Project 2

Music Data Analysis

|  |  |
| --- | --- |
| **Prepared For:** | AcadGild |
|  |  |
| **Document Approval:** | **AcadGild** |
|  |  |
|  |  |
|  |  |
|  |  |
| **Project Title:** | Project 2 |
|  |  |
| **Prepared By:** | Duncan Burgess |
|  |  |
|  | dburgess@duncb.com |
|  |  |
| **Primary Engineer:** | Duncan Burgess |
|  |  |
| **Document Reference:** | **Project 2** |
|  |  |
| **Start Date:** | 20/10/2017 |
|  |  |
|  |  |



# 

# Contents

[Contents 2](#_Toc496973787)

[Change History 3](#_Toc496973788)

[1. Problem Statement 4](#_Toc496973789)

[2. Datasets 6](#_Toc496973790)

[3. Data Ingestion 8](#_Toc496973791)

[3.1. Populate the Look-Up tables 8](#_Toc496973792)

[3.2. Perform Data Formatting 12](#_Toc496973793)

[3.3. Perform Data Enrichment and Cleaning 15](#_Toc496973794)

[4. Analysing Data 18](#_Toc496973795)

[5. Results 21](#_Toc496973796)

[6. Automation 22](#_Toc496973797)

# Change History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Document Revision** | **Date** | **Authored By** | **Authorised By** | **Sections Affected** | **Reason for Change** |
| Rev 01 | 26/10/2017 | Duncan Burgess |  | All | Initial release. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Problem Statement

A leading music-catering company is planning to analyse large amount of data received from varieties of sources, namely mobile app and website to track the behaviour of users, classify users, calculate royalties associated with the song and make appropriate business strategies. The file server receives data files periodically after every 3 hours.

**Fields present in the data files**

Data files contain below fields.

|  |  |
| --- | --- |
| Column Name/Field Name | Column Description/Field Description |
| User\_id | Unique identifier of every user |
| Song\_id | Unique identifier of every song |
| Artist\_id | Unique identifier of the lead artist of the song |
| Timestamp | Timestamp when the record was generated |
| Start\_ts | Start timestamp when the song started to play |
| End\_ts | End timestamp when the song was stopped |
| Geo\_cd | Can be 'A' for USA region, 'AP' for asia pacific |
|  | region, 'J' for Japan region, 'E' for europe and |
|  | 'AU' for australia region |
| Station\_id | Unique identifier of the station from where the |
|  | song was played |
| Song\_end\_type | How the song was terminated. |
|  | 0 means completed successfully |
|  | 1 means song was skipped |
|  | 2 means song was paused |
|  | 3 means other type of failure like device issue, |
|  | network error etc. |
| Like | 0 means song was not liked |
|  | 1 means song was liked |
| Dislike | 0 means song was not disliked |
|  | 1 means song was disliked |

**LookUp Tables**

There are some existing look up tables present in NoSQL databases. They play an important role in data enrichment and analysis.

|  |  |
| --- | --- |
| Table Name | Description |
| Station\_Geo\_Map | Contains mapping of a geo\_cd with station\_id |
| Subscribed\_Users | Contains user\_id, subscription\_start\_date and |
|  | subscription\_end\_date. |
|  | Contains details only for subscribed users |
| Song\_Artist\_Map | Contains mapping of song\_id with artist\_id |
|  | alongwith royalty associated with each play of |
|  | the song |
| User\_Artist\_Map | Contains an array of artist\_id(s) followed by a |
|  | user\_id |

**DATASET:**

1. Data coming from web applications reside in /data/web and has xml format.
2. Data coming from mobile applications reside in /data/mob and has csv format.
3. Data present in lookup directory should be used in HBase.

**Data Enrichment**

Rules for data enrichment

1. If any of *like* or *dislike* is NULL or *absent*, consider it as 0.
2. If fields like *Geo\_cd* and *Artist\_id* are NULL or *absent*, consult the lookup tables for fields *Station\_id* and *Song\_id* respectively to get the values of *Geo\_cd* and *Artist\_id*.
3. If corresponding lookup entry is not found, consider that record to be invalid.

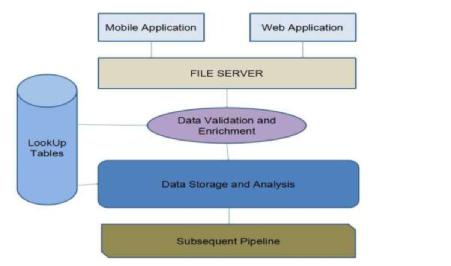
|  |  |  |
| --- | --- | --- |
| **NULL or absent field** | **Look up field** | **Look up table (Table from**  **which record can be updated)** |
| Geo\_cd | Station\_id | Station\_Geo\_Map |
| Artist\_id | Song\_id | Song\_Artist\_Map |

**Challenges and Optimisations:**

1. LookUp tables are in NoSQL databases. Integrate them with the actual data flow.
2. Try to make joins as less expensive as possible.
3. Data Cleaning, Validation, Enrichment, Analysis and Post Analysis have to be automated. Try using schedulers.
4. Appropriate logs have to be maintained to track the behaviour and overcome failures in the pipeline.

**Flow of operations**

A schematic flow of operations is shown below



# Datasets

The following datasets are to be used:

**Mobile – Files.txt**

U114,S207,A303,1465130523,1465230523,1475130523,A,ST415,3,1,0

U107,S202,A303,1495130523,1465230523,1465230523,U,ST415,0,1,1

U100,S204,A302,1495130523,1475130523,1465130523,AU,ST408,2,1,1

U104,S202,A303,1465230523,1475130523,1465130523,A,ST409,2,0,1

U102,S207,A301,1465230523,1485130523,1465230523,AU,ST403,3,1,1

,S203,A302,1495130523,1475130523,1465230523,E,ST400,0,0,1

U106,S202,A302,1465230523,1465130523,1465130523,AU,ST408,0,1,1

U105,S207,A300,1465230523,1485130523,1465130523,U,ST400,2,0,1

U108,S205,A304,1465130523,1465130523,1475130523,,ST410,2,1,0

U105,S203,,1475130523,1465230523,1465130523,AU,ST408,2,0,1

U110,S203,A300,1465230523,1465130523,1485130523,A,ST415,0,1,1

U113,S200,A303,1465230523,1475130523,1465130523,E,ST413,3,1,1

U119,S208,A302,1495130523,1465230523,1465230523,U,ST415,3,0,0

U118,S208,A303,1475130523,1465130523,1465230523,E,ST415,3,0,0

U107,S210,A302,1475130523,1485130523,1485130523,AP,ST404,2,1,0

U118,S202,A300,1495130523,1465230523,1465230523,AP,ST410,1,0,0

U111,S206,A305,1465130523,1465130523,1485130523,AU,ST415,0,1,1

U116,S208,A303,1465230523,1485130523,1475130523,A,ST413,1,0,1

U101,S202,A300,1465230523,1465130523,1475130523,U,ST401,0,0,1

U120,S206,A303,1495130523,1485130523,1465130523,AU,ST414,0,0,0

**Web file.xml exert**

<?xml version="1.0"?><records><record><user\_id>U106</user\_id><song\_id>S205</song\_id><artist\_id>A300</artist\_id><timestamp>2016-05-10 12:24:22</timestamp><start\_ts>2016-05-10 12:24:22</start\_ts><end\_ts>2017-05-09 08:09:22</end\_ts><geo\_cd>AP</geo\_cd><station\_id>ST407</station\_id><song\_end\_type>2</song\_end\_type><like>1</like><dislike>1</dislike></record><record><user\_id>U114</user\_id><song\_id>S209</song\_id><artist\_id>A303</artist\_id><timestamp>2016-06-09 22:12:36</timestamp><start\_ts>2016-05-10 12:24:22</start\_ts><end\_ts>2017-05-09 08:09:22</end\_ts><geo\_cd>U</geo\_cd><station\_id>ST411</station\_id><song\_end\_type>2</song\_end\_type><like>1</like><dislike>0</dislike></record>

**song-artist.txt**

S200,A300

S201,A301

S202,A302

S203,A303

S204,A304

S205,A301

S206,A302

S207,A303

S208,A304

S209,A305

**stn-geocd.txt**

ST400,A

ST401,AU

ST402,AP

ST403,J

ST404,E

ST405,A

ST406,AU

ST407,AP

ST408,E

ST409,E

ST410,A

ST411,A

ST412,AP

ST413,J

ST414,E

**user-artist.txt**

U100,A300&A301&A302

U101,A301&A302

U102,A302

U103,A303&A301&A302

U104,A304&A301

U105,A305&A301&A302

U106,A301&A302

U107,A302

U108,A300&A303&A304

U109,A301&A303

U110,A302&A301

U111,A303&A301

U112,A304&A301

U113,A305&A302

U114,A300&A301&A302

**user-subscn.txt**

U100,1465230523,1465130523

U101,1465230523,1475130523

U102,1465230523,1475130523

U103,1465230523,1475130523

U104,1465230523,1475130523

U105,1465230523,1475130523

U106,1465230523,1485130523

U107,1465230523,1455130523

U108,1465230523,1465230623

U109,1465230523,1475130523

U110,1465230523,1475130523

U111,1465230523,1475130523

U112,1465230523,1475130523

U113,1465230523,1485130523

U114,1465230523,1468130523

These are copied to the data directory within the Acadgild home directory

[acadgild@centos1 ~]$ cd Data

[acadgild@centos1 Data]$ tree

.

├── LookUp

│   ├── song-artist.txt

│   ├── stn-geocd.txt

│   ├── user-artist.txt

│   ├── user-subscn.txt

│   └── user-suscn.txt

├── Mob

│   └── file.txt

├── tree.txt

└── Web

└── file.xml

3 directories, 8 files

[acadgild@centos1 Data]$

# Data Ingestion

The following steps are to ingest the required data and to cleanse and to store for analysation.

## Populate the Look-Up tables

The following script populate-lookup.sh that is used to load the data for the lookup tables into HBase tables.

The following steps are performed:

* Get the batch id number from the batch file and get the **Log File** for the batch using the batch id. This will be **log\_batch\_1**
* Add logs to the Log File signifying that the lookup tables are being created and populated
* Create the HBase tables for the lookup data files: **song-artist, stn-geocd** and **user-subscn** with their column families

All files are processed line by and added to the associated HBase table

Run the hive script user-artist.hql. This will populate a hive table with the data in the lookup data file **user-artist**. I tried to modify the file and its separators but still could not it to load into HBase.

**Populate -lookup.sh**

#!/bin/bash

batchid=`cat /home/acadgild/project/logs/current-batch.txt`

LOGFILE=/home/acadgild/project/logs/log\_batch\_$batchid

echo "Creating LookUp Tables" >> $LOGFILE

echo "create 'station-geo-map', 'geo'" | hbase shell

echo "create 'subscribed-users', 'subscn'" | hbase shell

echo "create 'song-artist-map', 'artist'" | hbase shell

echo "Populating LookUp Tables" >> $LOGFILE

file="/home/acadgild/project/lookupfiles/stn-geocd.txt"

while IFS= read -r line

do

stnid=`echo $line | cut -d',' -f1`

geocd=`echo $line | cut -d',' -f2`

echo "put 'station-geo-map', '$stnid', 'geo:geo\_cd', '$geocd'" | hbase shell

done <"$file"

file="/home/acadgild/project/lookupfiles/song-artist.txt"

while IFS= read -r line

do

songid=`echo $line | cut -d',' -f1`

artistid=`echo $line | cut -d',' -f2`

echo "put 'song-artist-map', '$songid', 'artist:artistid', '$artistid'" | hbase shell

done <"$file"

file="/home/acadgild/project/lookupfiles/user-subscn.txt"

while IFS= read -r line

do

userid=`echo $line | cut -d',' -f1`

startdt=`echo $line | cut -d',' -f2`

enddt=`echo $line | cut -d',' -f3`

echo "put 'subscribed-users', '$userid', 'subscn:startdt', '$startdt'" | hbase shell

echo "put 'subscribed-users', '$userid', 'subscn:enddt', '$enddt'" | hbase shell

done <"$file"

hive -f /home/acadgild/project/scripts/user-artist.hql

**Checking Results**

hbase(main):001:0> list

TABLE

song-artist-map

station-geo-map

subscribed-users

3 row(s) in 0.3220 seconds

=> ["song-artist-map", "station-geo-map", "subscribed-users"]

hbase(main):002:0> describe song-artist-map'

Table song-artist-map is ENABLED

song-artist-map

COLUMN FAMILIES DESCRIPTION

{NAME => 'artist', DATA\_BLOCK\_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICAT

ION\_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', T

TL => 'FOREVER', KEEP\_DELETED\_CELLS => 'FALSE', BLOCKSIZE => '65536', IN\_MEMORY

=> 'false', BLOCKCACHE => 'true'}

1 row(s) in 0.1990 seconds

hbase(main):003:0> describe station-geo-map'

Table station-geo-map is ENABLED

station-geo-map

COLUMN FAMILIES DESCRIPTION

{NAME => 'geo', DATA\_BLOCK\_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION

\_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', TTL

=> 'FOREVER', KEEP\_DELETED\_CELLS => 'FALSE', BLOCKSIZE => '65536', IN\_MEMORY =>

'false', BLOCKCACHE => 'true'}

1 row(s) in 0.0400 seconds

hbase(main):004:0> describe 'subscribed-users'

Table subscribed-users is ENABLED

subscribed-users

COLUMN FAMILIES DESCRIPTION

{NAME => 'subscn', DATA\_BLOCK\_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICAT

ION\_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN\_VERSIONS => '0', T

TL => 'FOREVER', KEEP\_DELETED\_CELLS => 'FALSE', BLOCKSIZE => '65536', IN\_MEMORY

=> 'false', BLOCKCACHE => 'true'}

1 row(s) in 0.0410 seconds

hbase(main):005:0> scan 'song-artist-map'

ROW COLUMN+CELL

S200 column=artist:artistid, timestamp=1509193426938, value=A30

0

S201 column=artist:artistid, timestamp=1509193438773, value=A30

1

S202 column=artist:artistid, timestamp=1509193450940, value=A30

2

S203 column=artist:artistid, timestamp=1509193463285, value=A30

3

S204 column=artist:artistid, timestamp=1509193475952, value=A30

4

S205 column=artist:artistid, timestamp=1509193487982, value=A30

1

S206 column=artist:artistid, timestamp=1509193500426, value=A30

2

S207 column=artist:artistid, timestamp=1509193513057, value=A30

3

S208 column=artist:artistid, timestamp=1509193525211, value=A30

4

S209 column=artist:artistid, timestamp=1509193537176, value=A30

5

10 row(s) in 0.1540 seconds

hbase(main):006:0> scan station-geo-map'

ROW COLUMN+CELL

ST400 column=geo:geo\_cd, timestamp=1509193247512, value=A

ST401 column=geo:geo\_cd, timestamp=1509193260082, value=AU

ST402 column=geo:geo\_cd, timestamp=1509193270477, value=AP

ST403 column=geo:geo\_cd, timestamp=1509193282277, value=J

ST404 column=geo:geo\_cd, timestamp=1509193294097, value=E

ST405 column=geo:geo\_cd, timestamp=1509193305991, value=A

ST406 column=geo:geo\_cd, timestamp=1509193316103, value=AU

ST407 column=geo:geo\_cd, timestamp=1509193329756, value=AP

ST408 column=geo:geo\_cd, timestamp=1509193341908, value=E

ST409 column=geo:geo\_cd, timestamp=1509193354045, value=E

ST410 column=geo:geo\_cd, timestamp=1509193366462, value=A

ST411 column=geo:geo\_cd, timestamp=1509193379229, value=A

ST412 column=geo:geo\_cd, timestamp=1509193391158, value=AP

ST413 column=geo:geo\_cd, timestamp=1509193403054, value=J

ST414 column=geo:geo\_cd, timestamp=1509193415185, value=E

15 row(s) in 0.0700 seconds

hbase(main):007:0>scan 'subscribed-users'

ROW COLUMN+CELL

U100 column=subscn:enddt, timestamp=1509193561232, value=146513

0523

U100 column=subscn:startdt, timestamp=1509193549016, value=1465

230523

U101 column=subscn:enddt, timestamp=1509193589876, value=147513

0523

U101 column=subscn:startdt, timestamp=1509193576294, value=1465

230523

U102 column=subscn:enddt, timestamp=1509193618216, value=147513

0523

U102 column=subscn:startdt, timestamp=1509193603891, value=1465

230523

U103 column=subscn:enddt, timestamp=1509193645516, value=147513

0523

U103 column=subscn:startdt, timestamp=1509193631629, value=1465

230523

U104 column=subscn:enddt, timestamp=1509193679144, value=147513

0523

U104 column=subscn:startdt, timestamp=1509193661987, value=1465

230523

U105 column=subscn:enddt, timestamp=1509193710828, value=147513

0523

U105 column=subscn:startdt, timestamp=1509193694626, value=1465

230523

U106 column=subscn:enddt, timestamp=1509193744903, value=148513

0523

U106 column=subscn:startdt, timestamp=1509193728556, value=1465

230523

U107 column=subscn:enddt, timestamp=1509193777459, value=145513

0523

U107 column=subscn:startdt, timestamp=1509193760632, value=1465

230523

U108 column=subscn:enddt, timestamp=1509193804775, value=146523

0623

U108 column=subscn:startdt, timestamp=1509193792848, value=1465

230523

U109 column=subscn:enddt, timestamp=1509193828751, value=147513

0523

U109 column=subscn:startdt, timestamp=1509193816766, value=1465

230523

U110 column=subscn:enddt, timestamp=1509193852992, value=147513

0523

U110 column=subscn:startdt, timestamp=1509193840871, value=1465

230523

U111 column=subscn:enddt, timestamp=1509193878023, value=147513

0523

U111 column=subscn:startdt, timestamp=1509193865074, value=1465

230523

U112 column=subscn:enddt, timestamp=1509193907103, value=147513

0523

U112 column=subscn:startdt, timestamp=1509193888628, value=1465

230523

U113 column=subscn:enddt, timestamp=1509193933199, value=148513

0523

U113 column=subscn:startdt, timestamp=1509193921063, value=1465

230523

U114 column=subscn:enddt, timestamp=1509193957453, value=146813

0523

U114 column=subscn:startdt, timestamp=1509193945192, value=1465

230523

15 row(s) in 0.1290 seconds

## Perform Data Formatting

The following scripts **dataformatting.sh** performs the following actions:

* Format the web xml data using **Pig** to a csv format.
* Load the 2 data files, **mob** and **web** (ingested by Pig), to a Hive Table.

The script performs some file and directory management then calls the 2 more scripts

* Run the pig script **dataformatting.pig.** This will format the web data (stored in the web folder in the HDFS) in xml format to csv format and store it in the HDFS in the folder formattedweb.
* Run the hive script **formatted\_hive\_load.hql.** This will load the data in the mob folder and formattedweb folder in the HDFS to a table formatted\_input in Hive

**dataformatting\_hadoop.sh**

#!/bin/bash

batchid=`cat /home/acadgild/project/logs/current-batch.txt`

LOGFILE=/home/acadgild/project/logs/log\_batch\_$batchid

echo "Placing data files from local to HDFS..." >> $LOGFILE

hadoop fs -rm -r /user/acadgild/project/batch${batchid}/web/

hadoop fs -rm -r /user/acadgild/project/batch${batchid}/formattedweb/

hadoop fs -rm -r /user/acadgild/project/batch${batchid}/mob/

hadoop fs -mkdir -p /user/acadgild/project/batch${batchid}/web/

hadoop fs -mkdir -p /user/acadgild/project/batch${batchid}/mob/

hadoop fs -put /home/acadgild/project/data/web/\* /user/acadgild/project/batch${batchid}/web/

hadoop fs -put /home/acadgild/project/data/mob/\* /user/acadgild/project/batch${batchid}/mob/

echo "Running pig script for data formatting..." >> $LOGFILE

pig -param batchid=$batchid /home/acadgild/project/scripts/dataformatting.pig

echo "Running hive script for formatted data load..." >> $LOGFILE

hive -hiveconf batchid=$batchid -f /home/acadgild/project/scripts/formatted\_hive\_load.hql

**Dataformating.pig**

REGISTER /home/acadgild/project/lib/piggybank.jar;

DEFINE XPath org.apache.pig.piggybank.evaluation.xml.XPath();

A = LOAD '/user/acadgild/project/batch${batchid}/web/' using org.apache.pig.piggybank.storage.XMLLoader('record') as (x:chararray);

B = FOREACH A GENERATE TRIM(XPath(x, 'record/user\_id')) AS user\_id,

TRIM(XPath(x, 'record/song\_id')) AS song\_id,

TRIM(XPath(x, 'record/artist\_id')) AS artist\_id,

ToUnixTime(ToDate(TRIM(XPath(x, 'record/timestamp')),'yyyy-MM-dd HH:mm:ss')) AS timestamp,

ToUnixTime(ToDate(TRIM(XPath(x, 'record/start\_ts')),'yyyy-MM-dd HH:mm:ss')) AS start\_ts,

ToUnixTime(ToDate(TRIM(XPath(x, 'record/end\_ts')),'yyyy-MM-dd HH:mm:ss')) AS end\_ts,

TRIM(XPath(x, 'record/geo\_cd')) AS geo\_cd,

TRIM(XPath(x, 'record/station\_id')) AS station\_id,

TRIM(XPath(x, 'record/song\_end\_type')) AS song\_end\_type,

TRIM(XPath(x, 'record/like')) AS like,

TRIM(XPath(x, 'record/dislike')) AS dislike;

STORE B INTO '/user/acadgild/project/batch${batchid}/formattedweb/' USING PigStorage(',');

**Formatted\_hive\_load.hql**

USE project;

CREATE TABLE IF NOT EXISTS formatted\_input

(

User\_id STRING,

Song\_id STRING,

Artist\_id STRING,

Timestamp STRING,

Start\_ts STRING,

End\_ts STRING,

Geo\_cd STRING,

Station\_id STRING,

Song\_end\_type INT,

Like INT,

Dislike INT

)

PARTITIONED BY

(batchid INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',';

LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/formattedweb/'

INTO TABLE formatted\_input PARTITION (batchid=${hiveconf:batchid});

LOAD DATA INPATH '/user/acadgild/project/batch${hiveconf:batchid}/mob/'

INTO TABLE formatted\_input PARTITION (batchid=${hiveconf:batchid});

**Checking Results**

hive> show tables;

OK

formatted\_input

song\_artist\_map

station\_geo\_map

subscribed\_users

users\_artists

Time taken: 0.027 seconds, Fetched: 5 row(s)

hive> select \* from formated\_input;

OK

U117 S204 A301 1495130523 1465130523 1475130523 A ST402 0 1 0 1

U115 S203 A305 1465230523 1465130523 1475130523 AP ST409 0 1 0 1

U117 S208 A305 1465130523 1465130523 1465130523 AP ST407 3 0 1 1

U111 S206 A303 1465230523 1485130523 1465130523 U ST414 1 0 0 1

U119 S207 A301 1465230523 1475130523 1485130523 AU ST408 1 1 1 1

S209 A301 1465230523 1465230523 1485130523 U ST411 3 0 1 1

U112 S207 A302 1465230523 1465230523 1475130523 AU ST410 0 1 1 1

U118 S203 A304 1475130523 1465130523 1465230523 U ST403 0 0 0 1

U101 S204 A301 1475130523 1485130523 1485130523 ST411 2 0 1 1

U103 S207 1465230523 1465130523 1465130523 A ST400 1 1 1 1

U113 S202 A300 1465130523 1475130523 1475130523 U ST415 1 1 0 1

U104 S206 A303 1495130523 1465130523 1475130523 U ST401 1 1 1 1

U113 S207 A305 1495130523 1465130523 1485130523 AU ST402 0 0 1 1

U101 S206 A305 1465130523 1465230523 1465230523 AP ST415 3 0 0 1

U110 S202 A303 1495130523 1465130523 1465130523 AP ST413 0 0 1 1

U118 S208 A304 1465130523 1475130523 1465130523 E ST410 0 1 1 1

U118 S209 A305 1475130523 1465230523 1465230523 E ST400 0 0 0 1

U108 S200 A300 1495130523 1475130523 1465230523 U ST400 1 0 1 1

U105 S208 A300 1465130523 1475130523 1465230523 AU ST410 1 0 0 1

U118 S201 A304 1465230523 1475130523 1485130523 A ST408 2 1 1 1

U113 S205 A305 1462908262 1465535556 1462908262 AP ST407 3 0 1 1

U102 S200 A301 1494342562 1465535556 1465535556 A ST400 1 0 1 1

U115 S207 A301 1494342562 1468139889 1465535556 AU ST406 2 1 1 1

U110 S201 A300 1468139889 1462908262 1468139889 AU ST413 2 0 1 1

U102 S203 A305 1465535556 1494342562 1465535556 A ST414 2 0 0 1

S209 A304 1465535556 1462908262 1465535556 E ST412 0 0 1 1

U105 S203 A300 1462908262 1468139889 1468139889 U ST407 2 1 1 1

U113 S205 A303 1462908262 1468139889 1468139889 E ST415 2 0 1 1

U120 S205 A302 1494342562 1494342562 1494342562 ST400 0 1 0 1

U105 S210 1468139889 1462908262 1494342562 E ST410 1 0 1 1

U117 S206 A300 1468139889 1468139889 1465535556 A ST414 2 0 0 1

U114 S200 A301 1462908262 1468139889 1462908262 AP ST408 1 1 1 1

U110 S208 A303 1494342562 1468139889 1468139889 E ST405 1 0 1 1

U115 S201 A303 1465535556 1465535556 1494342562 AU ST407 2 1 1 1

U103 S209 A305 1465535556 1468139889 1468139889 AU ST408 3 0 1 1

U112 S210 A303 1494342562 1494342562 1462908262 AU ST408 2 1 0 1

U118 S202 A301 1468139889 1465535556 1468139889 AP ST414 0 0 1 1

U100 S200 A301 1462908262 1494342562 1494342562 AU ST408 2 0 0 1

U113 S210 A304 1468139889 1465535556 1494342562 E ST403 2 0 1 1

U104 S203 A300 1468139889 1468139889 1494342562 AU ST406 1 0 1 1

Time taken: 0.556 seconds, Fetched: 40 row(s)

hive> select \* from song\_artist\_map;

OK

S200 A300

S201 A301

S202 A302

S203 A303

S204 A304

S205 A301

S206 A302

S207 A303

S208 A304

S209 A305

Time taken: 0.447 seconds, Fetched: 10 row(s)

hive> select \* from station\_geo\_map;

OK

ST400 A

ST401 AU

ST402 AP

ST403 J

ST404 E

ST405 A

ST406 AU

ST407 AP

ST408 E

ST409 E

ST410 A

ST411 A

ST412 AP

ST413 J

ST414 E

Time taken: 0.237 seconds, Fetched: 15 row(s)

hive> select \* from subscribed\_users;

OK

U100 1465230523 1465130523

U101 1465230523 1475130523

U102 1465230523 1475130523

U103 1465230523 1475130523

U104 1465230523 1475130523

U105 1465230523 1475130523

U106 1465230523 1485130523

U107 1465230523 1455130523

U108 1465230523 1465230623

U109 1465230523 1475130523

U110 1465230523 1475130523

U111 1465230523 1475130523

U112 1465230523 1475130523

U113 1465230523 1485130523

U114 1465230523 1468130523

Time taken: 0.193 seconds, Fetched: 15 row(s)

hive> select \* from users\_artists;

OK

U100 ["A300","A301","A302"]

U101 ["A301","A302"]

U102 ["A302"]

U103 ["A303","A301","A302"]

U104 ["A304","A301"]

U105 ["A305","A301","A302"]

U106 ["A301","A302"]

U107 ["A302"]

U108 ["A300","A303","A304"]

U109 ["A301","A303"]

U110 ["A302","A301"]

U111 ["A303","A301"]

U112 ["A304","A301"]

U113 ["A305","A302"]

U114 ["A300","A301","A302"]

Time taken: 0.094 seconds, Fetched: 15 row(s)

## Perform Data Enrichment and Cleaning

The data enrichment is carried out in two steps:

Create lookup tables in Hive and import the data from the HBase lookup tables to them. This is performed by **data\_enrichment\_filtering\_schema.sh**

Perform the data enrichment to the data in formatted\_input using the lookup tables. This is done by shell script **data\_enrichment.sh**

The following scripst data\_enrichment\_filtering\_schema.sh where the following operations are performed:

* Get the batch id number from the batch file and get the **Log File** for the batch using the batch id. This will be **log\_batch\_1**
* Add logs to the Log File signifying that the Hive lookup tables are created from the HBase lookup tables.
* Run the hive script create\_hive\_hbase\_lookup.hql. This will create the lookup tables in Hive and import the data from the HBase lookup tables to the Hive lookup tables.

**Data enrichment**

#!/bin/bash

batchid=`cat /home/acadgild/project/logs/current-batch.txt`

LOGFILE=/home/acadgild/project/logs/log\_batch\_$batchid

VALIDDIR=/home/acadgild/project/processed\_dir/valid/batch\_$batchid

INVALIDDIR=/home/acadgild/project/processed\_dir/invalid/batch\_$batchid

echo "Running hive script for data enrichment and filtering..." >> $LOGFILE

hive -hiveconf batchid=$batchid -f /home/acadgild/project/scripts/data\_enrichment.hql

if [ ! -d "$VALIDDIR" ]

then

mkdir -p "$VALIDDIR"

fi

if [ ! -d "$INVALIDDIR" ]

then

mkdir -p "$INVALIDDIR"

fi

echo "Copying valid and invalid records in local file system..." >> $LOGFILE

hadoop fs -get /user/hive/warehouse/project.db/enriched\_data/batchid=$batchid/status=pass/\* $VALIDDIR

hadoop fs -get /user/hive/warehouse/project.db/enriched\_data/batchid=$batchid/status=fail/\* $INVALIDDIR

echo "Deleting older valid and invalid records from local file system..." >> $LOGFILE

find /home/acadgild/project/processed\_dir/ -mtime +7 -exec rm {} \;

**data\_enrichment.hql**

SET hive.auto.convert.join=false;

SET hive.exec.dynamic.partition.mode=nonstrict;

USE project;

CREATE TABLE IF NOT EXISTS enriched\_data

(

User\_id STRING,

Song\_id STRING,

Artist\_id STRING,

Timestamp STRING,

Start\_ts STRING,

End\_ts STRING,

Geo\_cd STRING,

Station\_id STRING,

Song\_end\_type INT,

Like INT,

Dislike INT

)

PARTITIONED BY

(batchid INT,

status STRING)

STORED AS ORC;

INSERT OVERWRITE TABLE enriched\_data

PARTITION (batchid, status)

SELECT

i.user\_id,

i.song\_id,

sa.artist\_id,

i.timestamp,

i.start\_ts,

i.end\_ts,

sg.geo\_cd,

i.station\_id,

IF (i.song\_end\_type IS NULL, 3, i.song\_end\_type) AS song\_end\_type,

IF (i.like IS NULL, 0, i.like) AS like,

IF (i.dislike IS NULL, 0, i.dislike) AS dislike,

i.batchid,

IF((i.like=1 AND i.dislike=1)

OR i.user\_id IS NULL

OR i.song\_id IS NULL

OR i.timestamp IS NULL

OR i.start\_ts IS NULL

OR i.end\_ts IS NULL

OR i.geo\_cd IS NULL

OR i.user\_id=''

OR i.song\_id=''

OR i.timestamp=''

OR i.start\_ts=''

OR i.end\_ts=''

OR i.geo\_cd=''

OR sg.geo\_cd IS NULL

OR sg.geo\_cd=''

OR sa.artist\_id IS NULL

OR sa.artist\_id='', 'fail', 'pass') AS status

FROM formatted\_input i

LEFT OUTER JOIN station\_geo\_map sg ON i.station\_id = sg.station\_id

LEFT OUTER JOIN song\_artist\_map sa ON i.song\_id = sa.song\_id

WHERE i.batchid=${hiveconf:batchid};

**Checking Results**

hive> select \* from formatted\_input;

OK

U117 S204 A301 1495130523 1465130523 1475130523 A ST402 0 1 0 1

U115 S203 A305 1465230523 1465130523 1475130523 AP ST409 0 1 0 1

U117 S208 A305 1465130523 1465130523 1465130523 AP ST407 3 0 1 1

U111 S206 A303 1465230523 1485130523 1465130523 U ST414 1 0 0 1

U119 S207 A301 1465230523 1475130523 1485130523 AU ST408 1 1 1 1

S209 A301 1465230523 1465230523 1485130523 U ST411 3 0 1 1

U112 S207 A302 1465230523 1465230523 1475130523 AU ST410 0 1 1 1

U118 S203 A304 1475130523 1465130523 1465230523 U ST403 0 0 0 1

U101 S204 A301 1475130523 1485130523 1485130523 ST411 2 0 1 1

U103 S207 1465230523 1465130523 1465130523 A ST400 1 1 1 1

U113 S202 A300 1465130523 1475130523 1475130523 U ST415 1 1 0 1

U104 S206 A303 1495130523 1465130523 1475130523 U ST401 1 1 1 1

U113 S207 A305 1495130523 1465130523 1485130523 AU ST402 0 0 1 1

U101 S206 A305 1465130523 1465230523 1465230523 AP ST415 3 0 0 1

U110 S202 A303 1495130523 1465130523 1465130523 AP ST413 0 0 1 1

U118 S208 A304 1465130523 1475130523 1465130523 E ST410 0 1 1 1

U118 S209 A305 1475130523 1465230523 1465230523 E ST400 0 0 0 1

U108 S200 A300 1495130523 1475130523 1465230523 U ST400 1 0 1 1

U105 S208 A300 1465130523 1475130523 1465230523 AU ST410 1 0 0 1

U118 S201 A304 1465230523 1475130523 1485130523 A ST408 2 1 1 1

U113 S205 A305 1462908262 1465535556 1462908262 AP ST407 3 0 1 1

U102 S200 A301 1494342562 1465535556 1465535556 A ST400 1 0 1 1

U115 S207 A301 1494342562 1468139889 1465535556 AU ST406 2 1 1 1

U110 S201 A300 1468139889 1462908262 1468139889 AU ST413 2 0 1 1

U102 S203 A305 1465535556 1494342562 1465535556 A ST414 2 0 0 1

S209 A304 1465535556 1462908262 1465535556 E ST412 0 0 1 1

U105 S203 A300 1462908262 1468139889 1468139889 U ST407 2 1 1 1

U113 S205 A303 1462908262 1468139889 1468139889 E ST415 2 0 1 1

U120 S205 A302 1494342562 1494342562 1494342562 ST400 0 1 0 1

U105 S210 1468139889 1462908262 1494342562 E ST410 1 0 1 1

U117 S206 A300 1468139889 1468139889 1465535556 A ST414 2 0 0 1

U114 S200 A301 1462908262 1468139889 1462908262 AP ST408 1 1 1 1

U110 S208 A303 1494342562 1468139889 1468139889 E ST405 1 0 1 1

U115 S201 A303 1465535556 1465535556 1494342562 AU ST407 2 1 1 1

U103 S209 A305 1465535556 1468139889 1468139889 AU ST408 3 0 1 1

U112 S210 A303 1494342562 1494342562 1462908262 AU ST408 2 1 0 1

U118 S202 A301 1468139889 1465535556 1468139889 AP ST414 0 0 1 1

U100 S200 A301 1462908262 1494342562 1494342562 AU ST408 2 0 0 1

U113 S210 A304 1468139889 1465535556 1494342562 E ST403 2 0 1 1

U104 S203 A300 1468139889 1468139889 1494342562 AU ST406 1 0 1 1

Time taken: 0.575 seconds, Fetched: 40 row(s)

hive> select \* from enriched\_data;

OK

U114 S200 A300 1462908262 1468139889 1462908262 E ST408 1 1 1 1 fail

U118 S201 A301 1465230523 1475130523 1485130523 E ST408 2 1 1 1 fail

U115 S201 A301 1465535556 1465535556 1494342562 AP ST407 2 1 1 1 fail

U113 S202 A302 1465130523 1475130523 1475130523 NULL ST415 1 1 0 1 fail

U105 S203 A303 1462908262 1468139889 1468139889 AP ST407 2 1 1 1 fail

U101 S204 A304 1475130523 1485130523 1485130523 A ST411 2 0 1 1 fail

U113 S205 A301 1462908262 1468139889 1468139889 NULL ST415 2 0 1 1 fail

U120 S205 A301 1494342562 1494342562 1494342562 A ST400 0 1 0 1 fail

U101 S206 A302 1465130523 1465230523 1465230523 NULL ST415 3 0 0 1 fail

U104 S206 A302 1495130523 1465130523 1475130523 AU ST401 1 1 1 1 fail

U112 S207 A303 1465230523 1465230523 1475130523 A ST410 0 1 1 1 fail

U103 S207 A303 1465230523 1465130523 1465130523 A ST400 1 1 1 1 fail

U119 S207 A303 1465230523 1475130523 1485130523 E ST408 1 1 1 1 fail

U115 S207 A303 1494342562 1468139889 1465535556 AU ST406 2 1 1 1 fail

U118 S208 A304 1465130523 1475130523 1465130523 A ST410 0 1 1 1 fail

S209 A305 1465230523 1465230523 1485130523 A ST411 3 0 1 1 fail

S209 A305 1465535556 1462908262 1465535556 AP ST412 0 0 1 1 fail

U113 S210 NULL 1468139889 1465535556 1494342562 J ST403 2 0 1 1 fail

U105 S210 NULL 1468139889 1462908262 1494342562 A ST410 1 0 1 1 fail

U112 S210 NULL 1494342562 1494342562 1462908262 E ST408 2 1 0 1 fail

U102 S200 A300 1494342562 1465535556 1465535556 A ST400 1 0 1 1 pass

U108 S200 A300 1495130523 1475130523 1465230523 A ST400 1 0 1 1 pass

U100 S200 A300 1462908262 1494342562 1494342562 E ST408 2 0 0 1 pass

U110 S201 A301 1468139889 1462908262 1468139889 J ST413 2 0 1 1 pass

U110 S202 A302 1495130523 1465130523 1465130523 J ST413 0 0 1 1 pass

U118 S202 A302 1468139889 1465535556 1468139889 E ST414 0 0 1 1 pass

U118 S203 A303 1475130523 1465130523 1465230523 J ST403 0 0 0 1 pass

U104 S203 A303 1468139889 1468139889 1494342562 AU ST406 1 0 1 1 pass

U115 S203 A303 1465230523 1465130523 1475130523 E ST409 0 1 0 1 pass

U102 S203 A303 1465535556 1494342562 1465535556 E ST414 2 0 0 1 pass

U117 S204 A304 1495130523 1465130523 1475130523 AP ST402 0 1 0 1 pass

U113 S205 A301 1462908262 1465535556 1462908262 AP ST407 3 0 1 1 pass

U117 S206 A302 1468139889 1468139889 1465535556 E ST414 2 0 0 1 pass

U111 S206 A302 1465230523 1485130523 1465130523 E ST414 1 0 0 1 pass

U113 S207 A303 1495130523 1465130523 1485130523 AP ST402 0 0 1 1 pass

U117 S208 A304 1465130523 1465130523 1465130523 AP ST407 3 0 1 1 pass

U110 S208 A304 1494342562 1468139889 1468139889 A ST405 1 0 1 1 pass

U105 S208 A304 1465130523 1475130523 1465230523 A ST410 1 0 0 1 pass

U103 S209 A305 1465535556 1468139889 1468139889 E ST408 3 0 1 1 pass

U118 S209 A305 1475130523 1465230523 1465230523 A ST400 0 0 0 1 pass

Time taken: 0.172 seconds, Fetched: 40 row(s)

# Analysing Data

The following spark SQL script was created to resolve the required solutions. hive> select \* from formatted\_input;

**Code written**

**package** com.duncb.spark

**import** org.apache.spark.SparkContext.\_

**import** org.apache.hadoop.hive.serde2.`lazy`.LazySimpleSerDe

**import** org.apache.log4j.\_

**import** org.apache.spark.sql.\_

**import** org.apache.spark.sql.functions.\_

**import** org.apache.spark.sql.SparkSession

**object** MP1 {

**def** main(args: Array[*String*]): Unit = {

**val** sc = SparkSession.builder()

.master("local[2]")

.appName("MP1")

.config("spark.sql.warehouse.dir","/user/hive/warehouse")

.config("hive.metastore.uris","thrift://10.100.100.69:9083")

.enableHiveSupport()

.getOrCreate()

Logger.getLogger("org").setLevel(Level.ERROR)

**val** batchId = args(0)

Println(“ top 10 station\_id(s) where maximum number of songs were played, which were liked by unique users.”)

**val** set\_properties = sc.sqlContext.sql("set hive.auto.convert.join=false")

**val** use\_project\_database = sc.sqlContext.sql("USE project")

**val** create\_hive\_table\_top\_10\_stations = sc.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top\_10\_stations"+

"("+

" station\_id STRING,"+

" total\_distinct\_songs\_played INT,"+

" distinct\_user\_count INT"+

")"+

" PARTITIONED BY (batchid INT)"+

" ROW FORMAT DELIMITED"+

" FIELDS TERMINATED BY ','"+

" STORED AS TEXTFILE")

**val** insert\_into\_top\_10\_stations = sc.sqlContext.sql("INSERT OVERWRITE TABLE project.top\_10\_stations"+

s" PARTITION (batchid=1)"+

" SELECT"+

" station\_id,"+

" COUNT(DISTINCT song\_id) AS total\_distinct\_songs\_played,"+

" COUNT(DISTINCT user\_id) AS distinct\_user\_count"+

" FROM project.enriched\_data"+

" WHERE status='pass'"+

s" AND (batchid=1)"+

" AND like=1"+

" GROUP BY station\_id"+

" ORDER BY total\_distinct\_songs\_played DESC"+

" LIMIT 10")

println(“ total duration of songs played by each type of user, where type of user can be 'subscribed' or 'unsubscribed'.”)

**val** create\_hive\_table\_song\_duration = sc.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.song\_duration"+

"("+

" user\_id STRING,"+

" user\_type STRING,"+

" song\_id STRING,"+

" artist\_id STRING,"+

" total\_duration\_in\_minutes DOUBLE"+

")"+

" PARTITIONED BY (batchid INT)"+

" ROW FORMAT DELIMITED"+

" FIELDS TERMINATED BY ','"+

" STORED AS TEXTFILE")

**val** insert\_into\_song\_duration = sc.sqlContext.sql("INSERT OVERWRITE TABLE project.song\_duration"+

s" PARTITION (batchid=1)"+

" SELECT"+

" e.user\_id STRING,"+

" IF(e.user\_id!=s.user\_id"+

" OR (CAST(s.subscn\_end\_dt as BIGINT) < CAST(e.start\_ts as BIGINT)),'unsubscribed','subscribed') AS user\_type,"+

" e.song\_id STRING,"+

" e.artist\_id STRING,"+

" (cast(e.end\_ts as BIGINT)-cast(e.start\_ts as BIGINT))/60 AS total\_duration\_in\_minutes"+

" FROM project.enriched\_data e"+

" LEFT OUTER JOIN project.subscribed\_users\_1 s"+

" ON e.user\_id=s.user\_id"+

" WHERE e.status='pass'"+

s" AND (batchid=1)")

println(“Determine top 10 connected artists”).

**val** create\_hive\_table\_top\_10\_connected\_artists = sc.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.connected\_artists"+

"("+

" artist\_id STRING,"+

" total\_distinct\_songs INT,"+

" unique\_followers INT"+

")"+

" PARTITIONED BY (batchid INT)"+

" ROW FORMAT DELIMITED"+

" FIELDS TERMINATED BY ','"+

" STORED AS TEXTFILE")

**val** insert\_into\_top\_10\_connected\_artists = sc.sqlContext.sql("INSERT OVERWRITE TABLE project.connected\_artists"+

s" PARTITION (batchid=1)"+

" SELECT"+

" artist\_id,"+

" COUNT(DISTINCT song\_id) AS total\_distinct\_songs,"+

" COUNT(DISTINCT user\_id) AS unique\_followers"+

" FROM project.enriched\_data"+

" WHERE status='pass'"+

s" AND (batchid=1)"+

" GROUP BY artist\_id"+

" ORDER BY unique\_followers desc,total\_distinct\_songs desc"+

" LIMIT 10")

println(“top 10 songs who have generated the maximum revenue.”)

**val** create\_hive\_table\_top\_10\_songs\_maxrevenue = sc.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top\_10\_songs\_maxrevenue"+

"("+

" song\_id STRING,"+

" artist\_id STRING,"+

" total\_duration\_in\_minutes DOUBLE"+

" )"+

" PARTITIONED BY (batchid INT)"+

" ROW FORMAT DELIMITED"+

" FIELDS TERMINATED BY ','"+

" STORED AS TEXTFILE")

**val** insert\_into\_top\_10\_songs\_maxrevenue = sc.sqlContext.sql("INSERT OVERWRITE TABLE project.top\_10\_songs\_maxrevenue"+

s" PARTITION (batchid=1)"+

" SELECT"+

" song\_id,"+

" artist\_id,"+

" (cast(end\_ts as BIGINT)-cast(start\_ts as BIGINT))/60 AS total\_duration\_in\_minutes"+

" FROM project.enriched\_data"+

" WHERE status='pass'" +

s" AND (batchid=1)"+

" AND (like=1 OR song\_end\_type=0 OR (like=1 and song\_end\_type=0))"+

" ORDER BY total\_duration\_in\_minutes desc"+

" LIMIT 10")

println(“ top 10 unsubscribed users who listened to the songs for the longest duration.”)

**val** create\_hive\_table\_top\_10\_unsubscribed\_users = sc.sqlContext.sql("CREATE TABLE IF NOT EXISTS project.top\_10\_unsubscribed\_users"+

"("+

" user\_id STRING,"+

" song\_id STRING,"+

" artist\_id STRING,"+

" total\_duration\_in\_minutes DOUBLE"+

")"+

" PARTITIONED BY (batchid INT)"+

" ROW FORMAT DELIMITED"+

" FIELDS TERMINATED BY ','"+

" STORED AS TEXTFILE")

**val** insert\_into\_unsubscribed\_users = sc.sqlContext.sql("INSERT OVERWRITE TABLE project.top\_10\_unsubscribed\_users"+

s" PARTITION (batchid=1)"+

" SELECT"+

" user\_id,"+

" song\_id,"+

" artist\_id,"+

" total\_duration\_in\_minutes"+

" FROM project.song\_duration"+

" WHERE user\_type='unsubscribed'"+

" AND total\_duration\_in\_minutes>=0"+

s" AND (batchid=1)"+

" ORDER BY total\_duration\_in\_minutes desc"+

" LIMIT 10")

}

}

# Results

top 10 station\_id(s) where maximum number of songs were played, which were liked by unique users.

+------------+

| station\_id|

+------------+

| ST402|

| ST411|

| ST405|

| ST410|

+------------+

total duration of songs played by each type of user, where type of user can be 'subscribed' or 'unsubscribed'.

+------------+---------------------------------+

| user\_type| total\_song\_duration|

+------------+---------------------------------+

|subscribed| 1904665.6500000001|

+----------+-----------------------------------+

Determine top 10 connected artists

+----------+

| artist\_id|

+----------+

| A300|

| A303|

| A304|

| A305|

| A302|

+----------+

top 10 songs who have generated the maximum revenue.

+----------+

| song\_id|

+----------+

| S209|

| S202|

| S205|

| S200|

| S203|

| S203|

| S206|

| S202|

| S206|

| S202|

+---------+

top 10 unsubscribed users who listened to the songs for the longest duration.

+---------+

| user\_id|

+---------+

| U115|

| U108|

| U120|

| U107|

| U101|

| U110|

| U118|

| U104|

| U102|

| U113|

+---------+

# Automation

The following scripts start all required services then process the data as shown during the whole project.

**Start daemons.sh**

batchid=`cat /home/acadgild/project/logs/current-batch.txt`

LOGFILE=/home/acadgild/project/logs/log\_batch\_$batchid

chmod 777 /home/acadgild/project/logs/\*

echo "Starting daemons..." >> $LOGFILE

start-all.sh

start-hbase.sh

mr-jobhistory-daemon.sh start historyserver

sudo service mysqld start

hive --service metastore

**Wrapper.sh**

#!/bin/bash

#This script calls other scripts in sequential fashion

sh /home/acadgild/project/scripts/start-daemons.sh

sh /home/acadgild/project/scripts/populate-lookup.sh

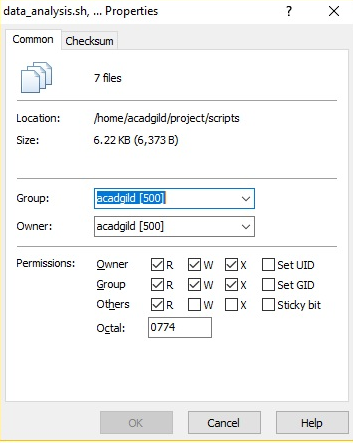
sh /home/acadgild/project/scripts/data\_enrichment\_filtering\_schema.sh

sh /home/acadgild/project/scripts/dataformatting\_hadoop.sh

sh /home/acadgild/project/scripts/data\_enrichment.sh

sh /home/acadgild/project/scripts/data\_analysis.sh

Permissions set on scripts to enable execution



To allow all to run automatically a cron job is created to run every 3 hours.

