**Music Data Analysis**

**Project Description:**

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.

**Data Creation:**

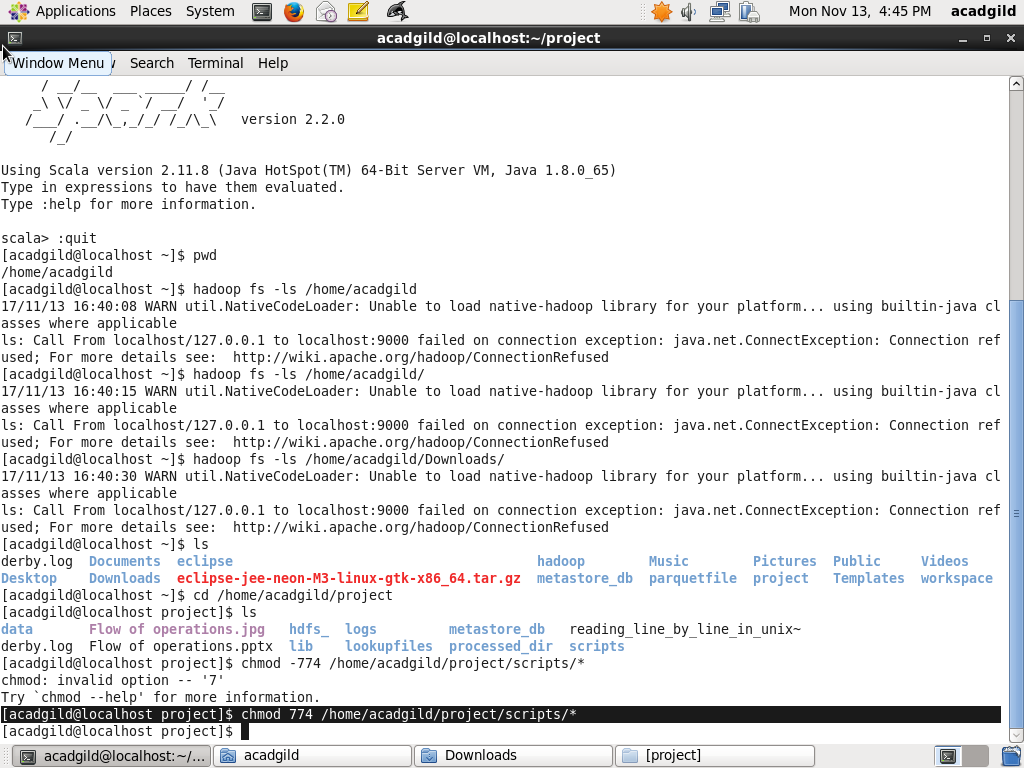
Run mysql

sudo sevice mysqld start

Make sure all the scripts in Scripts have 774 permissions

Run below command:

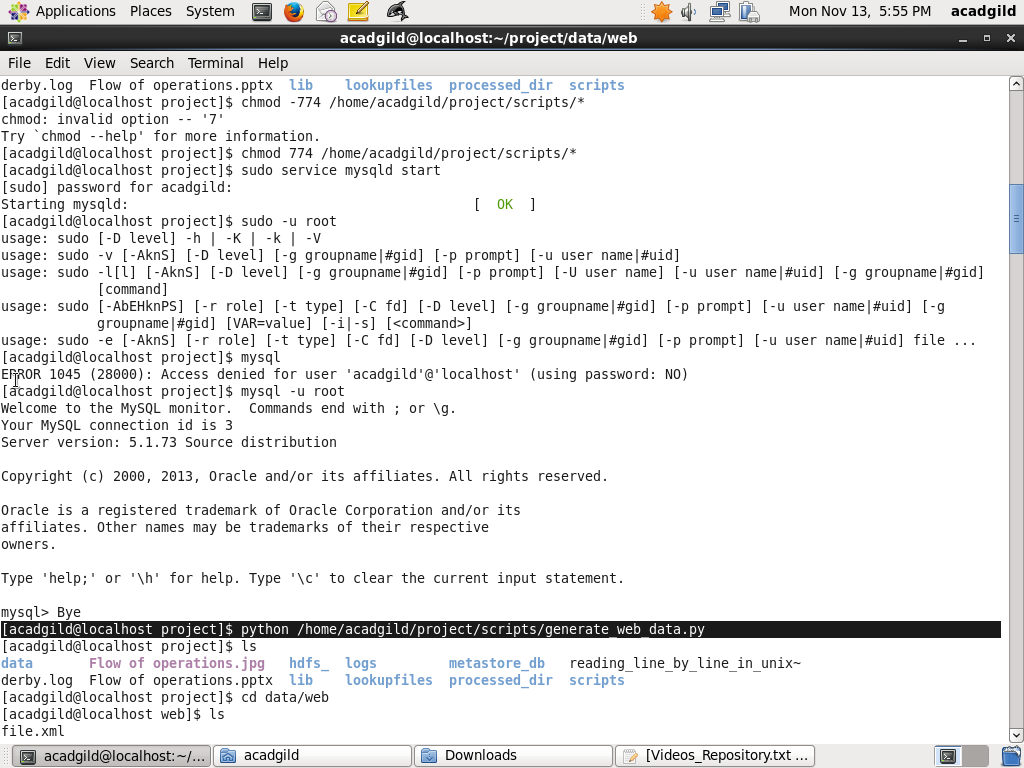
Chmod 774 “path”

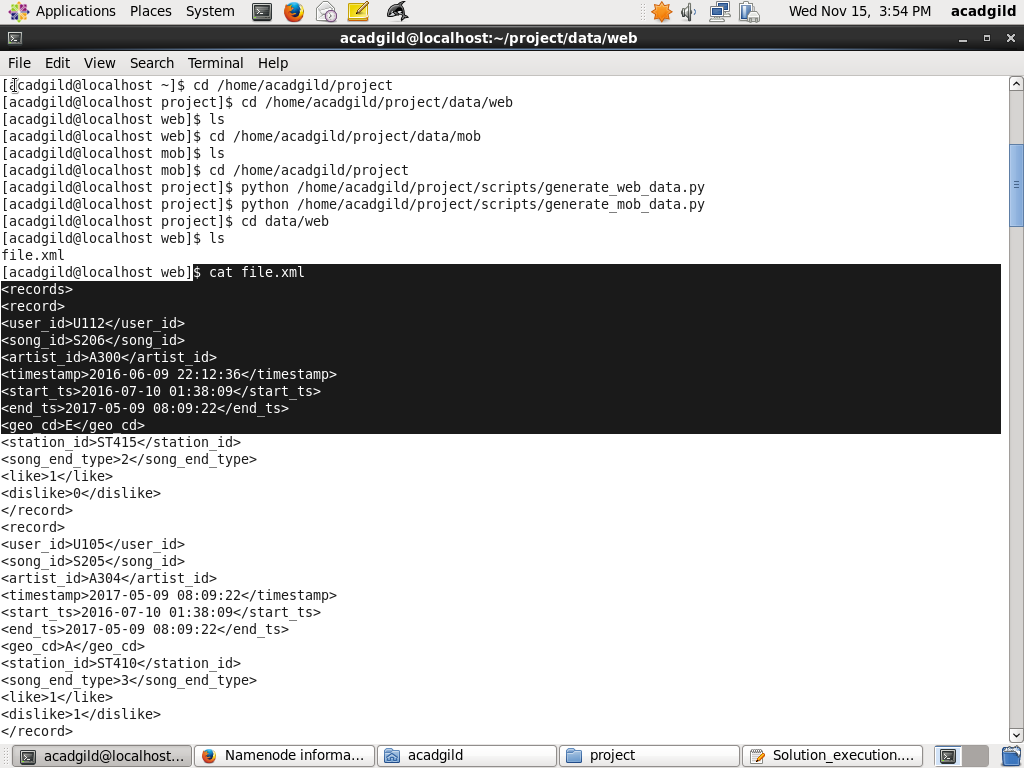


1. Generate Some Random Data coming from Web Application.(using python or UNIX scripting)

Sol: python /home/acadgild/project/scripts/generate\_web\_data.py

(Contains the set of records created Manually which contains both valid and Invalid Records in XML format )

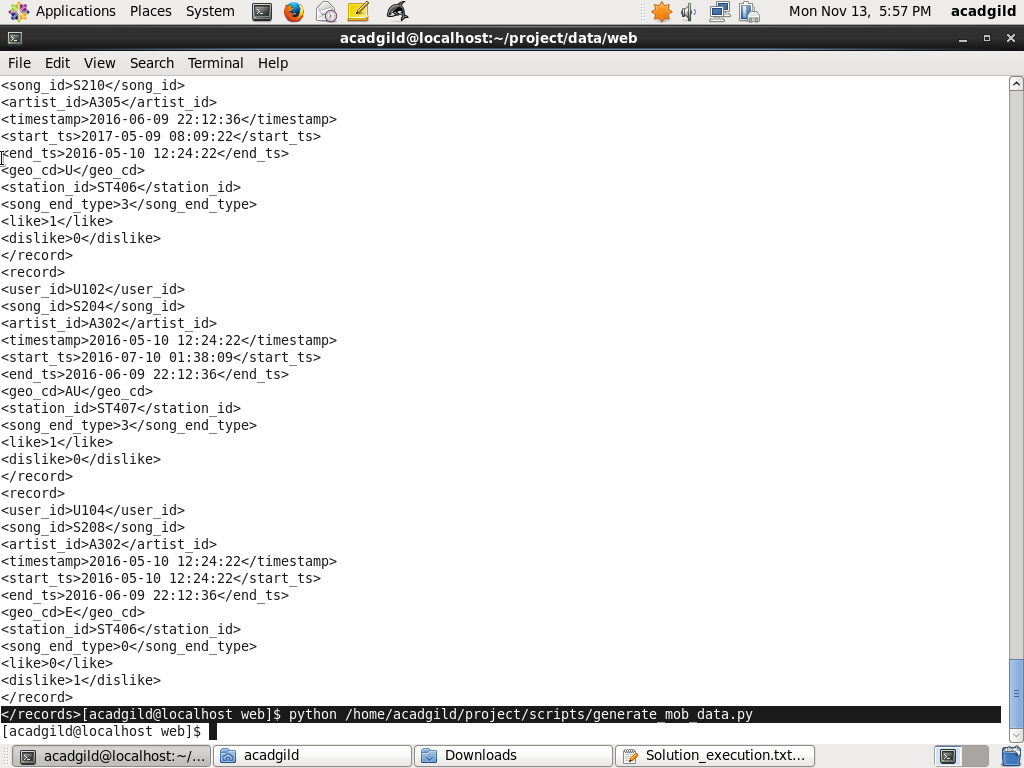


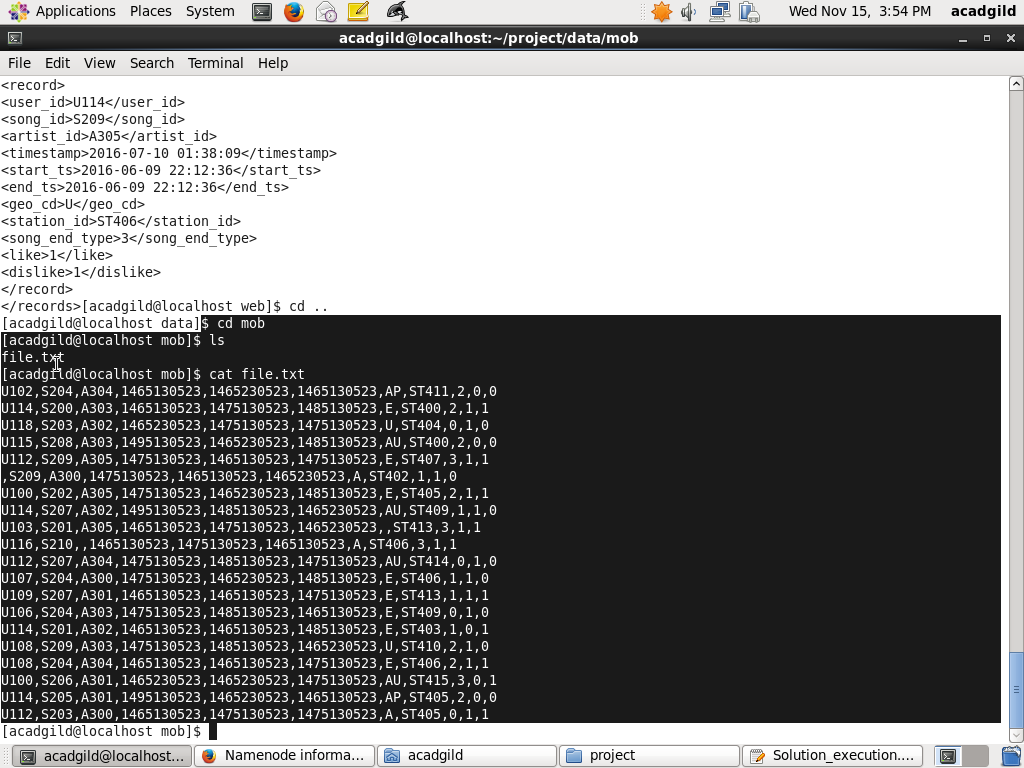


2. Generate some random data coming from mobile application

Sol: python /home/acadgild/project/scripts/generate\_mob\_data.py

(Contains the set of records created Manually which contains both valid and Invalid Records in CSV format)





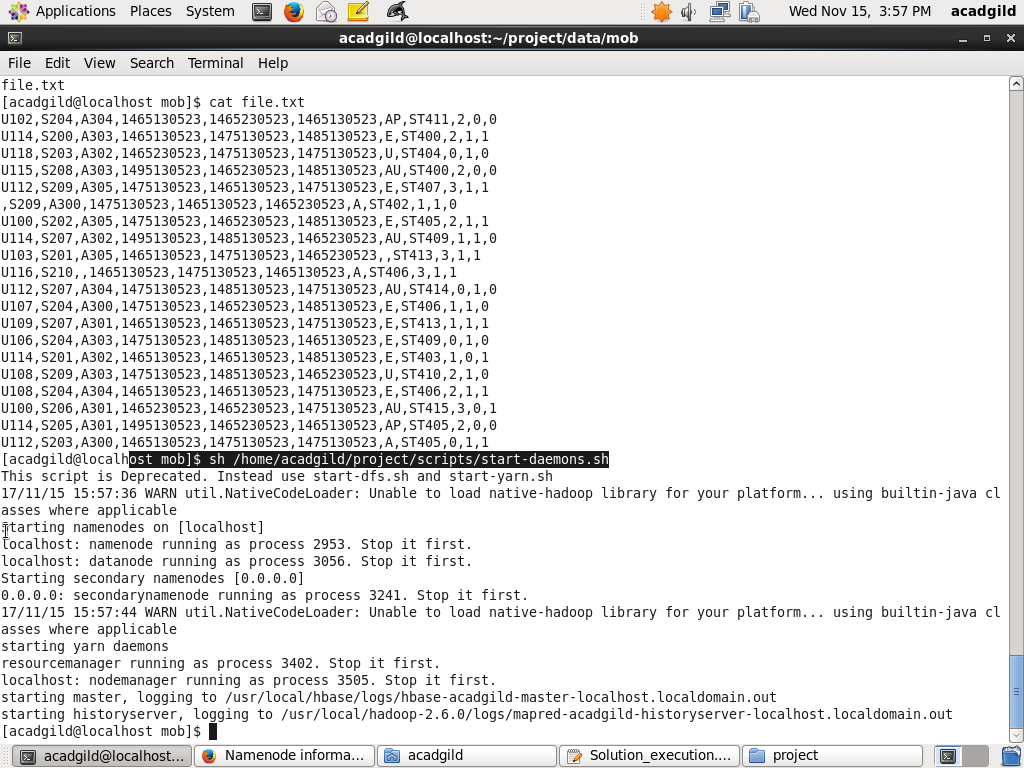
**Data Ingestion and Validation**

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. 3. Data present in lookup directory should be used in HBase.
3. All the Timestamps must have the format of long Integer interpreted as Unix Timestamps.
4. If both Like and dislike are 1 ,consider it to be Invalid.

* Launch all necessary daemons

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

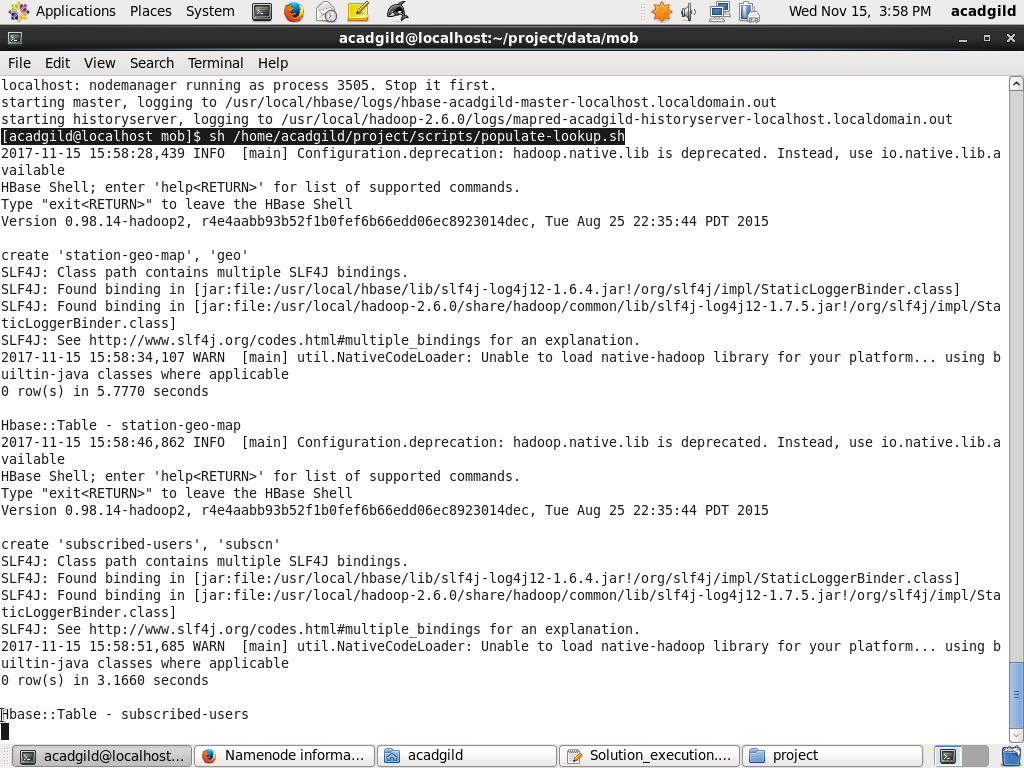
(Start-deamons.sh contains the batch id..To run the same job again n again we need BatchId for differentiating the jobs)

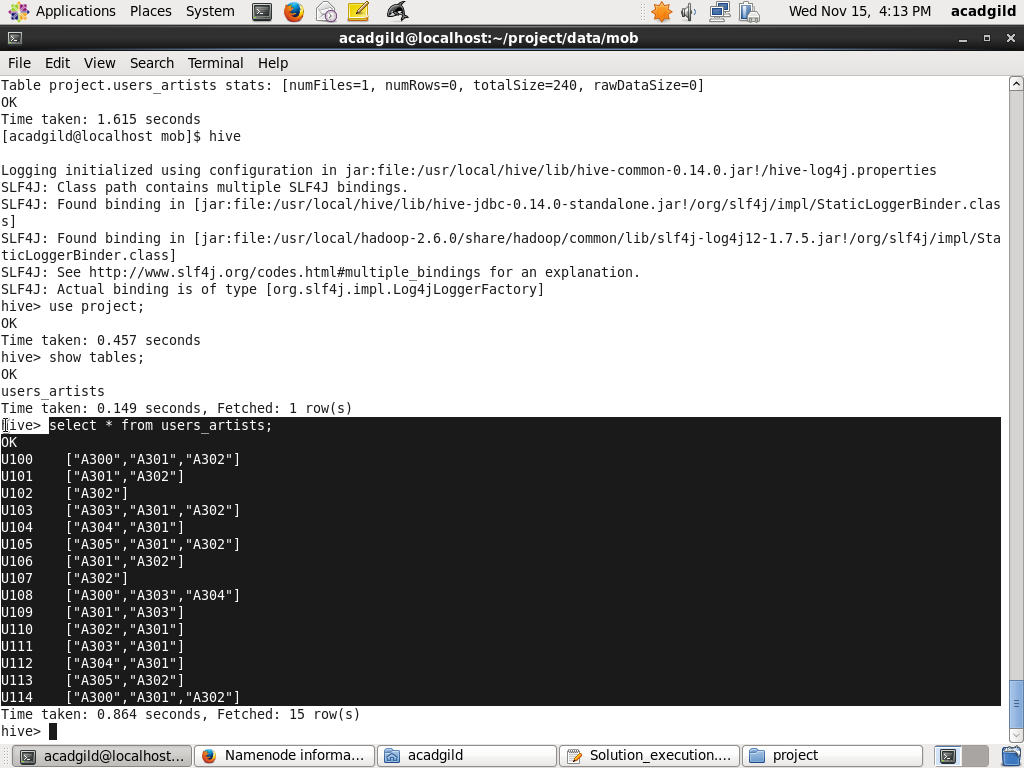


* Populates lookup tables

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

(Creating Required tables and loading the data from lookup files )

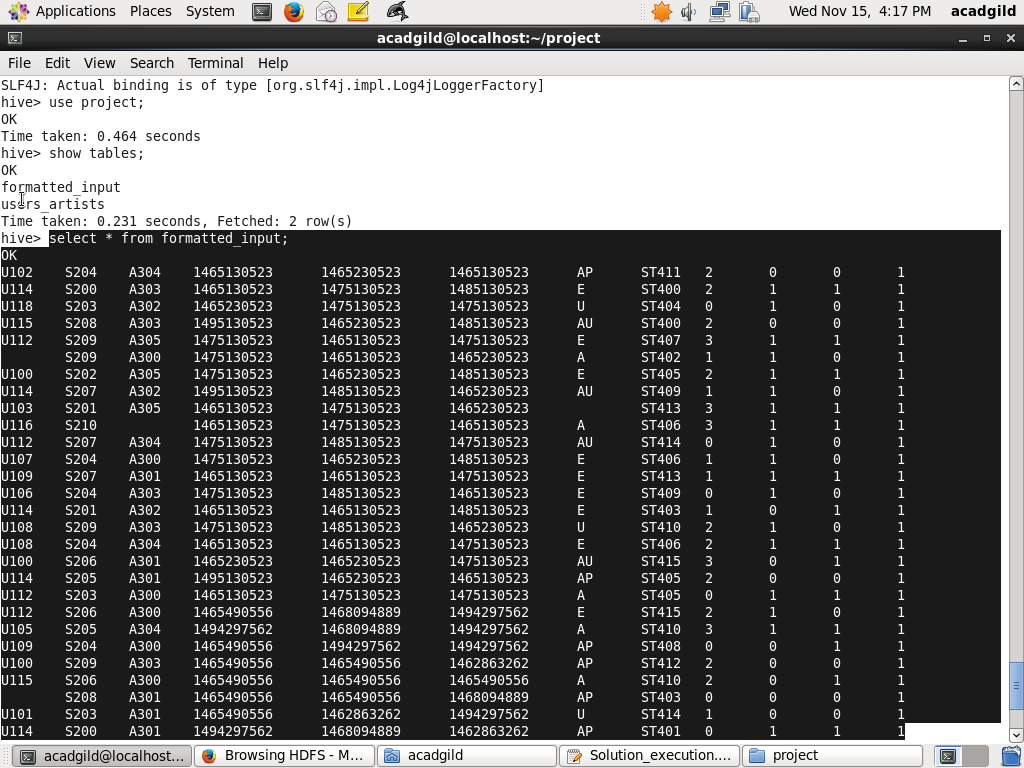


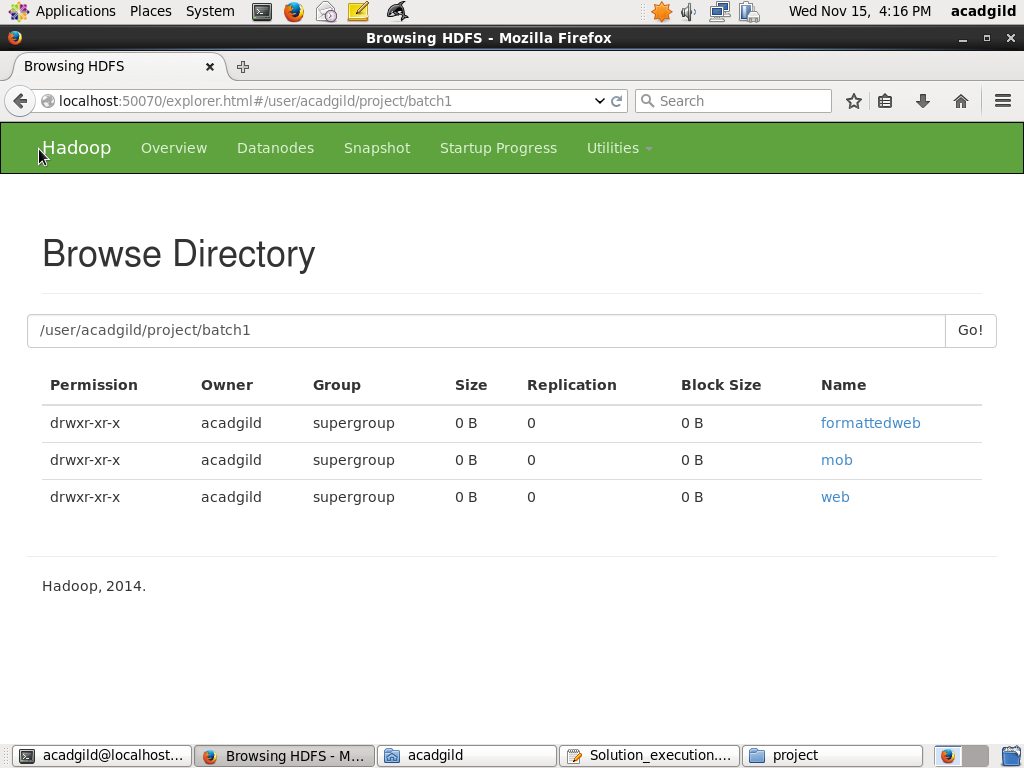


* dataformatting.sh -- Performs data formatting

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

(Loading the files from Local(web and mob )to hdfs)





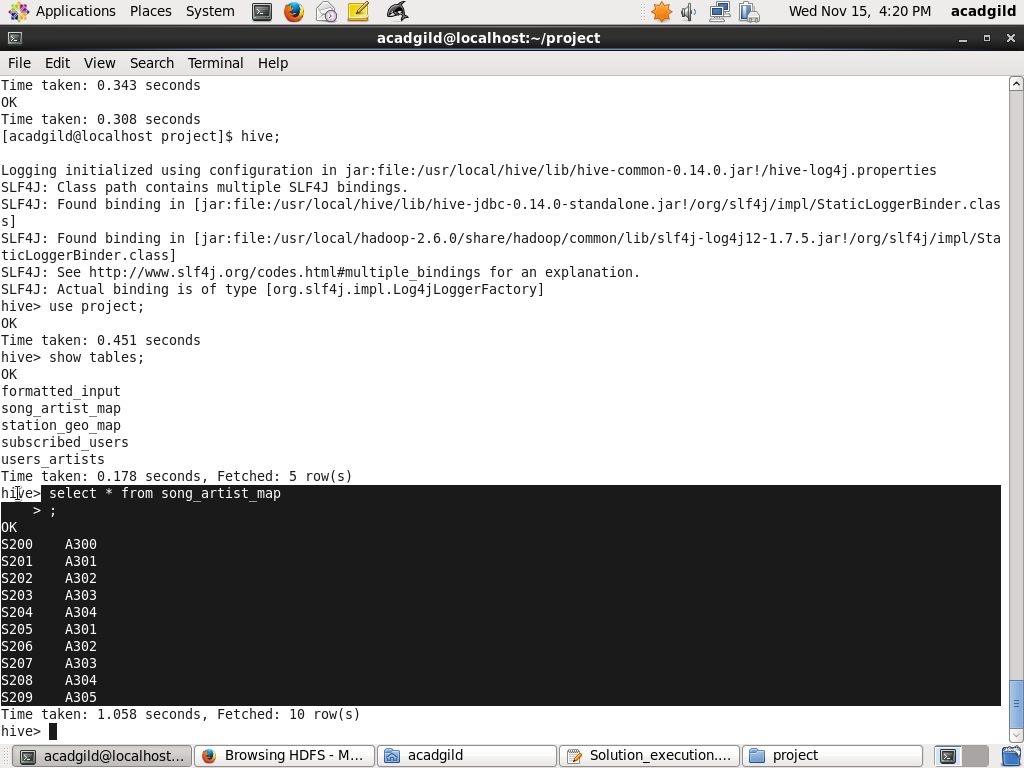
Pig query for Satisfying the Condition 4.

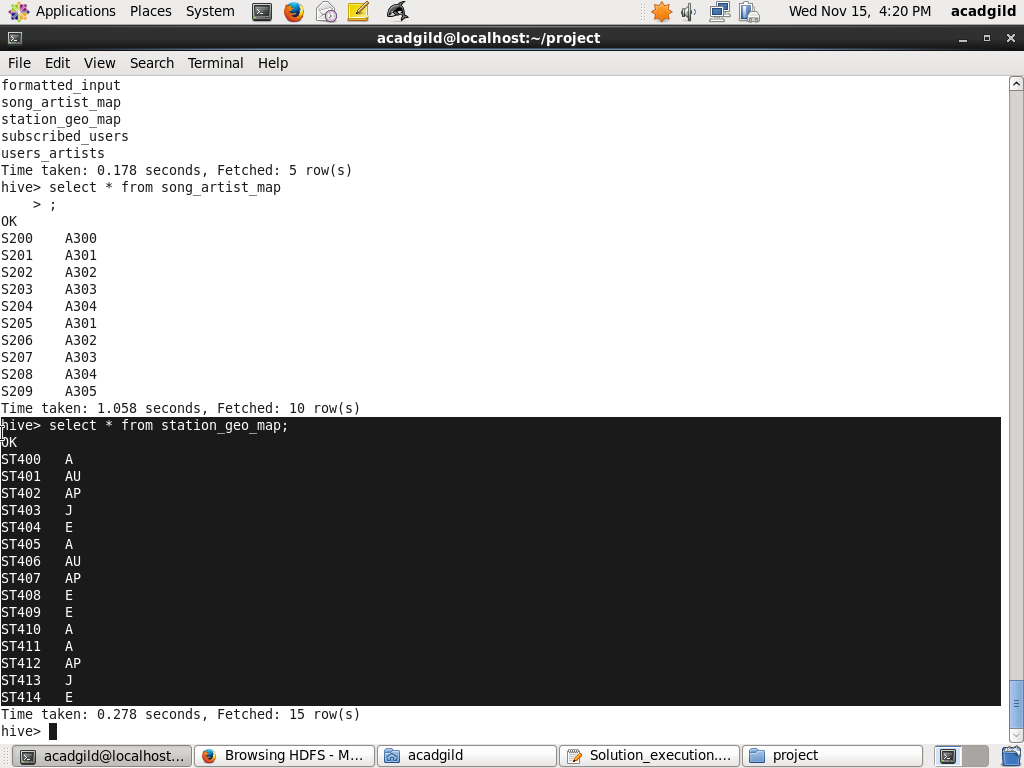
**Data Enrichment:(Rules)**

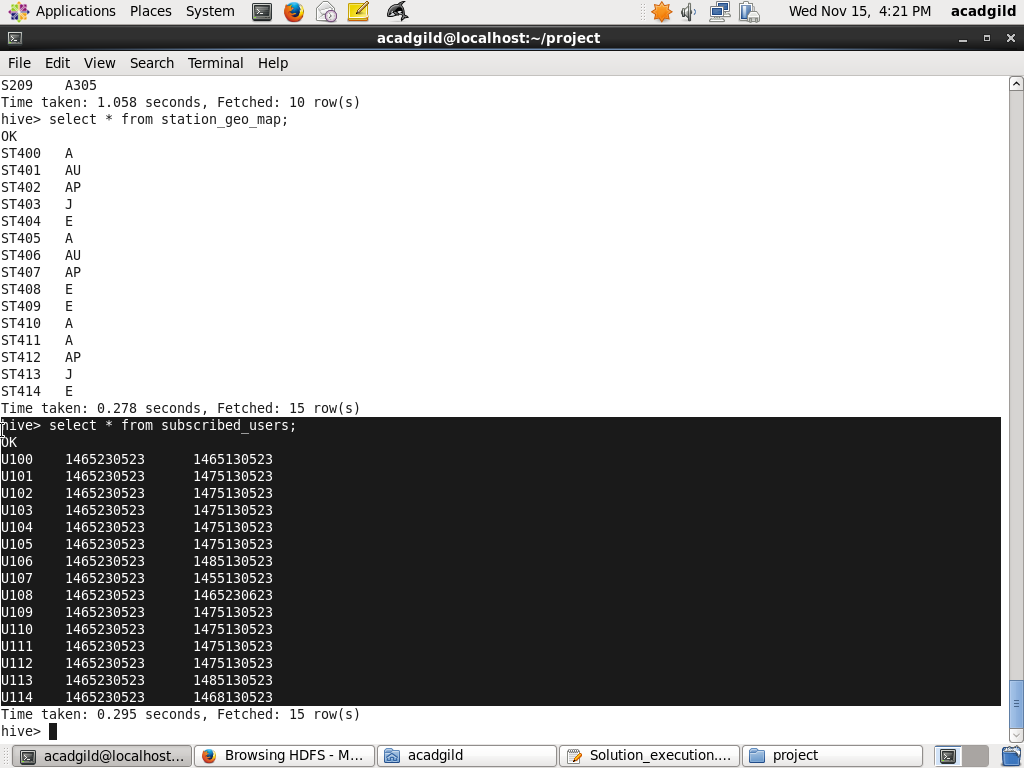
1. If any of the like or Dislike is Null or absent ,consider it as 0.
2. If the fields are absent consider lookup table for fields to get the values.
3. If corresponding lookup entry is not found ,consider that record to be invalid.

Lookup Table is in **Hbase** and Data is present in **hdfs**.

hive -f /home/acadgild/project/scripts/create\_hive\_hbase\_lookup.hql



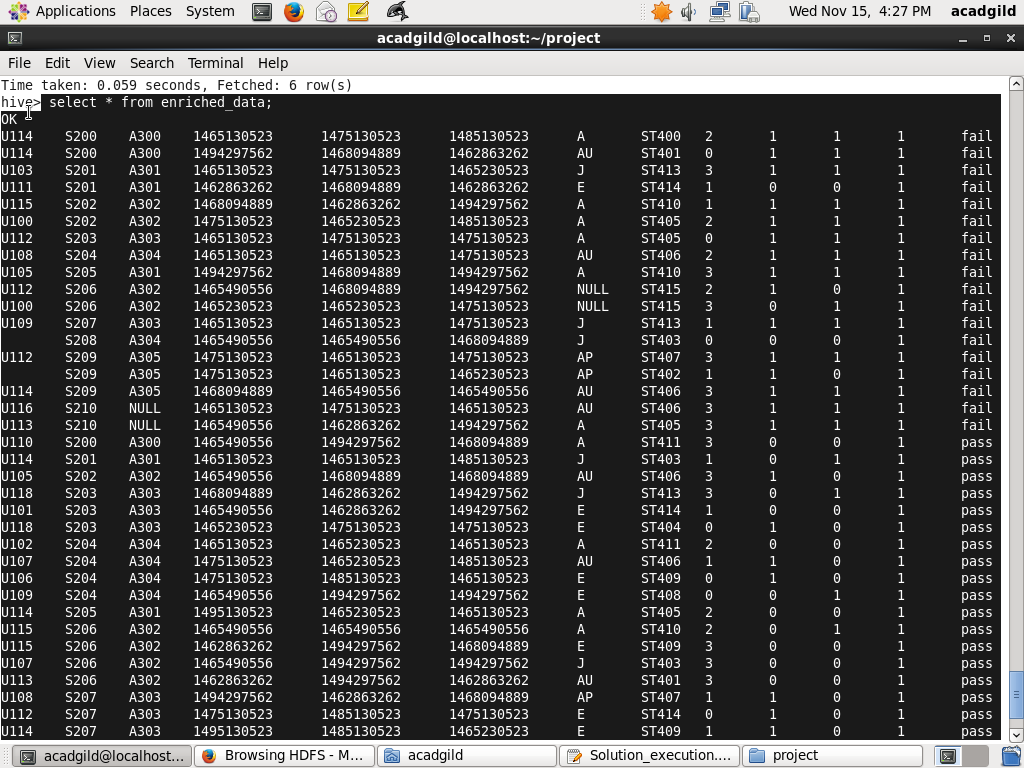




(Creating hive table that consists data of Hbase)

data\_enrichment.sh -- Performs data enrichment and cleaning

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



(Create a table which loads the data from previous table and check for null and lookup table if the value is present if not Assume that record to be invalid)

**Data Analysis:**

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

**Sol:**

CREATE TABLE IF NOT EXISTS 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;

INSERT OVERWRITE TABLE top\_10\_stations

PARTITION(batchid=${hiveconf:batchid})

SELECT

station\_id,

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

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

FROM enriched\_data

WHERE status='pass'

AND batchid=${hiveconf:batchid}

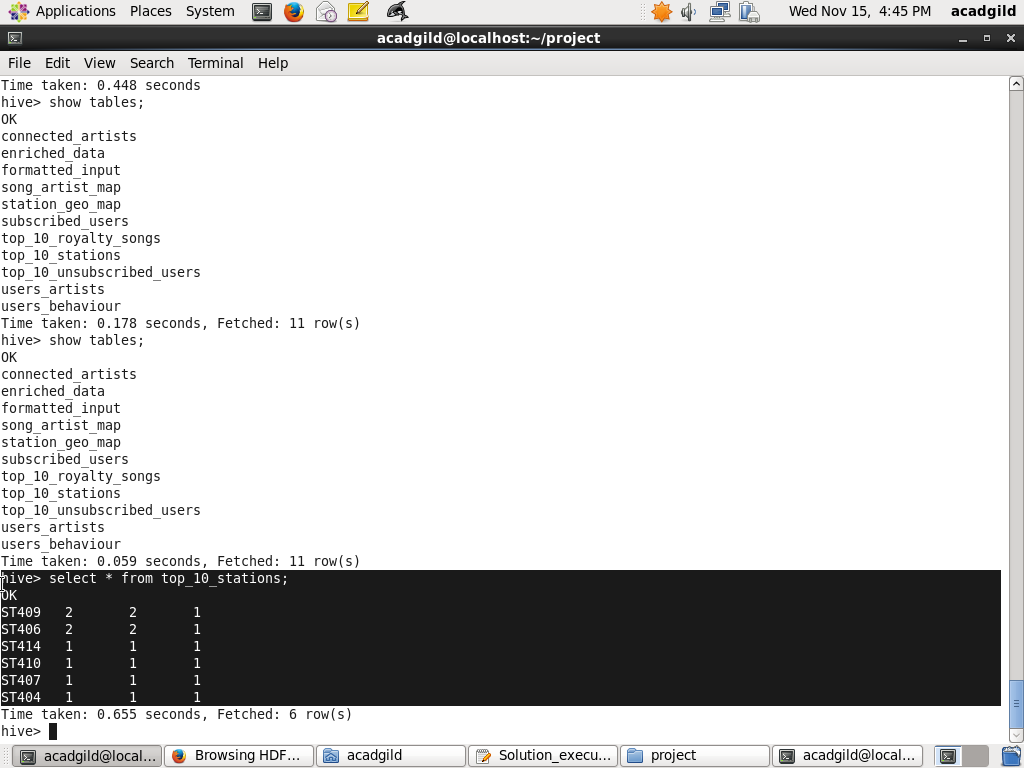
AND like=1

GROUP BY station\_id

ORDER BY total\_distinct\_songs\_played DESC

LIMIT 10;

**Output:**

****

1. **Determine total duration of songs played by each type of user, where type of user can be 'subscribed' or 'unsubscribed'. An unsubscribed user is the one whose record is either not present in Subscribed\_users lookup table or has subscription\_end\_date earlier than the timestamp of the song played by him.**

**Sol:**

CREATE TABLE IF NOT EXISTS users\_behaviour

(

user\_type STRING,

duration INT

)

PARTITIONED BY (batchid INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

INSERT OVERWRITE TABLE users\_behaviour

PARTITION(batchid=${hiveconf:batchid})

SELECT

CASE WHEN (su.user\_id IS NULL OR CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn\_end\_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED'

WHEN (su.user\_id IS NOT NULL AND CAST(ed.timestamp AS DECIMAL(20,0)) <= CAST(su.subscn\_end\_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED'

END AS user\_type,

SUM(ABS(CAST(ed.end\_ts AS DECIMAL(20,0))-CAST(ed.start\_ts AS DECIMAL(20,0)))) AS duration

FROM enriched\_data ed

LEFT OUTER JOIN subscribed\_users su

ON ed.user\_id=su.user\_id

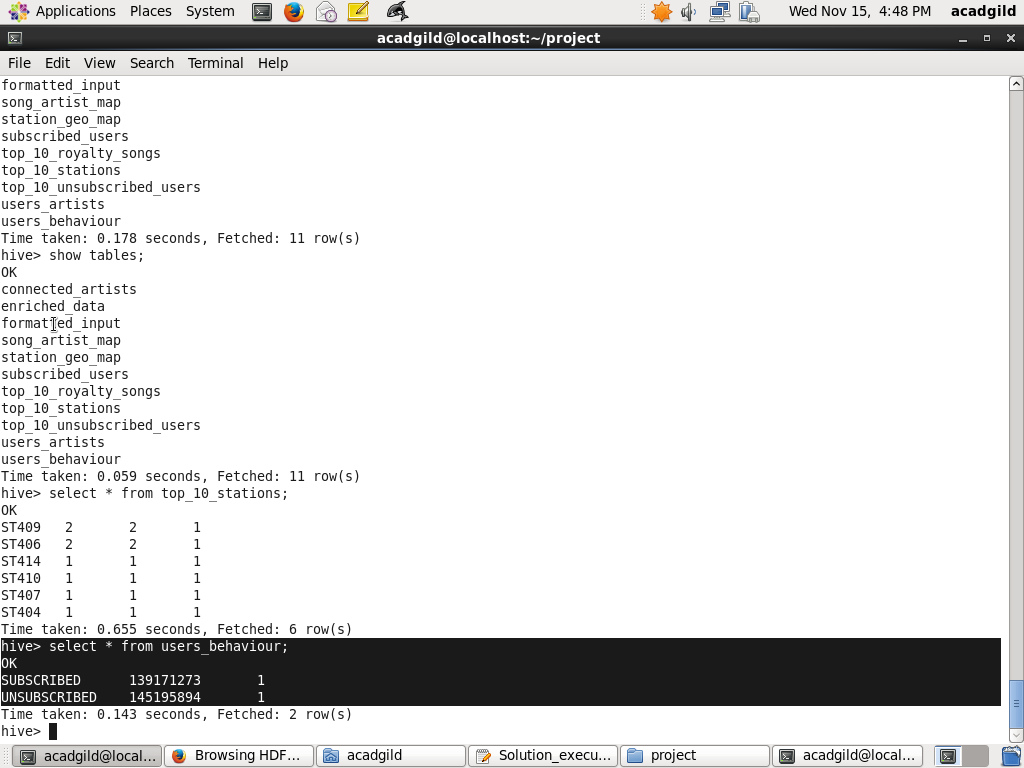
WHERE ed.status='pass'

AND ed.batchid=${hiveconf:batchid}

GROUP BY CASE WHEN (su.user\_id IS NULL OR CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn\_end\_dt AS DECIMAL(20,0))) THEN 'UNSUBSCRIBED'

WHEN (su.user\_id IS NOT NULL AND CAST(ed.timestamp AS DECIMAL(20,0)) <= CAST(su.subscn\_end\_dt AS DECIMAL(20,0))) THEN 'SUBSCRIBED' END;

**Output:**



1. **Determine top 10 connected artists. Connected artists are those whose songs are most listened by the unique users who follow them.**

**Sol:**

CREATE TABLE IF NOT EXISTS connected\_artists

(

artist\_id STRING,

user\_count INT

)

PARTITIONED BY (batchid INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

INSERT OVERWRITE TABLE connected\_artists

PARTITION(batchid=${hiveconf:batchid})

SELECT

ua.artist\_id,

COUNT(DISTINCT ua.user\_id) AS user\_count

FROM

(

SELECT user\_id, artist\_id FROM users\_artists

LATERAL VIEW explode(artists\_array) artists AS artist\_id

) ua

INNER JOIN

(

SELECT artist\_id, song\_id, user\_id

FROM enriched\_data

WHERE status='pass'

AND batchid=${hiveconf:batchid}

) ed

ON ua.artist\_id=ed.artist\_id

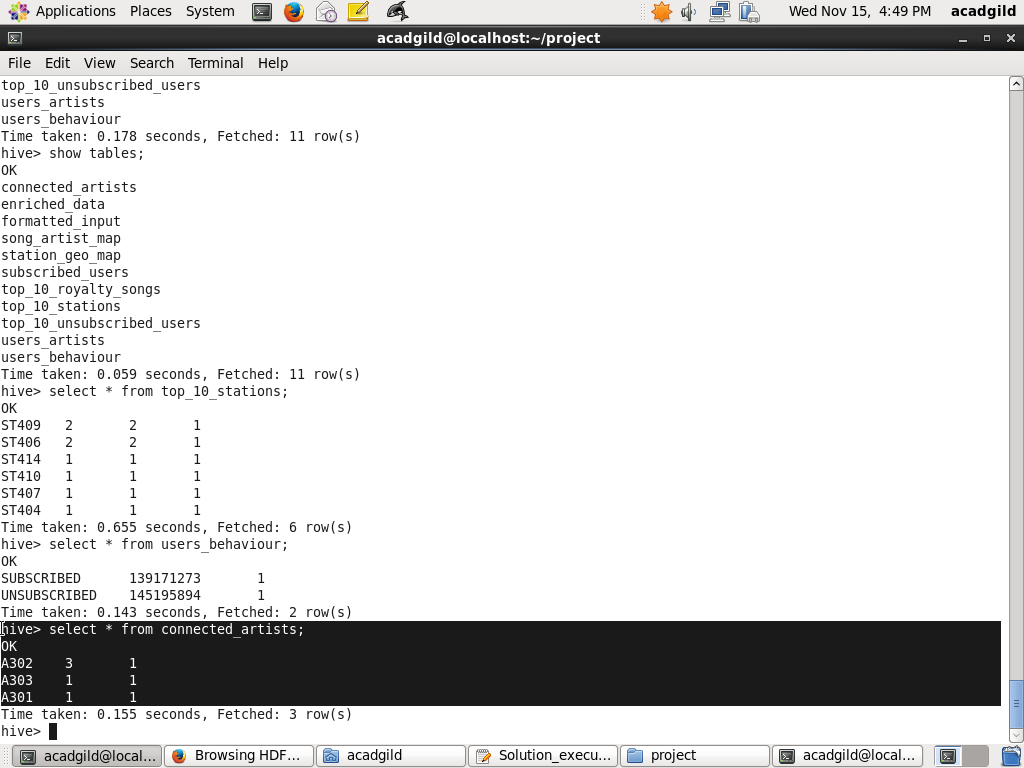
AND ua.user\_id=ed.user\_id

GROUP BY ua.artist\_id

ORDER BY user\_count DESC

LIMIT 10;

**Output:**

****

1. **Determine top 10 songs who have generated the maximum revenue. Royalty applies to a song only if it was liked or was completed successfully or both.**

**Sol:**

CREATE TABLE IF NOT EXISTS top\_10\_royalty\_songs

(

song\_id STRING,

duration INT

)

PARTITIONED BY (batchid INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

INSERT OVERWRITE TABLE top\_10\_royalty\_songs

PARTITION(batchid=${hiveconf:batchid})

SELECT song\_id,

SUM(ABS(CAST(end\_ts AS DECIMAL(20,0))-CAST(start\_ts AS DECIMAL(20,0)))) AS duration

FROM enriched\_data

WHERE status='pass'

AND batchid=${hiveconf:batchid}

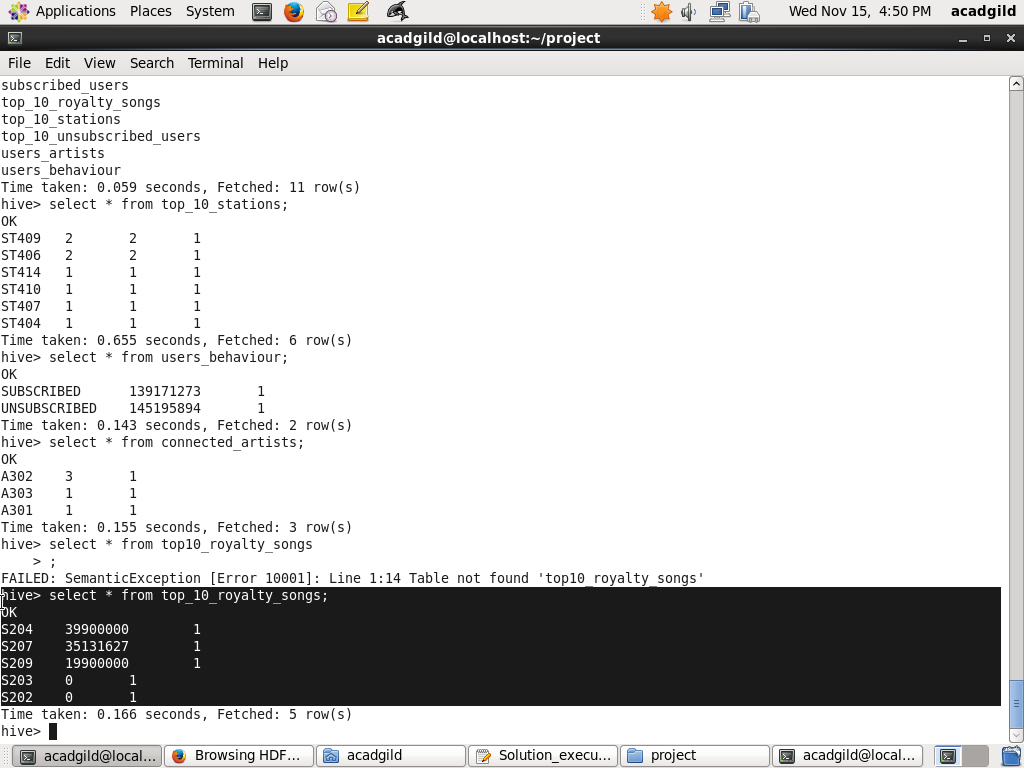
AND (like=1 OR song\_end\_type=0)

GROUP BY song\_id

ORDER BY duration DESC

LIMIT 10;

**Output:**

****

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

**Sol:**

CREATE TABLE IF NOT EXISTS top\_10\_unsubscribed\_users

(

user\_id STRING,

duration INT

)

PARTITIONED BY (batchid INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

INSERT OVERWRITE TABLE top\_10\_unsubscribed\_users

PARTITION(batchid=${hiveconf:batchid})

SELECT

ed.user\_id,

SUM(ABS(CAST(ed.end\_ts AS DECIMAL(20,0))-CAST(ed.start\_ts AS DECIMAL(20,0)))) AS duration

FROM enriched\_data ed

LEFT OUTER JOIN subscribed\_users su

ON ed.user\_id=su.user\_id

WHERE ed.status='pass'

AND ed.batchid=${hiveconf:batchid}

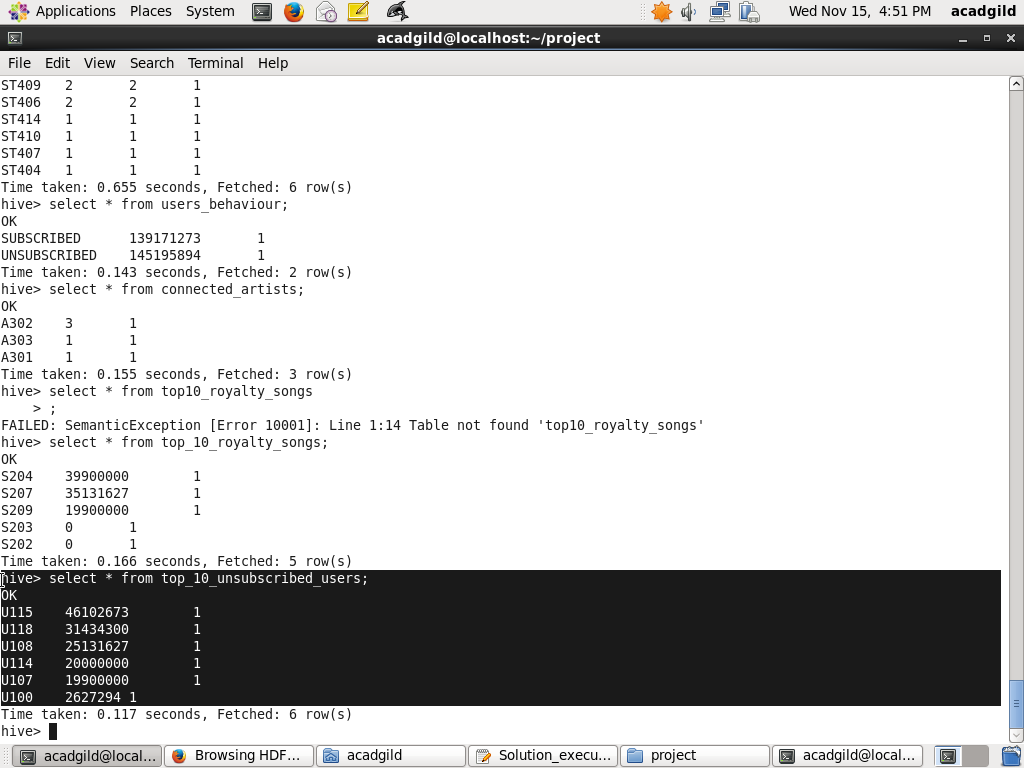
AND (su.user\_id IS NULL OR (CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn\_end\_dt AS DECIMAL(20,0))))

GROUP BY ed.user\_id

ORDER BY duration DESC

LIMIT 10;

**Output:**

****