

Group Details.....	1
Introduction and motivation for the app and the choice of the technologies.....	2
Related work – any related systems/application - Top 3.....	3
How does our approach differ?.....	4
Description of the dataset.....	4
a. Source of data.....	4
Description of data processing.....	5
Development of the application platform.....	7
Challenges and lessons learned.....	10
Challenge 1: Understanding the schema before the process.....	10
Challenge 2 : CSV is not big data friendly format, Parquet to the rescue.....	10
Challenge 3 : GCP dataflow template are still in the beta phase.....	10
Challenge 4 : EMR hadoop operations are hard to perform on CLI, Apache Hue to our rescue.....	11
Our take : Elasticsearch dashboard is very powerful.....	11
Responsibility section (which group member did what?).....	11
References.....	12
Screenshots.....	12
Creating AWS S3 clouformation.....	12
Athena operations.....	13
Glue ETL for CSV to Parquet.....	20
EMR Cluster configs and settings.....	22
Apache Hue for Hadoop operations UI.....	23
Export Final Report.....	26
pyspark operations.....	28
GCS integrations dataflow jobs.....	30
Elasticsearch cloud configs and final kibana dashboard.....	32

Details

Dataset	Music Brainz dataset https://musicbrainz.org/doc/MusicBrainz_Database
Dataset description	This data includes information about artists, release groups, releases, recordings, works, and labels, as well as the many relationships between them. The database also contains a

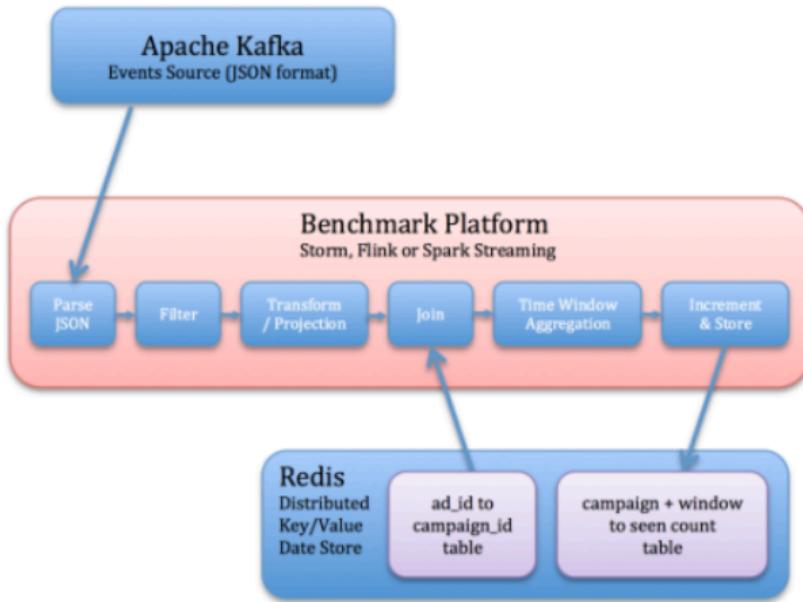
	full history of all the changes that the MusicBrainz community has made to the data
--	---

Introduction and motivation for the app and the choice of the technologies

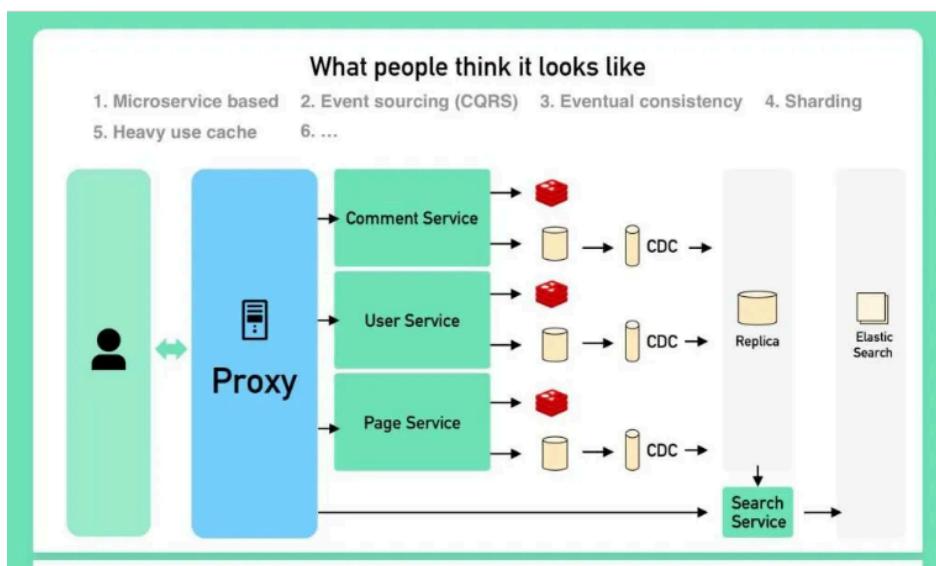
- So here for the initial steps, we identified the yardsticks for the challenge
 - a. 1.4 millions music records storage, processing and analysing
 - b. Selecting the right cloud service considering the free trials and ease of access within the limited credits
 - c. Right database to process and analyse the big data
 - d. Right distributed service for performing data visualisation and should be able to store and should readily be available for integrations
 - e. Good dev community, in case we stumble upon any issues for us to ask and take the references
- Considering the all the above mentioned yardsticks, our choice has become clearer to opt for
 - a. AWS S3, GCP bucket for initial and post storage options
 - b. Preprocessing and for major extract, transform tasks
 - Athena/Glue for initial processing of schema to understand the nature of data
 - pySpark, for cleaning and transforming the data
 - c. HDFS/Hadoop EMR cluster, for big data storage
 - d. Apache HIVE database, for storing, processing and analysing the dataset
 - e. Apache HUE UI, for HIVE, S3, PIG interactions through relatively good user interface rather than overly relying on the CLI tools
 - f. Elasticsearch Cloud/Kibana dashboard and integration tools for creating charts, visualisations and reports, hosted on GCP
 - g. Python and linux commands to perform miscellaneous tasks

Related work – any related systems/application - Top 3

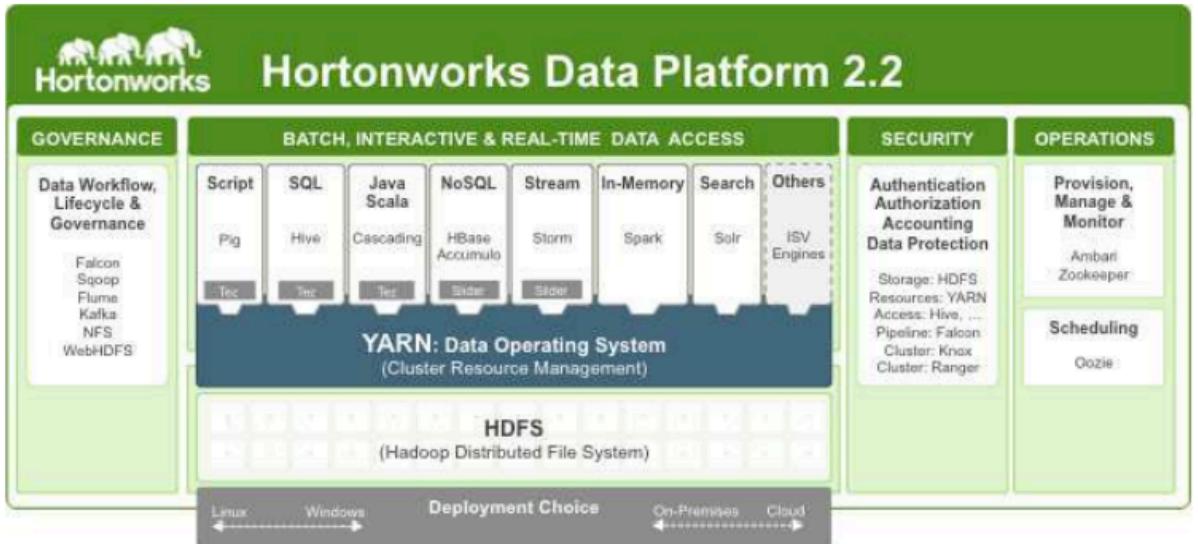
1. Started off with Yahoo architecture and how they have managed to process big data and their architecture around micro batch processing using Apache Kafka and Spark [2]



- So we took some of the reference from the stackoverflow[1] architecture design where it leveraged the ELK stack sitting on top of monolithic design, we helped them tackle their core issues of storage with eventual consistency and faster search/analytics through ELK (elasticsearch, logstash and kibana) stack.[1] [link](#)



3. Hortonworks selection of big data architecture, their approach for HDFS interactions [3]



How does our approach differ?

Above mentioned applications are designed to handle scalability with production ready environments, while we had to design the data flow architecture considering the cloud service which are available at our fingertips with minimal to no cost, so with that in mind.

Data -> AWS S3 -> Athena -> Glue -> EMR -> HIVE and Pyspark -> S3 -> Elastic Cloud service (Elasticsearch/Kibana)

With this data flow, we were able to achieve the ultimate goal of creating a good data analytics application with future scope of creating data ingestion pipelines, python endpoints/lambda triggers and handling big data processing with good horizontal scalability.

Description of the dataset

a. Source of data

Music Brainz dataset, https://musicbrainz.org/doc/MusicBrainz_Database

b. Process of extraction/collection

Download the musicbrainz data and upload it to S3 using AWS CLI

```
# to move the files to s3, TODO : use python boto3
aws s3 cp "C:\Users\patel\Downloads\artists.csv" s3://music-5bb01e20
```

c. Data preprocessing – preparation and cleaning

For the preprocessing and preparation, we have selected **AWS Athena** queries after creating the database, table and load the s3 file to the table

```

• CREATE EXTERNAL TABLE IF NOT EXISTS
  `music_brains`.`music_artists_all` (
    `mbid` string,
    `artist_mb` string,
    `artist_lastfm` string,
    `country_mb` string,
    `country_lastfm` string,
    `tags_mb` string,
    `tags_lastfm` string,
    `listeners_lastfm` string,
    `scrobbles_lastfm` string,
    `ambiguous_artist` string
  )
• ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  WITH SERDEPROPERTIES (
    'separatorChar' = '\t',
    'quoteChar' = '\"',
    'escapeChar' = '\\'
  )
• STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
  OUTPUTFORMAT
    'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
• LOCATION 's3://music-951f64e0/csv/'
• TBLPROPERTIES ('classification' = 'csv');

•
•
•
  SELECT * FROM "music_brains"."music_artists_all" limit 10;
•
  SELECT count(*) FROM "music_brains"."music_artists_all";
•
  SELECT count(*) FROM "music_brains"."music_artists_all" where
  ambiguous_artist = 'TRUE';

```

Description of data processing

In two phases,

Phase 1, using Glue to convert it to parquet format, create HIVE tables and fetch the relevant data then export it to S3 [link to gitlab glue file](#)

```

Create database music;

CREATE EXTERNAL TABLE IF NOT EXISTS `music`.`artists` (
`mbid` string,
`artist_mb` string,
`artist_lastfm` string,
`country_mb` string,
`country_lastfm` string,
`tags_mb` string,
`tags_lastfm` string,
`listeners_lastfm` string,
`scrobbles_lastfm` string,
`ambiguous_artist` string
)
STORED AS PARQUET
LOCATION 's3://music-d6cb4270/parquet';

--## country/artists count

select * from artists where mb_id in (
select mb_ib, artist_lastfm, count(*) from artists
where ambiguous_artist = 'FALSE'
group by mb_ib, artist_lastfm
having count(*) > 10) grouped;

--## cleaned

select * from artists where
ambiguous_artist is not null
and country_lastfm is not null
and country_mb is not null
and artist_mb is not null
and artist_lastfm is not null
and tags_mb is not null
and tags_lastfm is not null
and mbid != 'mbid';

--#### most listeners

Select
artist_lastfm artist_name,
country_mb country_origin,
split(country_lastfm, ';') country_list,

```

```

cast(listeners_lastfm as int) listeners_cnt,
cast(scrobbles_lastfm as int) scrobbles_cnt,
split(tags_mb,';') tags
from artists
where mbid != 'mbid'
and country_lastfm != ''
and artist_mb != ''
order by listeners_cnt desc

hive -e 'Select
artist_lastfm artist_name,
country_mb country_origin,
split(country_lastfm,';') country_list,
cast(listeners_lastfm as int) listeners_cnt,
cast(scrobbles_lastfm as int) scrobbles_cnt,
split(tags_mb,';') tags
from artists
where mbid != 'mbid'
and country_lastfm != ''
and artist_mb != ''
order by listeners_cnt desc' | sed 's/[\\t]//,/g' > /final.csv

```

Phase 2, pyspark, using google collab and perform RDD, SparkSQL operations then make the final file ready for the consumption, this step is just to eliminate all possibilities to surface the refined data, as we have observed pyspark has more support and an excellent developer community than Apache PIG [pyspark notebook](#)

Development of the application platform

Cloud platform we have used to achieve this step is GCP and that helped us integrate with Elastic cloud service

Elastic cloud comes with its integrations agent and GCP dataflow pipelines, however dataflow template pipelines are in beta phase and fails randomly at times, so we have leveraged Elastic cloud data integrations to connect our S3 files and create index using following mappings

```
{
  "known_artists": {
    "mappings": {
      "properties": {
        "name": {
          "type": "string"
        }
      }
    }
  }
}
```

```

    "ambiguous_artist": {
        "type": "keyword"
    },
    "artist_lastfm": {
        "type": "keyword"
    },
    "artist_mb": {
        "type": "keyword"
    },
    "country_lastfm": {
        "type": "keyword"
    },
    "country_mb": {
        "type": "keyword"
    },
    "listeners_lastfm": {
        "type": "long"
    },
    "mbid": {
        "type": "keyword"
    },
    "scrobbles_lastfm": {
        "type": "long"
    },
    "tags_lastfm": {
        "type": "text"
    },
    "tags_mb": {
        "type": "text"
    }
}
}

#### cluster setting, replica/master nodes
#### text fields are considered analysed by default for performing
### NLP operations

{
    "known_artists": {
        "settings": {
            "index": {
                "routing": {

```

```

    "allocation": {
      "include": {
        "_tier_preference": "data_content"
      }
    }
  },
  "number_of_shards": "1",
  "provided_name": "known_artists",
  "creation_date": "1702911573820",
  "number_of_replicas": "1",
  "uuid": "G7ni20CIRmidn8pnY14Q2A",
  "version": {
    "created": "8500003"
  }
}
}
}

```

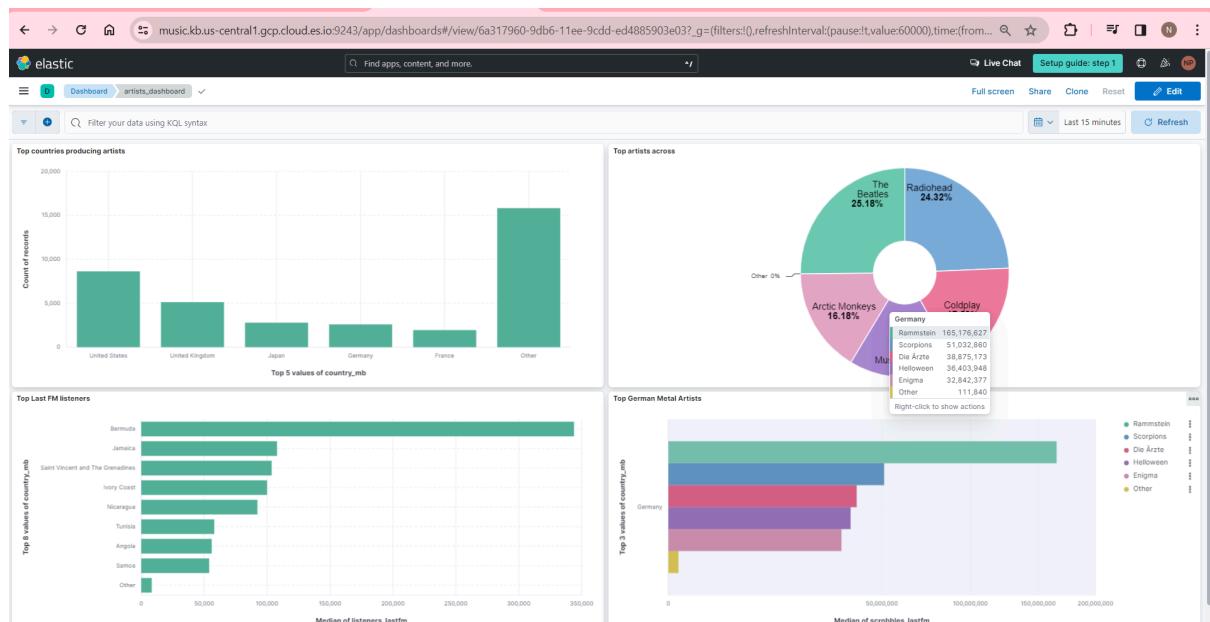
Once we successfully integrate the data the data flow would look like this,

The screenshot shows the Elasticsearch interface with the following details:

- URL:** music.kb.us-central1.gcp.cloud.es.io:9243/app/discover#/?_g=(filters:[],refreshInterval:(pause:1,value:60000),time:(from:now-15m,to:now))&_a=(columns:[],filters:[],index:...)
- Panel:** elastic
- Search Bar:** Find apps, content, and more.
- Header Buttons:** Live Chat, Setup guide: step 1, Save, Refresh.
- Left Sidebar:**
 - final-artists
 - Discover
 - Available fields:
 - artist_name
 - country_list
 - country_origin
 - listeners_cnt
 - scratches_cnt
 - tags
 - Empty fields
 - Meta fields:
 - _id
 - _index
 - _score
- Document View:** Shows a table with columns: artist_name, country_list, listeners_cnt, scratches_cnt, tags, _id, _index, _score.
- Results:** A list of artists with their counts and scores. Some entries are collapsed with a '...' button.

Then we can go to the visualization, and start creating the charts to answer the top artists, top artists producing countries and the top german music artists based on their listeners last.fm numbers

Final visualisation looks like this; and this helps us answer who is the most famous artist in Germany is Rammstein!



Challenges and lessons learned

Challenge 1: Understanding the schema before the process

So we have understood that, schema detection before the process phase is crucial, that's where we had to use the Athena which internally uses the HIVE and helped us figure out the schema, Glue crawler was also useful for schema detection

Challenge 2 : CSV is not big data friendly format, Parquet to the rescue

Another challenge, we faced was performing all the operations in CSV, first it consumed space, time and the due to no information on the de-limiter, we had to perform trial and error just to understand the delimiter with escaping sequence to load, then we understood instead of loading in CSV, parquet and avro seems to be better formats dealing with big data especially in apache tools and technology and that helped us load the data in HIVE easily

Challenge 3 : GCP dataflow template are still in the beta phase

GCP dataflow has defined templates, however to ingest the data into elastic cloud the template was in beta phase and fails randomly, so as fallback options we had to spin the elastic cloud integration agents to get through the ingestion of AWS S3 files to elasticsearch index, however AWS is again the sandbox environment which made things worse in the sync process.

Challenge 4 : EMR hadoop operations are hard to perform on CLI, Apache Hue to our rescue

Since hive, hdfs files, pig and spark streaming, it was a lot to go back and forth so we had the need for the user friendly UI to perform our tasks, stores the queries, visualise the operations and maintain the state, so we had performed SSH tunnelling using proxyswitch omega chrome extension and that ease our burden and helped us immensely in performing processing, hadoop operations.

The screenshot shows the Apache Hue web interface. On the left is a sidebar with various database and tool icons: MySQL, Hive, PostgreSQL, SparkSQL, Notebook, Scala, PySpark, Spark Submit Jar, Spark Submit Python, Text, Markdown, SQLite, Pig, Java, and Spark. The main area has tabs for Editor, Query History, Saved Queries, and Results (100+). The Results tab displays a table with columns: artist_name, country_origin, country_list, listeners_cnt, and scro. The table lists nine rows of data. To the right of the table is a sidebar titled 'Tables' with a list of tables: music.artists, mbid, artist_mb, artist_lastfm, country_mb, country_lastfm, tags_mb, tags_lastfm, listeners_lastfm, scrobbles_lastfm, and ambiguous_artist. At the top of the interface, there is a status bar indicating 'Not secure' and the URL 'ec2-44-221-63-224.compute-1.amazonaws.com:8888/hue/editor?editor=2&#lid=application_1702999451997_0007'.

Our take : Elasticsearch dashboard is very powerful

Elasticsearch we understood for the search engineer built on lucene, however the Kibana dashboard seems to be very smooth in fetch the categorical data and intuitive enough to suggest the user which charts and analysis to go for;

Plus, its integration supports AWS, GCP, AZURE . It seems to be a worthy inclusion for all the log, data and NLP operations and high end analytics.

Responsibility section (which group member did what?)

Name	Task	Notes
Vishnu Priya Gandewar	<ul style="list-style-type: none">• Dataset cleaning• Preprocessing with Athena• PySpark operations• Schema creations• Glue operations to transform to parquet	
Nirav Patel	<ul style="list-style-type: none">• EMR creations• Hive Queries• SSH HUE tunnelling• Elasticsearch cloud setup, GCP integrations, dataflow testing	

- | | | |
|--|---|--|
| | <ul style="list-style-type: none"> • Index, replica, cluster settings and dashboard creations • S3 integrations and GCP bigquery, dataflow analysis | |
|--|---|--|

References

1. Theresa (2022) *EP27: Stack overflow architecture. also..., EP27: Stack Overflow Architecture. Also... - by Theresa*. Available at: <https://blog.bytebytogo.com/p/ep27-stack-overflow-architecture> (Accessed: 17 December 2023).
2. Yahoo *et al.* (no date) *Yahoo Engineering*. Available at: <https://yahooenrg.tumblr.com/page/6> (Accessed: 17 December 2023).
3. *Big Data Reference Architecture - Cloudera*. Available at: https://hortonworks.com/wp-content/uploads/2013/10/Big-Data-Reference-Architecture_4AA5-6136ENW.pdf (Accessed: 17 December 2023).

Screenshots

Creating AWS S3 cloudformation

The screenshot shows the AWS CloudFormation console interface. On the left, the 'Stacks' list shows several stacks, with 's32' selected. The main pane displays the details for 's32'. At the top right, there are buttons for 'Delete', 'Update', 'Stack actions', and 'Create stack'. Below these are tabs for 'Stack info', 'Events' (which is selected), 'Resources', 'Outputs', 'Parameters', 'Template', 'Change sets', and 'Git sync - new'. The 'Events' tab shows a table with 9 rows of event logs. The columns are 'Timestamp', 'Logical ID', 'Status', and 'Status reason'. The events are all in progress, mostly for creating S3 buckets (S3Bucket2, S3Bucket3, S3Bucket, S3Bucket4). One event for 's32' is listed as 'User Initiated'.

Timestamp	Logical ID	Status	Status reason
2023-12-19 14:40:49 UTC+0000	S3Bucket2	CREATE_IN_PROGRESS	Resource creation Initiated
2023-12-19 14:40:49 UTC+0000	S3Bucket3	CREATE_IN_PROGRESS	Resource creation Initiated
2023-12-19 14:40:48 UTC+0000	S3Bucket	CREATE_IN_PROGRESS	Resource creation Initiated
2023-12-19 14:40:48 UTC+0000	S3Bucket4	CREATE_IN_PROGRESS	Resource creation Initiated
2023-12-19 14:40:44 UTC+0000	S3Bucket	CREATE_IN_PROGRESS	-
2023-12-19 14:40:44 UTC+0000	S3Bucket2	CREATE_IN_PROGRESS	-
2023-12-19 14:40:44 UTC+0000	S3Bucket3	CREATE_IN_PROGRESS	-
2023-12-19 14:40:44 UTC+0000	S3Bucket4	CREATE_IN_PROGRESS	-
2023-12-19 14:40:41 UTC+0000	s32	CREATE_IN_PROGRESS	User Initiated

s3.console.aws.amazon.com/s3/home?region=us-east-1

Search [Alt+S] Global v oclabs/user2798498=Nirav_Patel @ 3585-5587-36

You can enable advanced metrics in the "default-account-dashboard" configuration.

General purpose buckets Directory buckets

General purpose buckets (5) Info Buckets are containers for data stored in S3. Learn more

C Copy ARN Empty Delete Create bucket

Find buckets by name < 1 > ⌂

Name	AWS Region	Access	Creation date
cf-templates-50e2arqy0ab1-us-east-1	US East (N. Virginia) us-east-1	Bucket and objects not public	December 19, 2023, 14:40:23 (UTC+00:00)
final-951f64e0	US East (N. Virginia) us-east-1	Bucket and objects not public	December 19, 2023, 14:40:53 (UTC+00:00)
logs-951f64e0	US East (N. Virginia) us-east-1	Bucket and objects not public	December 19, 2023, 14:40:53 (UTC+00:00)
music-951f64e0	US East (N. Virginia) us-east-1	Bucket and objects not public	December 19, 2023, 14:40:53 (UTC+00:00)
output-951f64e0	US East (N. Virginia) us-east-1	Bucket and objects not public	December 19, 2023, 14:40:53 (UTC+00:00)

Upload succeeded View details below.

The information below will no longer be available after you navigate away from this page.

Summary

Destination	Succeeded	Failed
s3://music-951f64e0/csv/	1 file, 191.8 MB (100.00%)	0 files, 0 B (0%)

Files and folders Configuration

Files and folders (1 Total, 191.8 MB)

Find by name < 1 >

Name	Folder	Type	Size	Status	Error
artists.csv	-	text/csv	191.8 MB	Succeeded	-

Athena operations

The screenshot shows the AWS Athena Query editor interface. On the left, there's a sidebar with 'Data' settings, including 'Data source' set to 'AwsDataCatalog' and 'Database' set to 'default'. Below this are sections for 'Tables and views' with a 'Create' button and a search bar, and 'Views (0)'.

The main area is titled 'Query 10' and contains the SQL command:

```
1 create database music_brains;
```

Below the query, it says 'SQL Ln 1, Col 30'. There are buttons for 'Run again' (highlighted in orange), 'Explain', 'Cancel', 'Clear', and 'Create'. At the bottom, under 'Query results', it shows 'Completed' and the message 'Query successful.'

AWS Services Search [Alt+S]

Amazon Athena > Query editor > Create table from S3 bucket data

Create table from S3 bucket data Info

Table details

Table name
music_artists_all

Table name must be from 1-128 characters and must be unique. Valid characters are a-z, A-Z, 0-9, _(underscore). Table names tend to correspond to the directory where the data will be stored.

Description - optional
Type something

Table description must be from 1-1024 characters. 1024 characters remaining.

Database configuration Info

Choose an existing database or create a new database
Choose to access an existing database or to create a new database in order to create a new table. Athena stores the table schema in the AWS Glue Data Catalog.

Create a database
 Choose an existing database

music_brains ▾

Dataset Info

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/create-table

aws Services Search [Alt+S]

s3://music-951f64e0/csv X View Browse S3

Input the path to the data set you want to process on Amazon S3. For example if your data is stored at s3://input-data-set/logs/1.csv, please enter s3://input-data-set/logs/. If your data is already partitioned, e.g. s3://input-data-set/logs/year=2004/month=12/day=11/ just input the base path s3://input-data-set/logs/

Data format Info

Table type: Apache Hive

File format: CSV

SerDe library: org.apache.hadoop.hive.serde2.OpenCSVSerde

SerDe properties - *optional*

Name	Value	Remove
separatorChar	\t	Remove
quoteChar	'	Remove
escapeChar	\\	Remove

Add SerDe property

Column details

Column name must be from 1-128 characters. Valid characters are a-z, A-Z, 0-9, _(underscore). Certain advanced column types (namely, structs) are not exposed in this interface.

aws Services Search [Alt+S]

Column details

Column name must be from 1-128 characters. Valid characters are a-z, A-Z, 0-9, _(underscore). Certain advanced column types (namely, structs) are not exposed in this interface.

Column name	Column type	Description - optional	
mbid	string	Enter description	Remove
artist_mb	string	Enter description	Remove
artist_lastfm	string	Enter description	Remove
country_mb	string	Enter description	Remove
country_lastfm	string	Enter description	Remove
tags_mb	string	Enter description	Remove
tags_lastfm	string	Enter description	Remove
listeners_lastfm	string	Enter description	Remove

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/620375d9-1e39-4ed2-9972

AWS Services Search [Alt+S]

Amazon Athena > Query editor

Editor Recent queries Saved queries Settings

Data Data source: AwsDataCatalog Database: music_brains

Tables and views Create Filter tables and views

Tables (1) music_artists_all

Views (0)

Query 10 : X | Query 11 : X

```
1 - CREATE EXTERNAL TABLE IF NOT EXISTS `music_brains`.`music_artists_all` (
2   `mbid` string,
3   `artist_mb` string,
4   `artist_lastfm` string,
5   `country_mb` string,
6   `country_lastfm` string,
7   `tags_mb` string,
8   `tags_lastfm` string,
9   `listeners_lastfm` string,
10  `scrobbles_lastfm` string,
11  `ambiguous_artist` string
12 )
13 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
14 WITH SERDEPROPERTIES (
15   'separatorChar' = '\t',

```

SQL Ln 5, Col 23

Run again Explain Cancel Clear Create

Query results | Query stats

Completed

Query successful.

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/34325685-844c-4e02-924c-32f8695a0eb9

AWS Services Search [Alt+S]

N. Virginia vclabs/user2798498-Nirav_Patel at 3585-55

Amazon Athena > Query editor

Editor Recent queries Saved queries Settings Workgroup primary

Data Data source: AwsDataCatalog Database: music_brains

Tables and views Create Filter tables and views

Tables (1) music_artists_all

Views (0)

Query 10 : X | Query 11 : X | Query 12 : X | Query 13 : X | Query 14 : X | Query 15 : X

```
1 SELECT * FROM "music_brains"."music_artists_all" limit 10;
```

SQL Ln 1, Col 1

Run again Explain Cancel Clear Create Reuse query results up to 60 minutes ago

Query results | Query stats

Completed Time in queue: 103 ms Run time: 557 ms Data scanned: 726.16 KB

Results (10)

Copy Download results

us-east-1.console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/34325685-844c-4e02-924c-32f8695a0eb9

Services Search [Alt+S] N. Virginia v vocabulary/user2798496-Nirav_Patel @ 3585-5587-3695

music_artists_all Views (0) SQL Ln 1, Col 1 Run again Explain Cancel Clear Create Reuse query results up to 60 minutes ago

Query results Query stats Completed Time in queue: 103 ms Run time: 557 ms Data scanned: 726.16 KB

Results (10) Copy Download results

#	mbid	artist_mb	artist_lastfm	country_mb	country_lastfm	tags_mb	tags_last
7	fc4a9c65-a108-435f-bbb2-a66944edd912	"animalesco	o método"	"Animalesco	O Método"		Portugal
1	cbd4e540-2271-456e-95ea-01a78d73b304	Andy Medley and James Warres	Andy Medley And James Warres				
2	843dad8b-457f-4255-9def-9f49ab005e63	Andy Yola	Andy Yola				
3	01abdc81-c090-4fe8-8b0a-a46cd7b273ed	Angelino Loren	Angelino Loren				
4	af918c05-7d50-4523-b69d-52c659505697	Anger Reign	Anger Reign	United States	thrash metal		
5	b337fd5-a985-45bb-82a8-3e4a93fe1d71	Angry Tradesmen	Angry Tradesmen	Australia			
6	2a26ff06-7a80-4ae8-b594-53c93de64440	Anieger	Anieger				
8	9921d209-9394-47e4-a775-751cf067e939	Anjali Muralidharan	Anjali Muralidharan				
9	b6dcde01-58bd-4e23-86fa-6c701977ae0e	Anke Helfrich Trio	Anke Helfrich Trio Feat. Roy Hargrove		jazz		
10	8c33bc57-4e79-463e-8054-a590638ace95	Ann Carlberger	Ann Carlberger				

Query 10 : X | Query 11 : X | Query 12 : X | Query 13 : X | Query 14 : X | Q

1 SELECT count(*) FROM "music_brains"."music_artists_all";

views

SQL Ln 1, Col 56 Run again Explain Cancel Clear Create

Query results Query stats Completed

Results (1)

#	_col0
1	1466084

Query 10 : X | Query 11 : X | Query 12 : X | Query 13 : X | Query 14 : X | **Query 15 :**

```
1 SELECT count(*) FROM "music_brains"."music_artists_all" where ambiguous_artist = 'TRUE';
```

SQL Ln 1, Col 89

Run again **Explain** **Cancel** **Clear** **Create ▾**

Query results **Query stats**

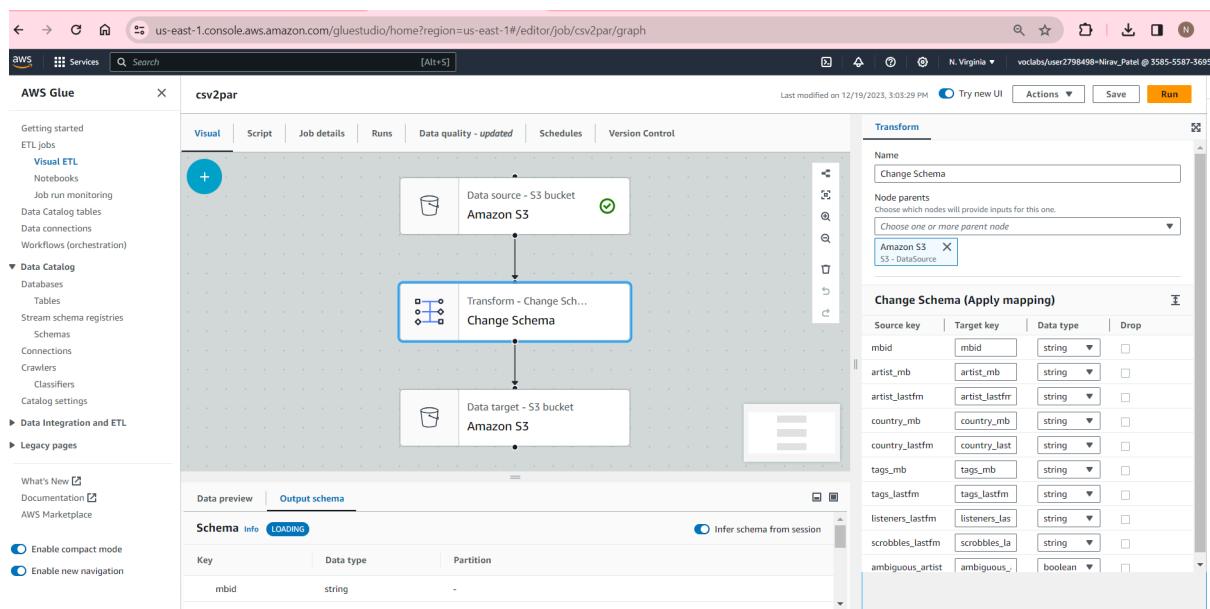
Completed

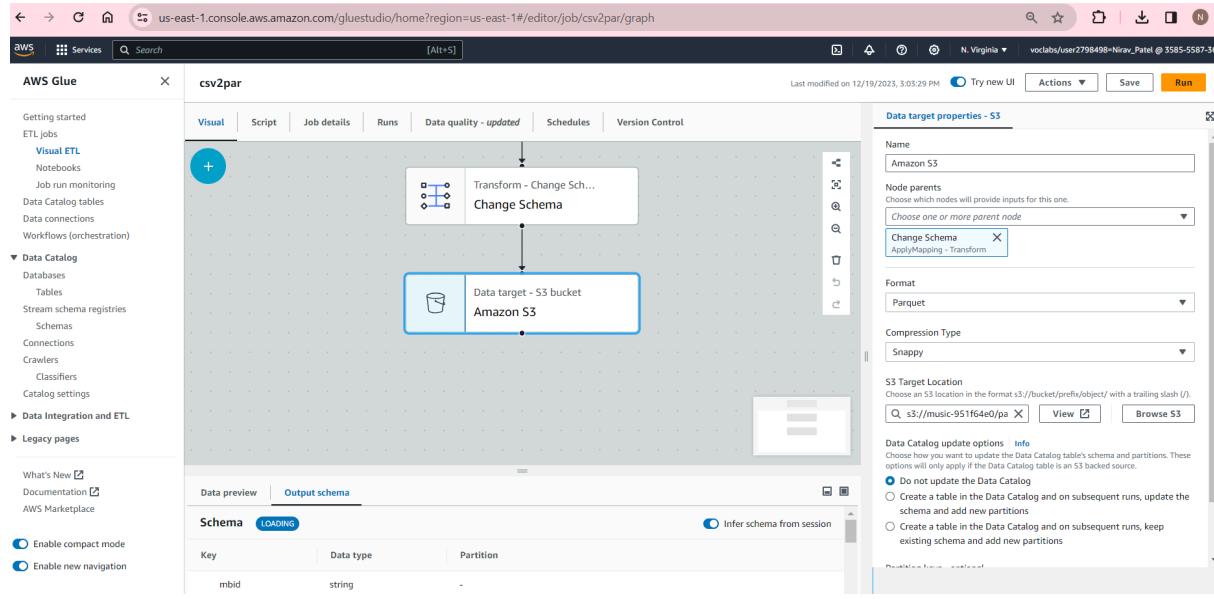
Results (1)

Search rows

#	_col0
1	42097

Glue ETL for CSV to Parquet





us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#/editor/job/csv2par/script

The screenshot shows the AWS Glue ETL job script editor. The script (Locked) contains the following Python code:

```

1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 args = getResolvedOptions(sys.argv, ["JOB_NAME"])
9 sc = SparkContext()
10 glueContext = GlueContext(sc)
11 spark = glueContext.spark_session
12 job = Job(glueContext)
13 job.init(args["JOB_NAME"], args)
14
15 # Script generated for node Amazon S3
16 AmazonS3_node1702998018319 = glueContext.create_dynamic_frame.from_options(
17     format_options={
18         "quoteChar": "'",
19         "withHeader": True,
20         "separator": ",",
21         "optimizePerformance": False,
22     },
23     connection_type="s3",
24     format="csv",
25     connection_options={
26         "paths": ["s3://music-951f64e0/csv/artists.csv"],
27         "recurse": True,
28     },
29     transformation_ctx="AmazonS3_node1702998018319",
30 )
31
32 # Script generated for node Change Schema
33 ChangeSchema_node1702998087670 = ApplyMapping.apply(
34     frame=AmazonS3_node1702998018319,
35     mappings=[
```

<https://us-east-1.console.aws.amazon.com/gluestudio/home?region=us-east-1#>

EMR Cluster configs and settings

us-east-1.console.aws.amazon.com/emr/home?region=us-east-1#/cloneCluster/j-88HN31ZKFW9A

Name and applications

Name: CA675 - Assignment 2

Amazon EMR release: emr-6.14.0

Application bundle

- Spark Interactive (selected)
- Core Hadoop
- Flink
- HBase
- Presto
- Trino
- Custom (selected)

Available applications:

- Flink 1.17.1
- HCatalog 3.1.3
- Hue 4.11.0 (selected)
- Livy 0.7.1
- Phoenix 5.1.3
- Spark 3.4.1 (selected)
- Tez 0.10.2
- ZooKeeper 3.5.10
- Ganglia 3.7.2
- Hadoop 3.3.3 (selected)
- JupyterEnterpriseGateway 2.6.0
- MXNet 1.9.1
- Pig 0.17.0
- Sqoop 1.4.7
- Trino 422
- HBase 2.4.17
- Hive 3.1.3 (selected)
- JupyterHub 1.5.0
- Oozie 5.2.1
- Presto 0.281
- TensorFlow 2.11.0
- Zeppelin 0.10.1

AWS Glue Data Catalog settings

Use the AWS Glue Data Catalog to provide an external metastore for your application.

Use for Hive table metadata

Use for Spark table metadata

Operating system options

Amazon Linux release

Custom Amazon Machine Image (AMI)

Automatically apply latest Amazon Linux updates

Summary

Name: CA675 - Assignment 2

Amazon EMR release: emr-6.14.0

Application bundle: Custom (Hadoop 3.3.3, Hive 3.1.3, Hue 4.11.0, JupyterHub 1.5.0, Livy 0.7.1, Pig 0.17.0, ...)

Amazon Linux release: 2.0.20230906.0

Cluster configuration

Uniform instance groups: Primary (m5.xlarge), Core (m5.xlarge)

Cluster scaling and provisioning

Provisioning configuration

Actions

Cancel | **Clone cluster**

CloudShell Feedback

Security Groups:

- sgr-06bf50f486529927a: All TCP, TCP, 0 - 65535, Custom, sg-06db5f8f69da304a9
- sgr-0c374e277b1359a5b: All ICMP - IPv4, ICMP, All, Custom, sg-01cd03b91c4ec06fe
- sgr-0463a85dc076b756: All UDP, UDP, 0 - 65535, Custom, sg-01cd03b91c4ec06fe
- sgr-071fcc0d12b9845b: All ICMP - IPv4, ICMP, All, Custom, sg-06db5f8f69da304a9
- : SSH, TCP, 22, Anywhere..., 0.0.0.0/0
- : All TCP, TCP, 0 - 65535, My IP, 78.16.71.192/32

Add rule

Warning: Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Actions

Cancel | Preview changes | **Save rules**

```
← → C 🏫 us-east-1.console.aws.amazon.com/ec2-instance-connect/ssh?connType=standard&instanceId=i-0be101da2c90908b
aws Services Search [Alt+S]

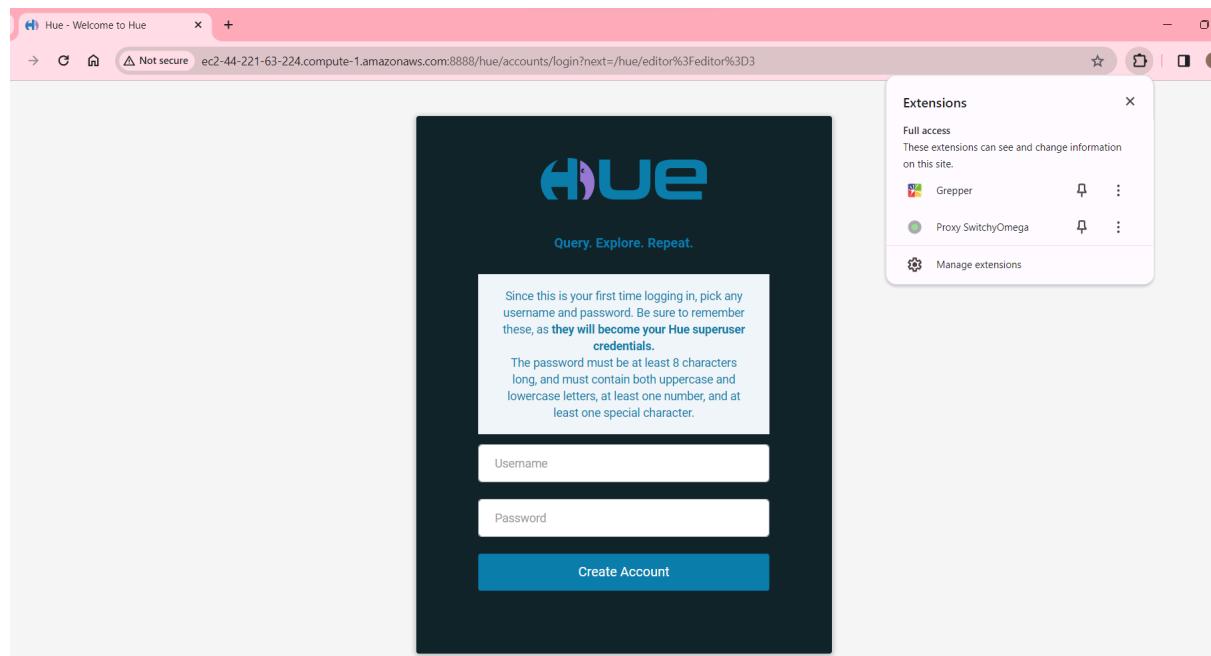
Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
53 package(s) needed for security, out of 72 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEE MMMMMMM MBBBBBBBBB RRRRRRRRRRRRRRRR
E:::::::E:::::M:::M M:::::M R:::::M:::::R
EE:::::EEEEE::::E M:::::M M:::::M R:::::RRRRRR::::R
E:::::E EEEEEE M:::::M M:::::M R:::::R R:::::R
E:::::E M:::::M M:::::M R:::::R R:::::R
E:::::EEEEE:::::E M:::::M M:::::M M:::::M R:::::RRRRRR::::R
E:::::EEEEE:::::E M:::::M M:::::M M:::::M R:::::RRRRRR::::RR
E:::::EEEEE:::::E M:::::M M:::::M M:::::M R:::::RRRRRR::::R
E:::::E M:::::M M:::::M M:::::M R:::::R R:::::R
E:::::E EEEEEE M:::::M M:::::M R:::::R R:::::R
EE:::::EEEEE:::::E M:::::M M:::::M R:::::R R:::::R
E:::::EEEEE:::::E M:::::M M:::::M R:::::R R:::::R
EEEEEEEEEEEEEEEEEEEEE MMMMMMM RRRRRRR RRRRRR

[root@ip-172-31-70-167 ~]#
```

Apache Hue for Hadoop operations UI



Not secure ec2-44-221-63-224.compute-1.amazonaws.com:8888/hue/editor?editor=3

Hive Add a name... Add a description...

default (0) +

No entries found

Tables

create database music;

```
INFO : Starting task [Stage 0.0/0] in serial mode
INFO : Completed executing command(queryId=hive_20231219152834_870af479-7df8-4e74-85b3-82627734eede)
0.31 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
```

Success.

Query History Saved Queries

a few seconds ago → create database music

Not secure ec2-44-221-63-224.compute-1.amazonaws.com:8888/hue/editor?editor=4

Hive artists tables and analysis to load and filter the data fo...

music (1) +

Tables Filter...

artists

```
3 | artist_mb` string,
4 | artist_lastfm` string,
5 | country_mb` string,
6 | country_lastfm` string,
7 | tags_mb` string,
8 | tags_lastfm` string,
9 | listeners_lastfm` string,
10 | scrobbles_lastfm` string,
11 | ambiguous_artist` BOOLEAN
12 )
13 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
14 WITH SERDEPROPERTIES (
15   'separatorChar' = ',',
16   'quoteChar' = '\"',
17   'escapeChar' = '\"'
18 )
19 STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat' OUTPUTFORMAT 'org.apache.hadoop.h
20 LOCATION 's3://music-951f64e0/csv'
21 TBLPROPERTIES ('classification' = 'csv');
22 ;
```

```
INFO : Starting task [Stage 0.0/0] in serial mode
INFO : Completed executing command(queryId=hive_20231219153229_8eaac624-31a3-4934-8888-cfaa84e581c1)
1.338 seconds
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
```

Success.

Not secure ec2-44-221-63-224.compute-1.amazonaws.com:8888/hue/editor?editor=5

Hive artists tables and analysis to load and filter the data fo... 0.18s music ?

Tables Filter... Tables

music.artists

	mbid	artist_mb	artist_lastfm	country_mb
1	mbid	artist_mb	artist_lastfm	country_mb
2	cc197bad-de9c-440d-a5b5-d52ba2e14234	Coldplay	Coldplay	United Kingdom
3	a74b1b7f-71a5-4011-9441-d0b5e4122711	Radiohead	Radiohead	United Kingdom
4	8bfac288-ccc5-448d-9573-c33ea2aa5c30	Red Hot Chili Peppers	Red Hot Chili Peppers	United States
5	73e5e69d-3554-40d8-8516-00cb38737a1c	Rihanna	Rihanna	United States
6	b95ce3ff-3d05-4e87-9e01-c97b66af13d4	Eminem	Eminem	United States
7	95e1ead9-4d31-4808-a7ac-32c3614c116b	The Killers	The Killers	United States

Query History Saved Queries Results (100+)

artists.mbid artists.artist_mb artists.artist_lastfm artists.country_mb

music.artists

	mbid	artist_mb	artist_lastfm	country_mb
1	mbid	artist_mb	artist_lastfm	country_mb
2	cc197bad-de9c-440d-a5b5-d52ba2e14234	Coldplay	Coldplay	United Kingdom
3	a74b1b7f-71a5-4011-9441-d0b5e4122711	Radiohead	Radiohead	United Kingdom
4	8bfac288-ccc5-448d-9573-c33ea2aa5c30	Red Hot Chili Peppers	Red Hot Chili Peppers	United States
5	73e5e69d-3554-40d8-8516-00cb38737a1c	Rihanna	Rihanna	United States
6	b95ce3ff-3d05-4e87-9e01-c97b66af13d4	Eminem	Eminem	United States
7	95e1ead9-4d31-4808-a7ac-32c3614c116b	The Killers	The Killers	United States

Not secure ec2-44-221-63-224.compute-1.amazonaws.com:8888/hue/editor?editor=8

Hive artists tables and analysis to load and filter the data fo... 0.35s music ?

Tables Filter... Tables

music.artists

```
1 Select * from artists where mbid != 'mbid'
2 and artist_mb = 'Radiohead'
```

INFO : Compiling command(queryId=hive_20231219154502_0fd85bd5-4515-4d95-9a0d-f5816640bdd3): Select * from artists
where mbid != 'mbid'
and artist_mb = 'Radiohead'
INFO : Concurrency mode is disabled, not creating a lock manager
INFO : Semantic Analysis Completed (cost=1.00s)

Query History Saved Queries Results (1)

artists.mbid artists.artist_mb artists.artist_lastfm artists.country_mb

	mbid	artist_mb	artist_lastfm	country_mb
1	a74b1b7f-71a5-4011-9441-d0b5e4122711	Radiohead	Radiohead	United Kingdom

Not secure ec2-44-221-63-224.compute-1.amazonaws.com:8888/hue/editor?editor=13

The screenshot shows the Apache Hue Editor interface. On the left is a sidebar with icons for file operations like New, Open, Save, and Delete. The main area has a search bar at the top. Below it, there's a navigation bar with tabs for Hive, artists tables and analysis, and a placeholder to load and filter the data. A status bar indicates "0.35s music".

Tables (1) + **music**

Query History (100+)

	artist_name	tags
1	Coldplay	["rock", "pop", "alternative rock", "british", "uk", "britannique", "britpop", "pop rock", "piano pop"]
2	Radiohead	["rock", "electronic", "alternative rock", "british", "grunge", "uk", "britannique", "britpop", "art rock"]
3	Red Hot Chili Peppers	["rock", "alternative rock", "80s", "90s", "rap", "metal", "american", "crossover", "usa", "funk"]
4	Rihanna	["pop", "dance", "hip hop", "reggae", "contemporary r b", "electropop", "rnb", "barbadian", "dance"]
5	Eminem	["turkish", "rap", "american", "hip-hop", "hip hop", "hiphop", "midwest hip-hop", "detroit hip-hop"]
6	The Killers	["synthpop", "alternative rock", "american", "new wave", "usa", "indie rock", "alternative", "american"]

Not secure ec2-44-221-63-224.compute-1.amazonaws.com:8888/hue/editor?editor=24#id=application_1702999451997_0002

The screenshot shows the Apache Hue Editor interface with a similar layout to the first one. The left sidebar includes a question mark icon. The main area has a search bar and a navigation bar with tabs for Hive, artists tables and analysis, and a placeholder to load and filter the data.

Tables (1) + **music**

Query History (100+)

	artist_name	tags
1	Coldplay	["rock", "pop", "alternative rock", "british", "uk", "britannique", "britpop", "pop rock", "piano pop"]
2	Radiohead	["rock", "electronic", "alternative rock", "british", "grunge", "uk", "britannique", "britpop", "art rock"]
3	Red Hot Chili Peppers	["rock", "alternative rock", "80s", "90s", "rap", "metal", "american", "crossover", "usa", "funk"]
4	Rihanna	["pop", "dance", "hip hop", "reggae", "contemporary r b", "electropop", "rnb", "barbadian", "dance"]
5	Eminem	["turkish", "rap", "american", "hip-hop", "hip hop", "hiphop", "midwest hip-hop", "detroit hip-hop"]
6	The Killers	["synthpop", "alternative rock", "american", "new wave", "usa", "indie rock", "alternative", "american"]

Jobs **Workflows** **Schedules**

HIVE-0bd979ec-ef79-43b2-bc13-417d5ab0e498

ID: application_1702999451997_0002

TYPE: TEZ

PROGRESS: 19.093746%

Logs **Attempts**

```

7.283: [GC (Allocation Failure) [PSYoungGen: 389741K->19957K(467456K)] 335578K->46332K(612864K), 0.0174164 secs] [Times: user=0.04 sys=0.01, real=0.02 secs]
7.740: [GC (Metadata GC Threshold) [PSYoungGen: 20788K->16750K(481280K)] 234263K->43125K(626688K), 0.0189727 secs] [Times: user=0.04 sys=0.01, real=0.02 secs]
7.759: [Full GC (Metadata GC Threshold) [PSYoungGen: 16750K->0K(481280K)] ParallelGC: 26375K->0K(197128K)] 43125K->30035K(678400K), [Metaspace: 58414K->5413K(1101824K)], 0.1559352 secs] [Times: user=0.39 sys=0.00, real=0.16 secs]
9.594: [GC (Allocation Failure) [PSYoungGen: 458752K->11672K(688768K)] 488787K->41723K(805888K), 0.0128573 secs] [Times: user=0.03 sys=0.01, real=0.01 secs]
2023-12-19 16:06:23 Completed Dag: dag_1702999451997_0002_1
2023-12-19 16:10:01 Running Dag: dag_1702999451997_0002_2

```

Export Final Report

```
[root@ip-172-31-70-167 ~]# hive -e "Select artist.lastfm.artist_name, country_mb.country_origin, split(country.lastfm,';') as country_list, cast(listeners.lastfm as int) listeners_cnt, cast(scrobbles.lastfm as int) scrobbles_cnt, split(tags_mb,';') tags from music.artists where mbid != 'mbid' and country.lastfm != '' and artist_mb != '' order by listeners_cnt desc' > /final.csv
hive Session ID = 43dfe4d2-e91a-4203-94a5-5c521aef578
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
Query ID = root_2023121916161818_dce81ee3-a8ce-45a3-925e-e2d9c3860a3
Portal Job: Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1702999451997_0006)
Map 1: /- Reducer 2: 0/1
Map 1: 0/3 Reducer 2: 0/1
Map 1: 0/3 Reducer 2: 0/1
Map 1: 0/(2)/3 Reducer 2: 0/1
Map 1: 1/(1)/3 Reducer 2: 0/1
Map 1: 1/(2)/3 Reducer 2: 0/1
Map 1: 1/(2)/3 Reducer 2: 0/(1)/1
Map 1: 3/3 Reducer 2: 0/(1)/1
Map 1: 3/3 Reducer 2: 1/1
OK
Time taken: 31.547 seconds, Fetched: 218698 row(s)
[root@ip-172-31-70-167 ~]#
```

i-0be101da2c90908b7
PublicIPs: 44.221.63.224 PrivateIPs: 172.31.70.167

```
[root@ip-172-31-70-167 ~]# ls .
[root@ip-172-31-70-167 ~]# hadoop fs -ls
Found 2 items
drwxr-xr-x - root hdfsadmingroup 0 2023-12-19 15:40 .hiveJars
-rw-r--r-- 1 root hdfsadmingroup 12945549 2023-12-19 16:21 final.csv
[root@ip-172-31-70-167 ~]# hadoop fs -copyToLocal /final.csv final.csv
copyToLocal: '/final.csv': No such file or directory
[root@ip-172-31-70-167 ~]# hadoop fs -copyToLocal final.csv final.csv
[root@ip-172-31-70-167 ~]# ls
final.csv
[root@ip-172-31-70-167 ~]# aws s3 cp final.csv s3://output-951f64e0/final.csv
upload: ./final.csv to s3://output-951f64e0/final.csv
[root@ip-172-31-70-167 ~]#
```

i-0be101da2c90908b7

PublicIPs: 44.221.63.224 PrivateIPs: 172.31.70.167

Amazon S3 > Buckets > output-951f64e0

output-951f64e0 [Info](#)

[Objects](#) [Properties](#) [Permissions](#) [Metrics](#) [Management](#) [Access Points](#)

Objects (1) [Info](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects and grant them permissions. [Learn more](#)

[Refresh](#) [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#)

[Find objects by prefix](#) [Show versions](#)

<input checked="" type="checkbox"/>	Name	Type	Last modified
<input checked="" type="checkbox"/>	final.csv	CSV	December 19, 2023, 16:27:37 (UTC+00:00)

pyspark operations

ca675-pyspark [star](#)

File Edit View Insert Runtime Tools Help Last saved at 2:18PM [Comment](#)

Table of contents [+ Code](#) [+ Text](#)

Section

```
[ ] !apt-get install openjdk-8-jdk-headless -qq > /dev/null
[ ] !wget -q http://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
[ ] !tar xf spark-3.1.1-bin-hadoop3.2.tgz
[ ] !pip install -q findspark
```

```
[ ] import os
[ ] os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
[ ] os.environ["SPARK_HOME"] = "/content/spark-3.1.1-bin-hadoop3.2"
```

```
[ ] !ls
[ ] artists.csv sample_data spark-3.1.1-bin-hadoop3.2 spark-3.1.1-bin-hadoop3.2.tgz
```

```
[ ] import findspark
[ ] findspark.init()
[ ] from pyspark.sql import SparkSession
[ ] spark = SparkSession.builder.master("local[*]").getOrCreate()
[ ] spark.conf.set("spark.sql.repl.eagerEval.enabled", True) # Property used to format output tables better
[ ] spark
```

SparkSession - in-memory
SparkContext
Spark UI
Version v3.1.1
Master local[*]

+ Code + Text

```
# header=True means the first row is a header
# sep=';' means the column are separated using ''
df_artists = spark.read.csv('artists.csv', header=True, sep=";")
df_artists.show(5)
df_artists.count()
```

mbid	artist_mb	artist_lastfm	country_mb	country_lastfm	tags_mb	tags_lastfm
cc197bad-dc9c-440...	Coldplay	Coldplay	United Kingdom	United Kingdom rock; pop; altern...	rock; alternative...	
a74b1b7f-71a5-401...	Radiohead	Radiohead	United Kingdom	United Kingdom rock; electronic;...	alternative; alte...	
b8fac288-ccc5-448...	Red Hot Chili Pep...	Red Hot Chili Pep...	United States	United States rock; alternative...	rock; alternative...	
j3e5e69d-3554-40d...	Rihanna	Rihanna	United States Barbados; United ...	pop; dance; hip h...	pop; rnb; female ...	
b95ce3ff-3d05-4e8...	Eminem	Eminem	United States	United States turkish; rap; ame...	rap; Hip-Hop; Emi...	

only showing top 5 rows

594907

+ Code + Text

```
[ ] #df_artists.columns;
df_artists.printSchema()
```

```
root
|-- mbid: string (nullable = true)
|-- artist_mb: string (nullable = true)
|-- artist_lastfm: string (nullable = true)
|-- country_mb: string (nullable = true)
|-- country_lastfm: string (nullable = true)
|-- tags_mb: string (nullable = true)
|-- tags_lastfm: string (nullable = true)
|-- listeners_lastfm: string (nullable = true)
|-- scrobbles_lastfm: string (nullable = true)
```

+ Code + Text

594907

+ Code + Text

```
[ ] # string cleaning and removing unwanted
str_cols = [item[0] for item in df_artists.dtypes if item[1].startswith('string')]
for cols in str_cols:
    df_artists = df_artists.withColumn(cols, trim(df_artists[cols]))
```

```
[ ] # lowercasing the columns and put the underscore
```

```
# create the temp view table for the sql operations
df_artists.createOrReplaceTempView("artisttbl")
```

+ Code + Text

```
# to understand the data
# define the categories
# 1. artist_mb
# 2. artist_lastfm
# 3. artist_lastfm
# 4. country_mb
# 5. country_lastfm
# 6. tags_mb
# 7. ambiguous_artist bool check True or false

# * indicator_code : to represent the valuation indicator code unit name
```

```
query_country_counts = '''select row_number() over (order by country_mb asc) rn, country_mb country, count(*) count
from artisttbl art
where art.country_mb is not null
group by country_mb having count(*) > 1'''
```

```
query_artist_counts = '''select row_number() over (order by artist_mb asc) rn, country_mb country, artist_mb artist, count(*) count
```

The screenshot shows a Jupyter Notebook interface with the following details:

- Toolbar:** Includes icons for File, Edit, Insert, Cell, Kernel, Help, and a Connect dropdown.
- Code Cell:** Contains Python code for saving dataframes to CSV files:


```
df_all.to_csv('all_artists.csv', index = False, encoding='utf-8')
df_known_artists.to_csv('known_artists.csv', index = False, encoding='utf-8')
```
- Data Preview:** A table titled "df_all_artists" is displayed with columns: mbid, artist_mb, artist_lastfm, country_mb, country_lastfm, tags_mb, tags_lastfm, listeners_lastfm, scrobbles_lastfm, and ambiguous_artist. The table contains 10 rows of data.

GCP/ElasticCloud integrations dataflow jobs

The screenshot shows the Google Cloud Dataflow console with the following details:

- Left Sidebar:** Includes links for Dataflow, Overview, Monitoring, Jobs, Pipelines, Workbench, Snapshots, and SQL workspace.
- Header:** Shows the URL "console.cloud.google.com/dataflow/jobs?project=uplifted-stream-408416".
- Jobs Tab:** Displays a table with one row for a job named "gcs123". The table columns include Name, Type, End time, Elapsed time, Start time, Status, SDK version, ID, Region, and Insights.
- Right Sidebar:** Features sections for "Try Dataflow", "Try with Java", "Try with Python", and "Try with notebooks".

Installing agents for the elastic cloud S3 integrations

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

S C:\WINDOWS\system32> $ProgressPreference = 'SilentlyContinue'
> Invoke-WebRequest -Uri https://artifacts.elastic.co/downloads/beats/elastic-agent/elastic-agent-8.11.3-windows-x86_64.zip -OutFile elastic-agent-8.11.3-windows-x86_64.zip
> Expand-Archive .\elastic-agent-8.11.3-windows-x86_64.zip -DestinationPath .
> cd elastic-agent-8.11.3-windows-x86_64
> .\elastic-agent.exe install --url=https://e32b28ceae647258819f37d5d742ab2.fleet.us-central1.gcp.cloud.es.io:443 --enrollment-token=ekVoN2c0dJnd0pHTGdTcVJ2NW86VH16M2JvazNScXVNvp6NUZFWUNEZw==
lastic Agent will be installed at C:\Program Files\Elastic\Agent and will run as a service. Do you want to continue? [Yn]:y
opying files..... DONE
```

The screenshot shows the 'Set up AWS integration' page in the Elastic Cloud interface. The top navigation bar includes links for 'Integrations', 'Amazon S3', and 'Add integration'. The main title is 'Set up AWS integration'. A progress bar indicates three steps: 'Install Elastic Agent' (green dot), 'Add the integration' (blue dot), and 'Confirm incoming data' (white dot). The 'Add the integration' section is active, showing configuration for collecting S3 access logs and metrics. Under 'Collect S3 access logs from S3', the 'AWS S3 Access Logs' option is selected, with a Queue URL of 's3://output-527c9be0/final_artists.csv'. There are options to 'Preserve original event' and 'Advanced options'. Under 'Collect S3 metrics', the 'AWS S3 daily storage metrics' option is selected, with a 'Collection Period' of '24h' and a 'Regions' field set to 'Optional'. At the bottom are 'Go back' and 'Confirm incoming data' buttons.

music.kb.us-central1.gcp.cloud.es.io:9243/app/fleet/integrations/aws-2.11.0/add-integration/s3?useMultiPageLayout

elastic Find apps, content, and more. Live Chat Setup guide: step 1

Integrations > Amazon S3 Add integration

Set up AWS integration

Install Elastic Agent Add the integration Confirm incoming data

✓ Incoming data received from 1 enrolled agent.

Preview of incoming data:

```

Dec 19, 2023 @ 19:28:02.363 agent.name: "shanks-pc" agent.type: "fleetbeat" agent.version: "8.11.3" log.file.path: "C:\Program Files\Elastic\Agent\data\elasticsearch-agent-f4fbfb\log\filebeat-20231219-1.ndjson" log.file.vol: "334856515" log.file.idolo: "18171" log.file.idoh: "1245184" log.offset: 187217 log.source: "elastic-agent" elastic.agent.version: "8.11.3" elastic.agent.snapshot: false message: "Source URI changes from \https://artifacts.elastic.co/downloads/ to \https://artifacts.elastic.co/downloads/\log.origin.file.line: 111 log.origin.file.name: "upgrade/upgrade.ic.co/downloaded/" log.origin.file.line: 111 log.origin.file.name: "upgrade/upgrade.
Dec 19, 2023 @ 19:28:02.363 agent.name: "shanks-pc" agent.type: "fleetbeat" agent.version: "8.11.3" log.file.path: "C:\Program Files\Elastic\Agent\data\elasticsearch-agent-f4fbfb\log\filebeat-20231219-1.ndjson" log.file.vol: "334856515" log.file.idolo: "18171" log.file.idoh: "1245184" log.offset: 187524 log.source: "elastic-agent" elastic.agent.version: "8.11.3" elastic.agent.snapshot: false message: "Updating running component model" log.origin.file.line: 1076 log.origin.file.name: "coordinator/coordinator.go" input.type: "filestream" ecs.version: "8.0.0" data.stream.type: "log
Dec 19, 2023 @ 19:28:03.358 agent.name: "shanks-pc" agent.type: "fleetbeat" agent.version: "8.11.3" log.file.path: "C:\Program Files\Elastic\Agent\data\elasticsearch-agent-f4fbfb\log\filebeat-20231219-1.ndjson" log.file.vol: "334856515" log.file.idolo: "18171" log.file.idoh: "1245184" log.offset: 123438 log.source: "elastic-agent" elastic.agent.version: "8.11.3" elastic.agent.snapshot: false message: "Spanned new component aws-s3-default: Starting: spawned pid: 428" log.origin.file.line: 519 log.origin.file.name: "coordinator/coordinator.go" input.type: "filestream"

```

Add another integration View assets

Elasticsearch cloud configs and final kibana dashboard

music.kb.us-central1.gcp.cloud.es.io:9243/app/dev_tools#/console

elastic Find apps, content, and more. Live Chat Setup guide: step 1

Dev Tools Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Variables Help

Click to send request

```

1 # Click the Variables button, above, to create your own variables.
2 GET ${exampleVariable}/_search
3 {
4   "query": {
5     "$exampleVariable": {} // match_all
6   }
7 }
8
9
10 GET _cat/indices?v
11
12 GET known_artists/_mapping
13
14 GET known_artists/_settings
15
16 GET known_artists/_search
17 {
18   "query": {
19     "bool": {
20       "must": [
21         {
22           "terms": {
23             "artist_mb": [
24               "Metallica",
25               "Metallica"
26             ]
27           }
28         }
29       ]
30     }
31   }
32 }
33
34 GET known_artists/_search
35 {
36   "query": {
37     "bool": {
38       "tags_lastfm": {
39         "value": "rock"
40       }
41     }
42   }
43 }

```

id	pri	rep	docs.count	docs.deleted	store.size	pri.store.size	dataset.size
l1-Ctrfj5lyk_4Xy-6Nlgi	1	1	100000	0	10.7kb	10.7kb	10.7kb
5pHmxe025Gm7tW1_11Rg	1	0	0	0	250b	250b	250b
97zawexX5Geeh40CrRng	1	0	0	0	250b	250b	250b
67n12oC81bd0npY14Q2A	1	1	2652	0	778.9kb	778.9kb	778.9kb
36898	0	250b	250b	250b	250b	250b	250b
j1ffybe00muovft6x1bA	1	0	0	0	250b	250b	250b
5T4dVfzq4-1	1	0	0	0	250b	250b	250b
creDyfrf5cq4-Pnud7yaw	1	0	0	0	250b	250b	250b
7Zbimt542ycbnh80ig	1	0	4	0	65.7kb	65.7kb	65.7kb
11 green open _ds-log-enterprise_search.api-default-2023.12.18-000001	1	0	8	0	94.1kb	94.1kb	94.1kb
12 green open _ds-log-enterprise_search.audit-default-2023.12.18-000001	1	0	0	0	500b	500b	500b
13 green open _elastic-connectors-v1	0	0	0	0	0	0	0
ygbV2a15-C70HgdxZw	1	0	1	0	19.6kb	19.6kb	19.6kb
14 green open _internal-alerts-observability.metrics.alerts-default-000001	1	0	0	0	250b	250b	250b
15 green open _internal-alerts-observability.threshold.alerts-default-000001	1	0	0	0	250b	250b	250b
16 green open _internal-alerts-observability.threshold.alerts-default-000001	1	0	0	0	250b	250b	250b
17 green open _internal-alerts-observability.sync-jobs-v1	1	0	0	0	240b	240b	240b
18 green open _internal-connectors-sync-jobs-v1	1	0	0	0	250b	250b	250b
19 green open _kibana-observability-ai-assistant-kb-000001	1	0	0	0	250b	250b	250b
20 yellow open _ds-metrics-apm.apm_server-default-2023.12.18-000001	1	1	4689	0	490.5kb	490.5kb	490.5kb
21 green open _internal.alerts-stack.alerts-default-000001	1	0	0	0	250b	250b	250b

music.kb.us-central1.gcp.cloud.es.io:9243/app/dev_tools#/console

elastic Dev Tools Console

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Variables Help

```

1 # Click the Variables button, above, to create your own variables.
2 GET ${exampleVariable1} // _search
3 {
4   "query": {
5     "term": {
6       "field": "exampleVariable2"
7     }
8   }
9 }
10 GET _cat/indices?v
11 GET known_artists/_mapping
12 GET known_artists/_settings
13 GET known_artists/_search
14 GET known_artists/_search
15 GET known_artists/_search
16 GET known_artists/_search
17 {
18   "query": {
19     "bool": {
20       "must": [
21         {
22           "term": {
23             "artist_mb": [
24               "Metallica",
25               "Metallica"
26             ]
27           }
28         }
29       ]
30     }
31   }
32 }
33
34 GET known_artists/_search
35 {
36   "query": {
37     "fuzzy": {
38       "tags_lastfm": {
39         "value": "rock"
40       }
41     }
42 }
43 }
44
45

```

1 {
2 "took": 0,
3 "timed_out": false,
4 "_shards": {
5 "total": 1,
6 "successful": 1,
7 "skipped": 0,
8 "failed": 0
9 },
10 "hits": {
11 "total": 1,
12 "max_score": 1,
13 "relation": "eq"
14 },
15 "max_score": 1,
16 "hits": [
17 {
18 "index": "known_artists",
19 "id": "C2DrFvWg0D1gAgc11",
20 "score": 1,
21 "source": {
22 "name": "Metallica",
23 "tags": "speed metal; speed metal; californians; los angeles; crap; bay area; trash metal; 80s metal; classic metal; thrash metal; rock and indie; metallica; cl metal; 80s metal; classic thrash metal; 90s metal; 80's; 90's; hard'n/heavy; lame; a filk artist; the unforgiving; stopped being good after '97; stopped b after '88; poncho; ass-kicker",
24 "tags_lastfm": "Metallica",
25 "mbid": "65f4fe05-ef9e-490c-aec3-909e7ae6b2ab",
26 "artist_mb": "Metallica",
27 "tags": "speed metal; speed metal; californians; los angeles; crap; bay area; trash metal; 80s metal; classic metal; thrash metal; rock and indie; metallica; cl metal; 80s metal; classic thrash metal; 90s metal; 80's; 90's; hard'n/heavy; lame; a filk artist; the unforgiving; stopped being good after '97; stopped b after '88; poncho; ass-kicker",
28 "tags_lastfm": "Metallica",
29 "country": "United States",
30 "listeners": 2894382, "lastfm": 2894382, "tags": "speed metal; speed metal; californians; los angeles; crap; bay area; trash metal; 80s metal; classic metal; thrash metal; rock and indie; metallica; cl metal; 80s metal; classic thrash metal; 90s metal; 80's; 90's; hard'n/heavy; lame; a filk artist; the unforgiving; stopped being good after '97; stopped b after '88; poncho; ass-kicker",
31 "tags_lastfm": "thrash metal; metal; heavy metal; hard rock; rock; seen live; metallica; speed metal; classic rock; american; thrash; 80s; trash metal; 90s; alternative; classic metal; Progressive metal; alternative rock; heavy; legends; Power metal",
32 "tags": "speed metal; speed metal; californians; los angeles; crap; bay area; trash metal; 80s metal; classic metal; thrash metal; rock and indie; metallica; cl metal; 80s metal; classic thrash metal; 90s metal; 80's; 90's; hard'n/heavy; lame; a filk artist; the unforgiving; stopped being good after '97; stopped b after '88; poncho; ass-kicker",
33 "lastfm": 2894382, "tags": "speed metal; speed metal; californians; los angeles; crap; bay area; trash metal; 80s metal; classic metal; thrash metal; rock and indie; metallica; cl metal; 80s metal; classic thrash metal; 90s metal; 80's; 90's; hard'n/heavy; lame; a filk artist; the unforgiving; stopped being good after '97; stopped b after '88; poncho; ass-kicker",
34 "tags": "speed metal; speed metal; californians; los angeles; crap; bay area; trash metal; 80s metal; classic metal; thrash metal; rock and indie; metallica; cl metal; 80s metal; classic thrash metal; 90s metal; 80's; 90's; hard'n/heavy; lame; a filk artist; the unforgiving; stopped being good after '97; stopped b after '88; poncho; ass-kicker",
35 "lastfm": 2894382, "tags": "speed metal; speed metal; californians; los angeles; crap; bay area; trash metal; 80s metal; classic metal; thrash metal; rock and indie; metallica; cl metal; 80s metal; classic thrash metal; 90s metal; 80's; 90's; hard'n/heavy; lame; a filk artist; the unforgiving; stopped being good after '97; stopped b after '88; poncho; ass-kicker",
36 }
37 }
38]
39 }
40 }
41 }
42 }
43 }
44 }
45 }

