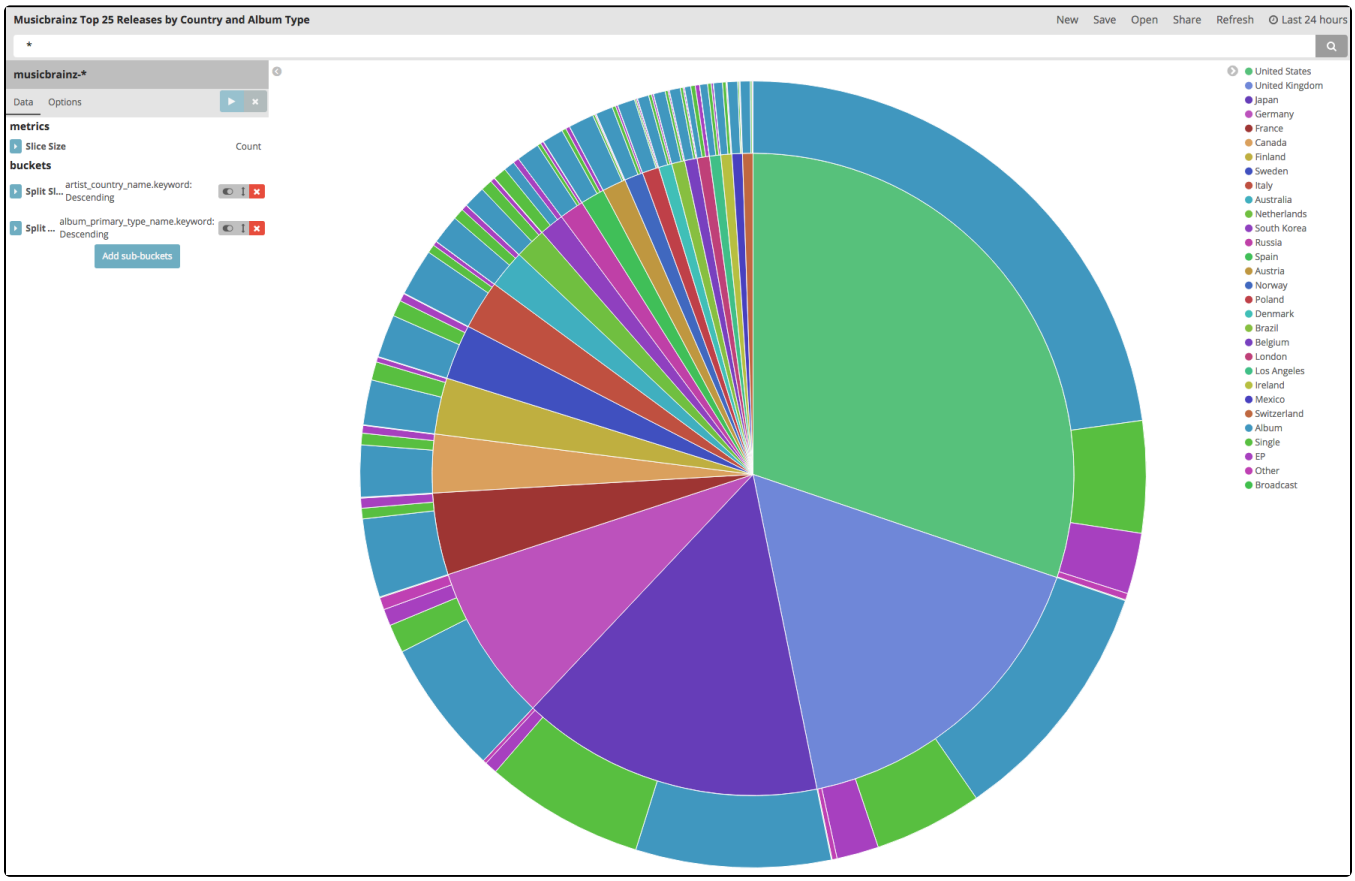


Samples

Here is a sample of all albums from "Denmark" with "Peter" in the Artist name:



A few Visualizations..



Whats Next..

Well, this is unfinished business...there are so much more data to combine.....

And another project could be parsing IMDB data ... http://www.imdb.com/interfaces