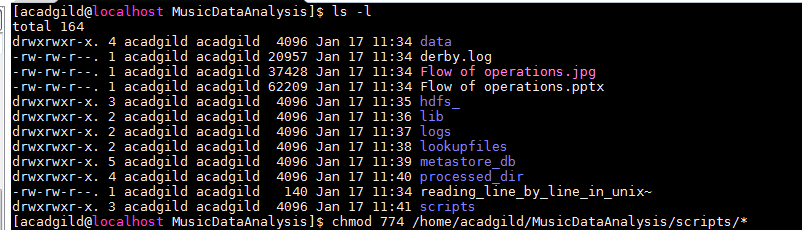
**Music Data Analysis**

The entire directory structure has been moved to /home/acadgild

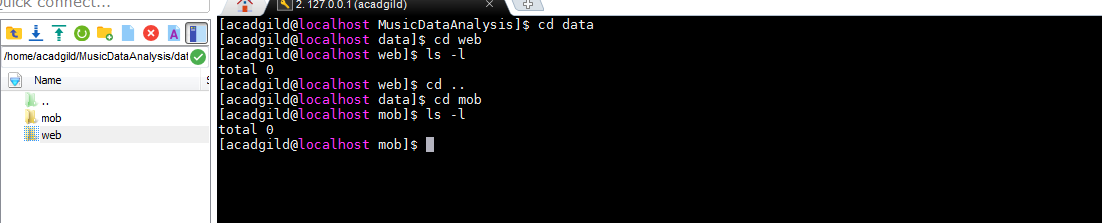
**All the scripts are given 774 permissions:**

chmod 774 /home/acadgild/project/scripts/\*



**Scripts:**

--------

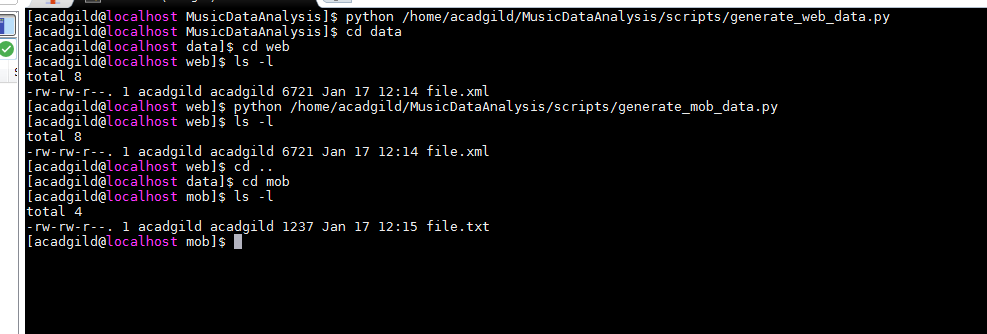


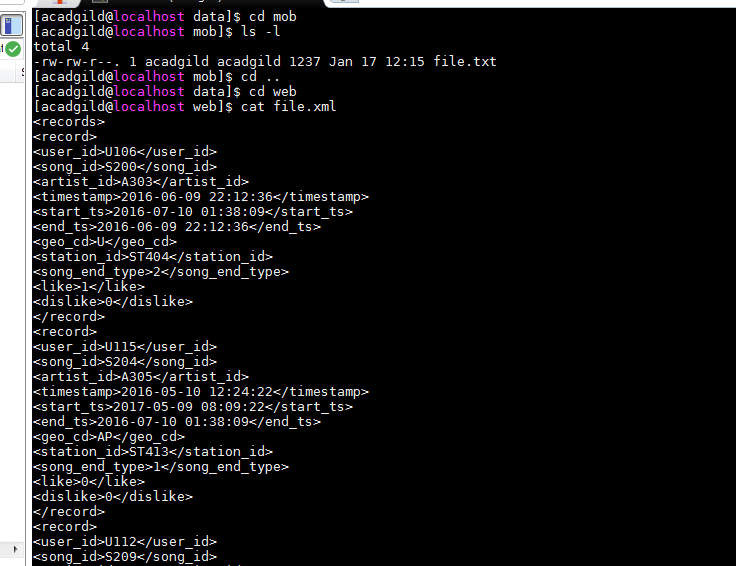
**generate\_web\_data.py -- Generates some random data coming from web application**

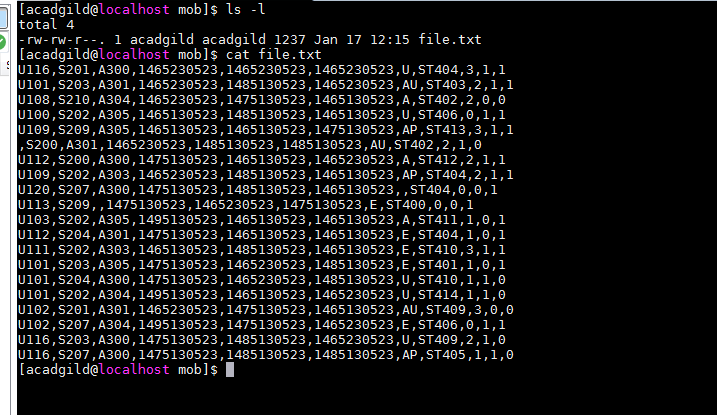
python /home/acadgild/MusicDataAnalysis/scripts/generate\_web\_data.py

**generate\_mob\_data.py -- Generates some random data coming from mobile application**

python /home/acadgild/ MusicDataAnalysis /scripts/generate\_mob\_data.py

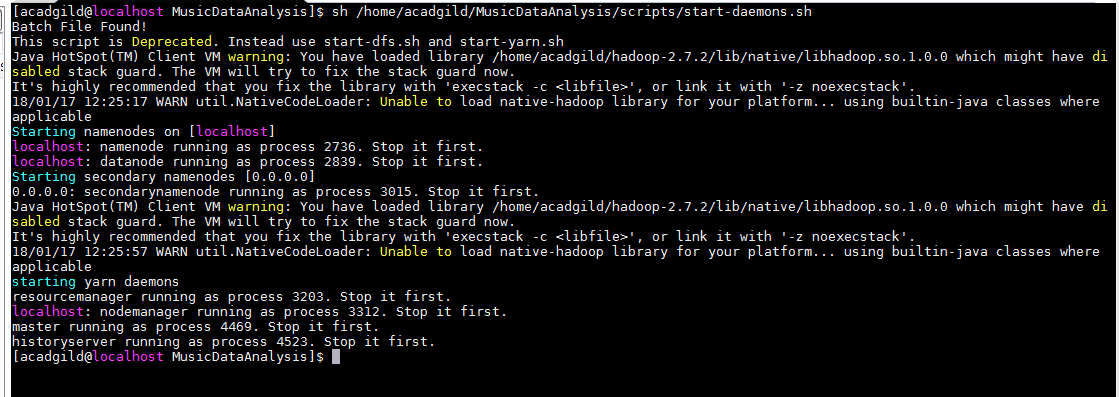






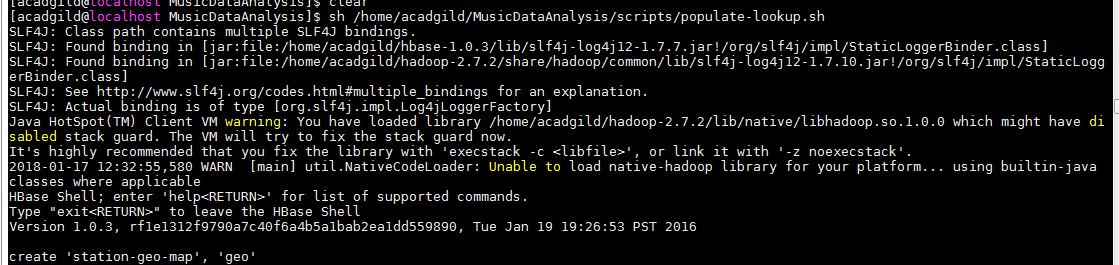
**start-daemons.sh -- Launches all necessary daemons**

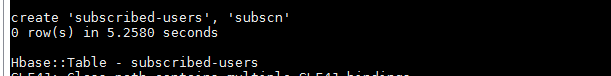
sh /home/acadgild/MusicDataAnalysis/scripts/start-daemons.sh

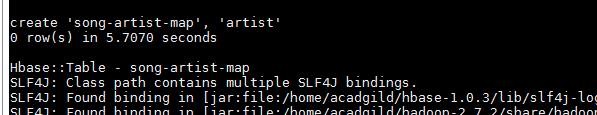


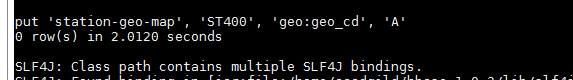
**populate-lookup.sh -- Populates lookup tables**

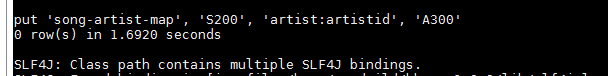
sh /home/acadgild/ MusicDataAnalysis /scripts/populate-lookup.sh

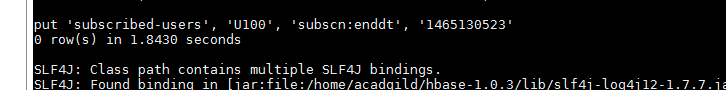








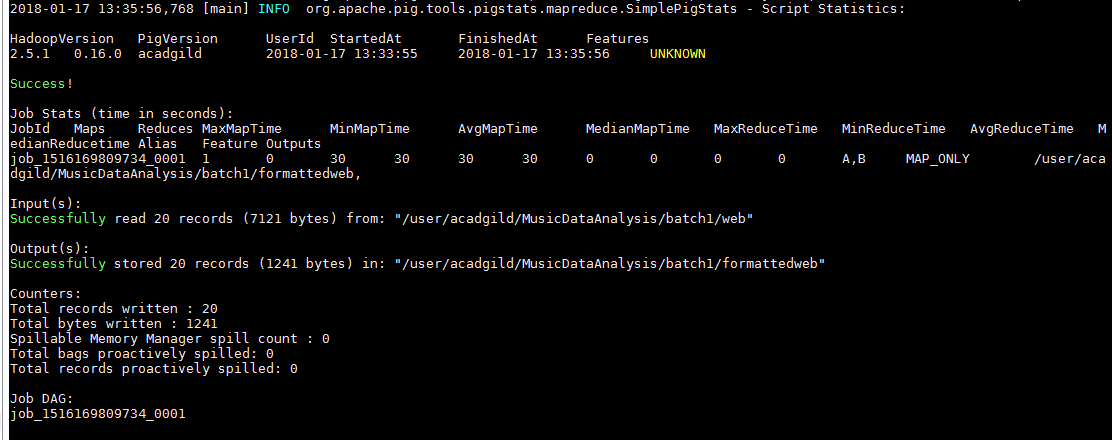




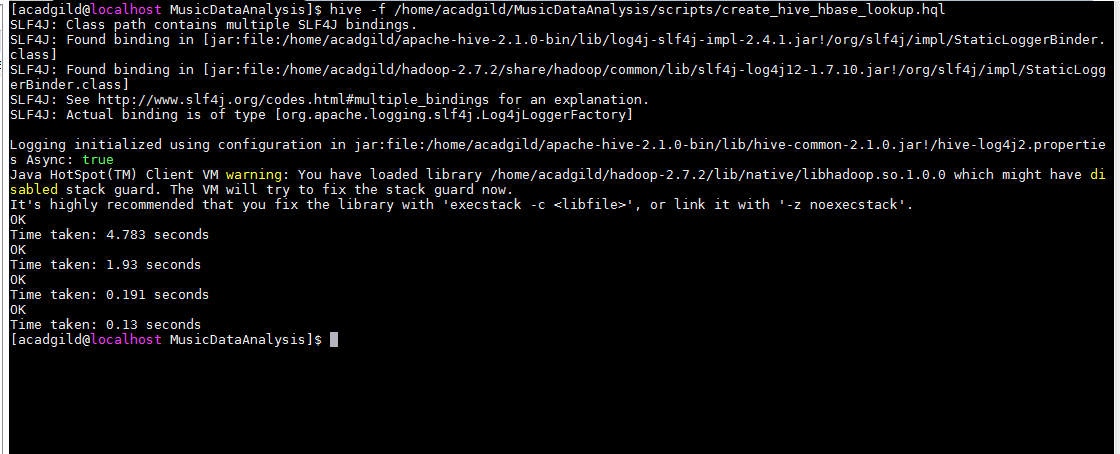
**dataformatting.sh -- Performs data formatting**

sh /home/acadgild/ MusicDataAnalysis /scripts/dataformatting.sh





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



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

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

**data\_analysis.sh -- Performs data analysis**

--sh /home/acadgild/ MusicDataAnalysis /scripts/data\_analysis.sh

**Determine top 10 station\_id(s) where maximum number of songs were played, which were**

**liked by unique users.**

SET hive.auto.convert.join=false;

USE project;

SET hive.support.sql11.reserved.keywords=false;

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=1)

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'

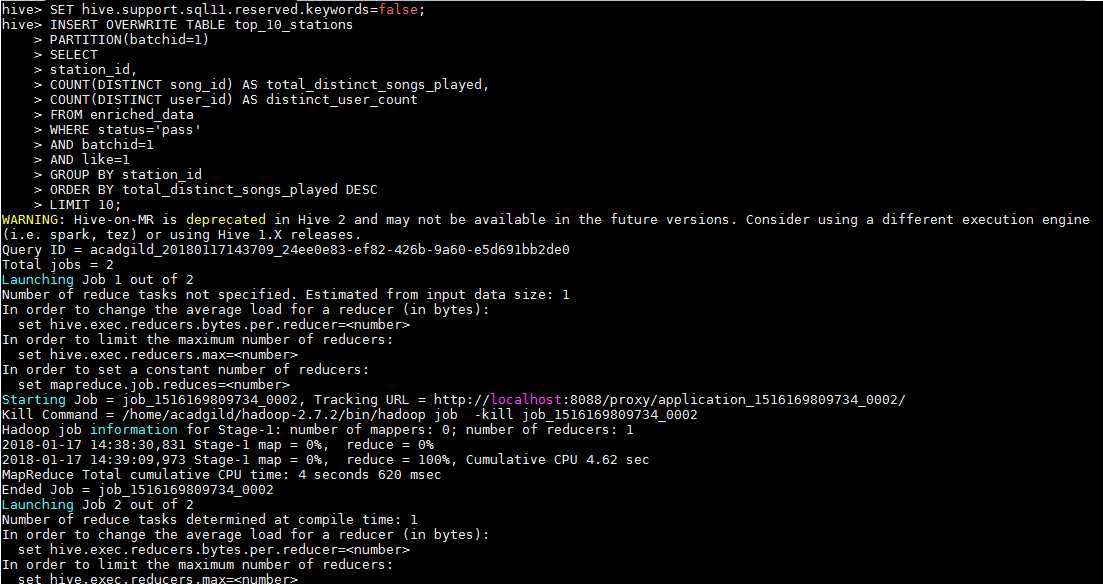
AND batchid=1

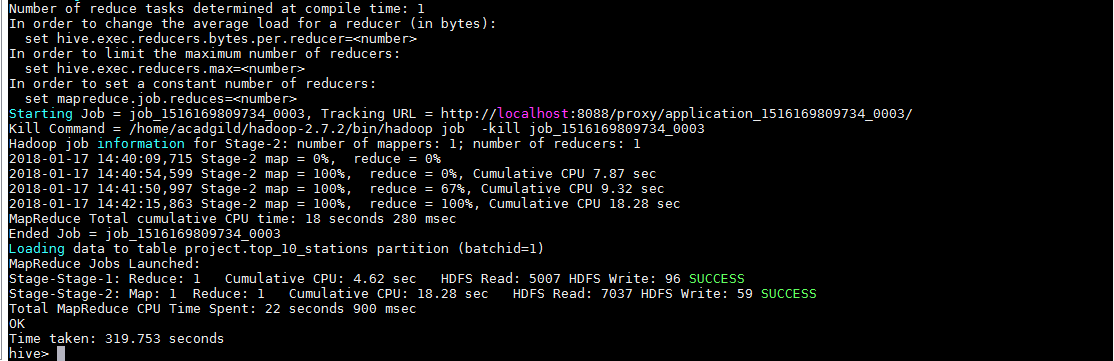
AND like=1

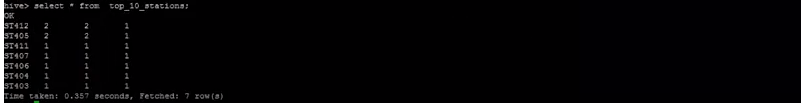
GROUP BY station\_id

ORDER BY total\_distinct\_songs\_played DESC

LIMIT 10;







**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.**

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=1)

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

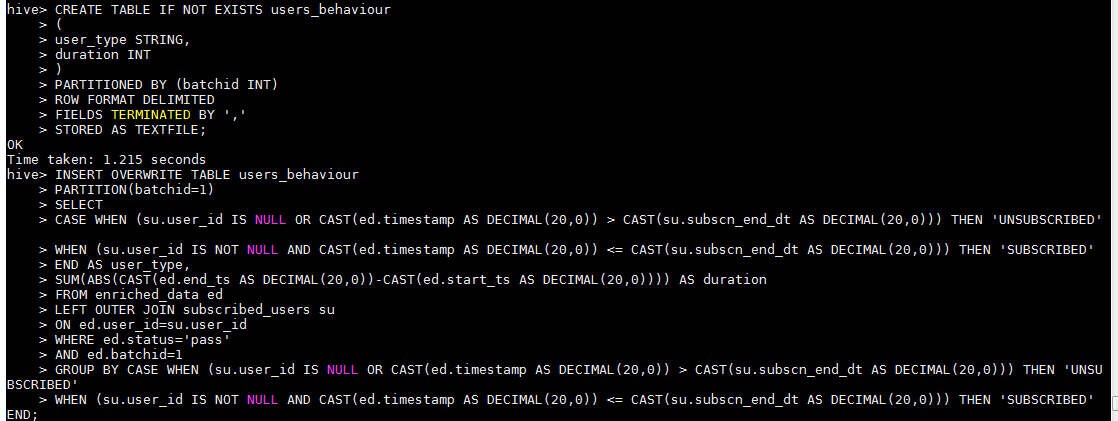
ON ed.user\_id=su.user\_id

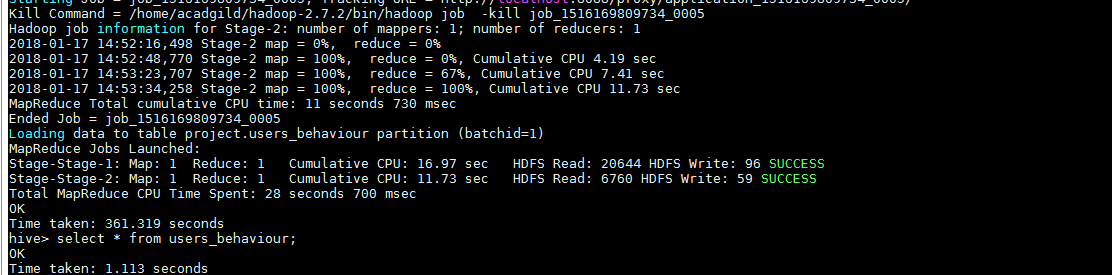
WHERE ed.status='pass'

AND ed.batchid=1

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;





**Determine top 10 connected artists. Connected artists are those whose songs are most**

**listened by the unique users who follow them.**

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=1)

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=1

) ed

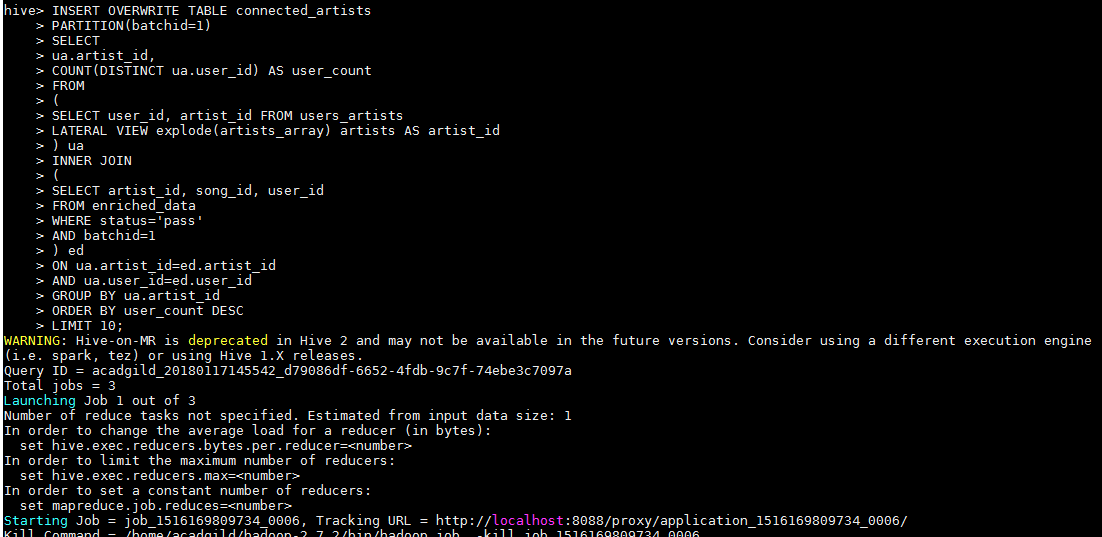
ON ua.artist\_id=ed.artist\_id

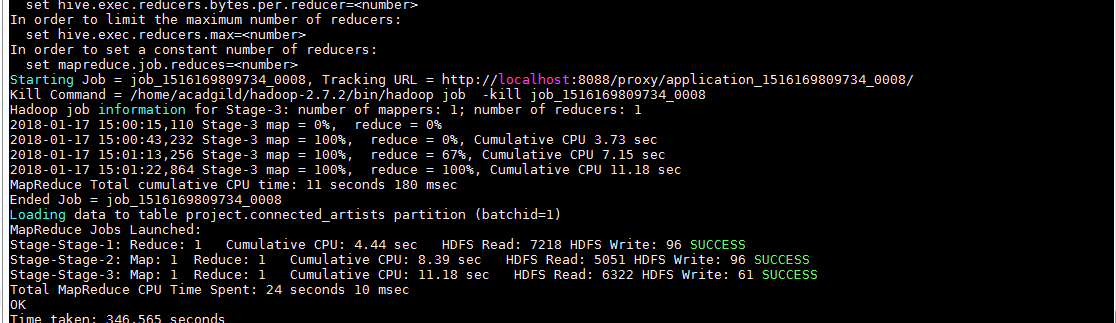
AND ua.user\_id=ed.user\_id

GROUP BY ua.artist\_id

ORDER BY user\_count DESC

LIMIT 10;





**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.**

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=1)

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'

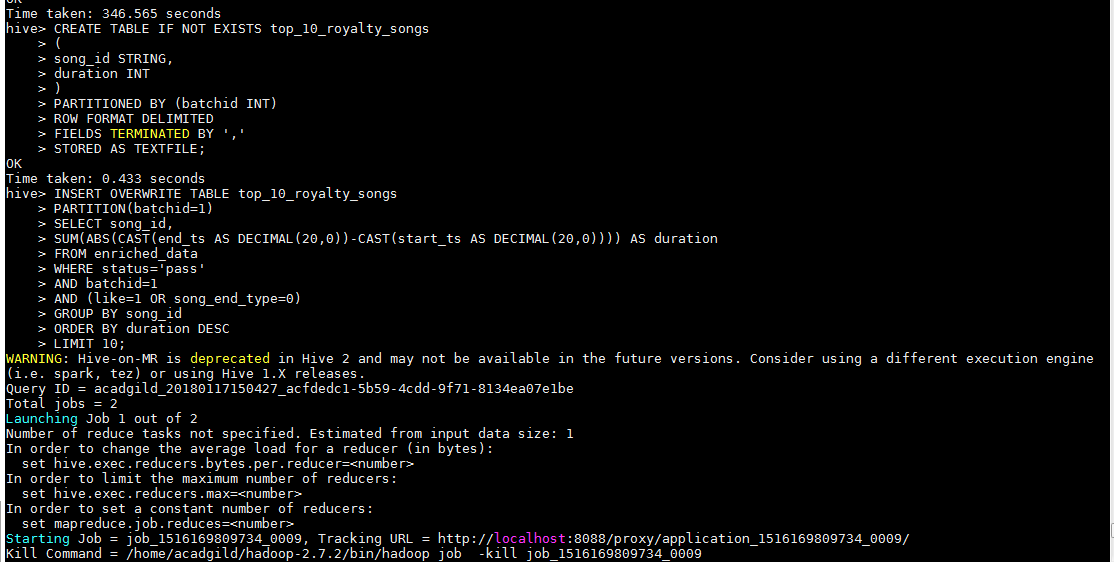
AND batchid=1

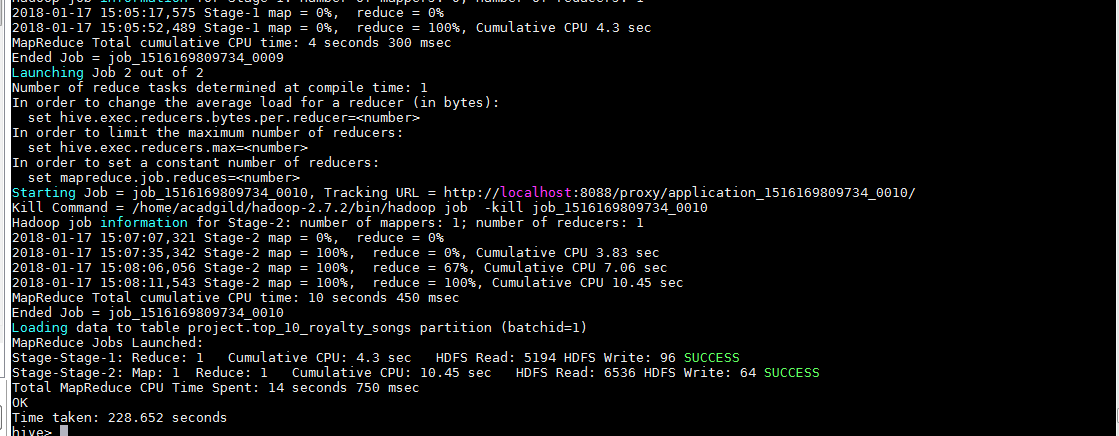
AND (like=1 OR song\_end\_type=0)

GROUP BY song\_id

ORDER BY duration DESC

LIMIT 10;





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

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=1)

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'

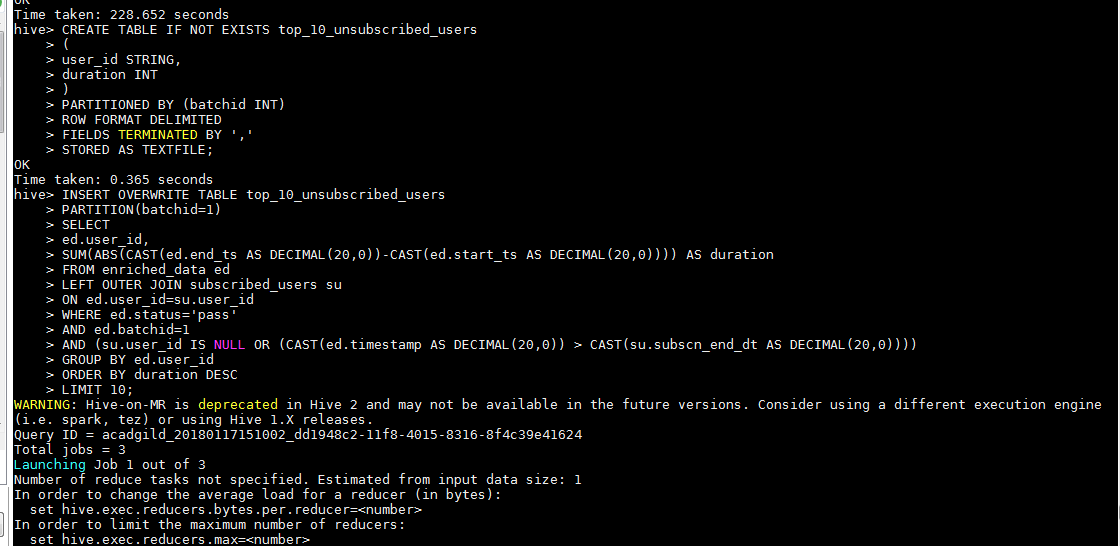
AND ed.batchid=1

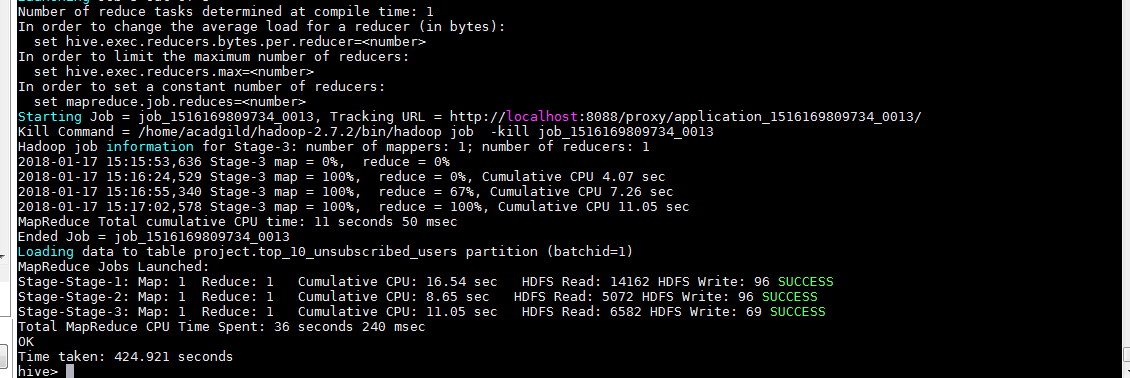
AND (su.user\_id IS NULL OR (CAST(ed.timestamp AS DECIMAL(20,0)) > CAST(su.subscn\_end\_dt AS DECIMAL(20,0))))

GROUP BY ed.user\_id

ORDER BY duration DESC

LIMIT 10;





Scheduling:

-----------

We'll be using crontab for job scheduling.

Job has to run every 3 hours

sudo crontab -e

(Press i to enter insert mode)

\* \*/3 \* \* \* /home/acadgild/project/scripts/wrapper.sh

(Press Esc and then type :wq! and press Enter)