**Banking data analysis**

**Creating database bank in mysql**

CREATE DATABASE bank;

USE bank;



**Creating tables in mysql and inserting the data into mysql tables**

**Creating table loan\_info**

CREATE TABLE loan\_info (

loan\_id int,

user\_id int,

last\_payment\_date DATE,

payment\_installation DOUBLE,

date\_payable DATE

);



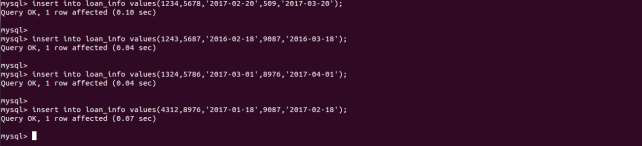
**Inserting data into loan\_info table**

insert into loan\_info values(1234,5678,'2017-02-20',509,'2017-03-20');

insert into loan\_info values(1243,5687,'2016-02-18',9087,'2016-03-18');

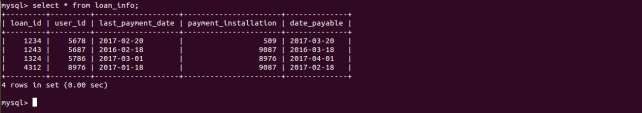
insert into loan\_info values(1324,5786,'2017-03-01',8976,'2017-04-01');

insert into loan\_info values(4312,8976,'2017-01-18',9087,'2017-02-18');



**Checking the data in loan\_info table**

select \* from loan\_info



**Creating table credit\_card\_info**

CREATE TABLE credit\_card\_info

(

cc\_number bigint,

user\_id int,

maximum\_credit DOUBLE,

outstanding\_balance DOUBLE,

due\_date DATE

);



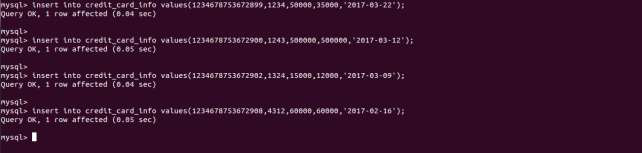
**Inserting data into the credit\_card\_info table**

insert into credit\_card\_info values(1234678753672899,1234,50000,35000,'2017-03-22');

insert into credit\_card\_info values(1234678753672900,1243,500000,500000,'2017-03-12');

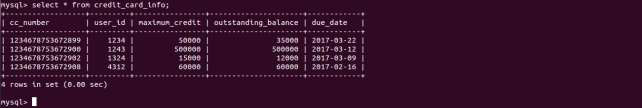
insert into credit\_card\_info values(1234678753672902,1324,15000,12000,'2017-03-09');

insert into credit\_card\_info values(1234678753672908,4312,60000,60000,'2017-02-16');



**Checking the data in credit\_card\_info table**

select \* from credit\_card\_info;



**Creating table shares\_info**

CREATE TABLE shares\_info

(

share\_id varchar(10),

company\_name varchar(20),

gmt\_timestamp bigint,

share\_price DOUBLE

);

**Inserting data into shares\_info table**

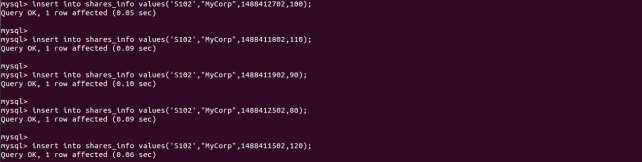
insert into shares\_info values('S102',"MyCorp",1488412702,100);

insert into shares\_info values('S102',"MyCorp",1488411802,110);

insert into shares\_info values('S102',"MyCorp",1488411902,90);

insert into shares\_info values('S102',"MyCorp",1488412502,80);

insert into shares\_info values('S102',"MyCorp",1488411502,120);



**Checking the data in shares\_info table**

select \* from shares\_info;



**Commit;**



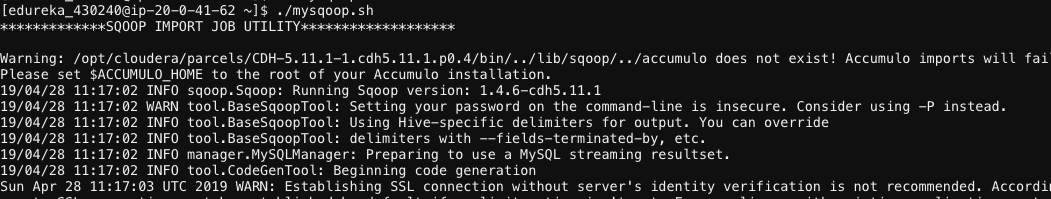
**2.Exporting data from Mysql to HDFS using sqoop**

You can insert data in hdfs first and then move it into hive tables but I am using sqoop to directly insert my data in hive tables.

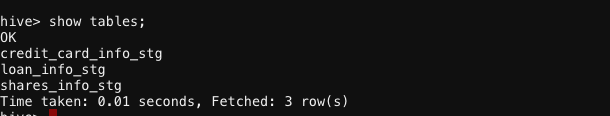
Since there needs to be three tables in hive. I have created shell script for multiple sqoop statements.

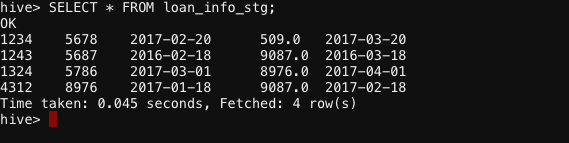
I have created script with name mysqoop.sh, so running script on my terminal.

./mysqoop.sh



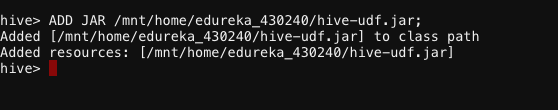
**Checking Output in Hive:**





**3. Creating core tables and loading the data into the core tables from stg tables**

**Adding the udf into hive shell.**

****

CREATE TEMPORARY FUNCTION encrypt AS 'encryption.AESencrypt';

CREATE TEMPORARY FUNCTION decrypt AS 'encryption.AESdecrypt';

**Creating loan\_info table**

CREATE TABLE loan\_info (

Loan\_id string,

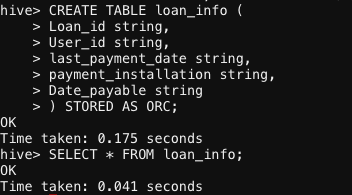
User\_id string,

last\_payment\_date string,

payment\_installation string,

Date\_payable string

) STORED AS ORC;



**Inserting data into loan\_info table (encrypt data)**

INSERT INTO TABLE loan\_info

SELECT encrypt(Loan\_id),

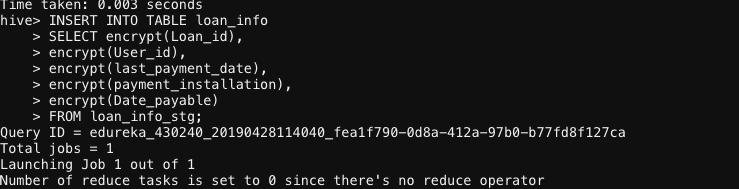
encrypt(User\_id),

encrypt(last\_payment\_date),

encrypt(payment\_installation),

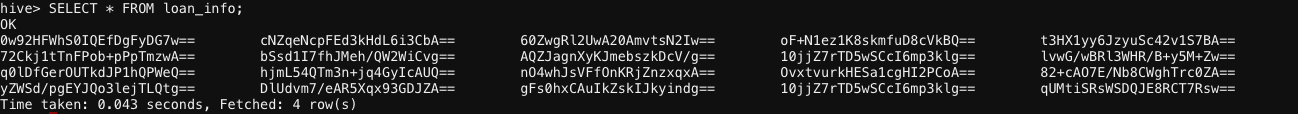
encrypt(Date\_payable)

FROM loan\_info\_stg;



**Checking Data:**

SELECT \* FROM loan\_info;



**Creating shares\_info table**

CREATE TABLE shares\_info

(

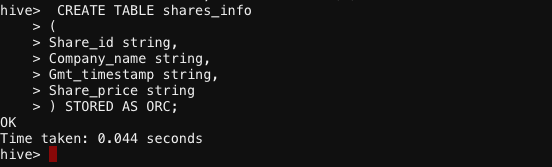
Share\_id string,

Company\_name string,

Gmt\_timestamp string,

Share\_price string

) STORED AS ORC;



**Inserting data into shares\_info table**

INSERT INTO TABLE shares\_info

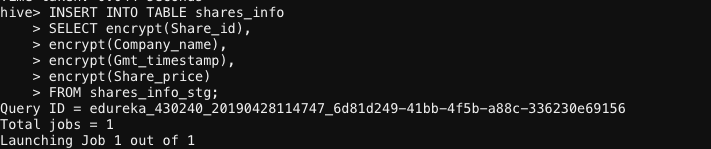
SELECT encrypt(Share\_id),

encrypt(Company\_name),

encrypt(Gmt\_timestamp),

encrypt(Share\_price)

FROM shares\_info\_stg;



**Creating credit\_card\_info table**

CREATE TABLE credit\_card\_info

(

cc\_number string,

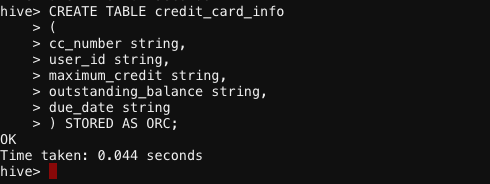
user\_id string,

maximum\_credit string,

outstanding\_balance string,

due\_date string

) STORED AS ORC;



**Inserting data into credit\_card\_info table**

INSERT INTO TABLE credit\_card\_info

SELECT encrypt(cc\_number),

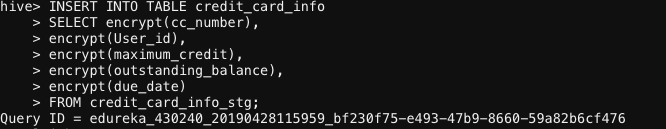
encrypt(User\_id),

encrypt(maximum\_credit),

encrypt(outstanding\_balance),

encrypt(due\_date)

FROM credit\_card\_info\_stg;



**4. Analysis**

**Decrypting the data for analysis:**

CREATE TEMPORARY FUNCTION encrypt AS 'encryption.AESencrypt';

CREATE TEMPORARY FUNCTION decrypt AS 'encryption.AESdecrypt';

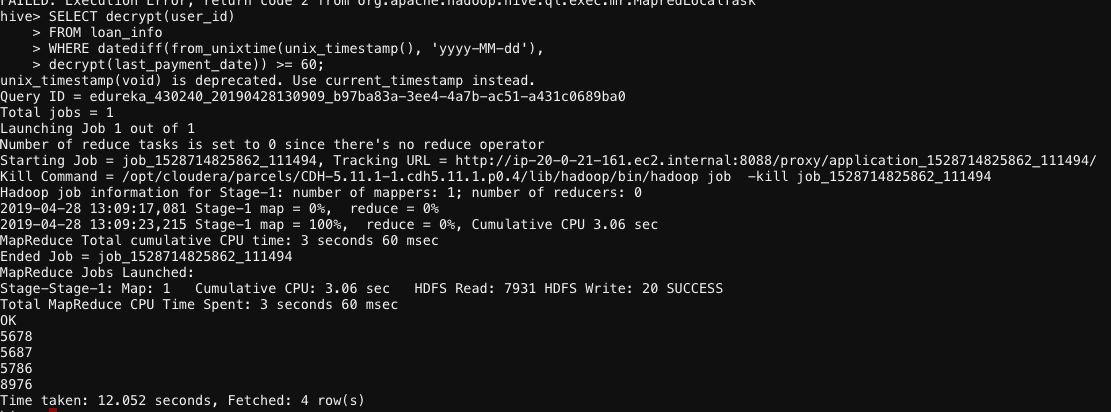
CREATE TEMPORARY FUNCTION max\_profit AS 'maxprofit.MaxProfit';

**6.1. Find out the list of users who have at least 2 loan instalments pending.**

SELECT decrypt(User\_id) FROM

Loan\_info WHERE  
from\_unixtime(unix\_timestamp(), 'yyyy-MM-dd'),

decrypt(last\_payment\_date)) >=60;



**6.2**. **Find the list of users who have a healthy credit card but outstanding loan account. Healthy credit card means no outstanding balance.**

SELECT decrypt(li.user\_id)

FROM loan\_info li INNER JOIN credit\_card\_info cci

ON decrypt(li.user\_id) = decrypt(cci.user\_id)

WHERE CAST(decrypt(cci.outstanding\_balance) AS double) = 0.0

AND datediff(from\_unixtime(unix\_timestamp(), 'yyyy-MM-dd'), decrypt(li.last\_payment\_date)) >=

30;

**6.3. For every share and for every date, find the maximum profit one could have made on the share.**

**Bear in mind that a share purchase must be before share sell and if share prices fall throughout the day, maximum possible profit may be negative.**

SELECT share\_id, share\_date, max\_profit(collect\_list(share\_price))

FROM

(

SELECT decrypt(Share\_id) AS share\_id,

decrypt(Gmt\_timestamp) AS Gmt\_timestamp,

from\_unixtime(CAST(decrypt(Gmt\_timestamp) AS int), 'yyyy-MM-dd') AS share\_date,

CAST (decrypt(Share\_price) AS double) AS share\_price

FROM shares\_info

DISTRIBUTE BY share\_id,

from\_unixtime(CAST(Gmt\_timestamp AS int), 'yyyy-MM-dd')

SORT BY share\_id,

CAST(Gmt\_timestamp AS int)

) inne GROUP BY share\_id, share\_date;



**7.Archival**

8.Survey data analysis

We have 3 survery part files. So we will copy the contents into a single file using the below linux

Commands:

cd /home/acadgild/survey\_files

cat \*.txt > survey\_data



**Creating hive table to load survey\_data**

CREATE TABLE survey\_analysis (

survey\_date string,

survey\_question string,

Rating int,

user\_id int,

survey\_id string

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ',';

**Loading data into survey\_analysis table**

LOAD DATA LOCAL INPATH 'survey\_files/survey\_data' INTO TABLE

Survey\_analysis;

**8.1. How many surveys got the average rating less than 3, provided at least 10 distinct users**

**gave the rating?**

SELECT survey\_id, AVG(rating)

FROM

(SELECT survey\_id,rating, COUNT(user\_id) OVER (PARTITION BY survey\_id ) AS Total\_USERS

FROM survey\_analysis) a

WHERE

a.Total\_USERS>10

GROUP BY survey\_id

HAVING AVG(rating) < 3;

**8.2. Find the details of the survey which received the minimum rating. The condition is that the**

**survey must have been rated by at least 20 users.**

SELECT survey\_id, rank FROM

(

SELECT survey\_id, RANK() OVER (ORDER BY avg\_rating) AS rank

FROM

(

SELECT survey\_id, AVG(rating) AS avg\_rating FROM

(

SELECT survey\_id, rating, COUNT(user\_id) OVER (PARTITION BY survey\_id) AS num\_users

FROM bank.survey\_analysis

) inner\_1

WHERE num\_users >= 20

GROUP BY survey\_id

) inner\_2

) inner\_3

WHERE rank = 1;



**Email data analysis**

The organisation also has lots of emails stored in small files.

The metadata about the email is present in an XML file email\_metadata.xml

Read the XML file for email structure and pack all the email files in HDFS.

For reading XML data, we are using below python script to read the data from xml file.

xml.etree.ElementTree as ET

import commands

base\_str = file("/home/acadgild/email\_schema.xml", "r").read().replace("\t","").replace(" ","")

root = ET.fromstring(base\_str)

structure\_list = []

for each\_col in root.findall("column"):

name = each\_col.find("name").text

type = each\_col.find("type").text

structure\_list.append(name + " " + type)

create\_table = "CREATE TABLE email\_analysis (" + ",".join(structure\_list) + ") ROW FORMAT

DELIMITED FIELDS TERMINATED BY ',';"

hive\_file = file("/home/acadgild/hive\_query.hql", "w")

hive\_file.write("CREATE DATABASE IF NOT EXISTS bank;\n")

hive\_file.write("USE bank;\n")

hive\_file.write(create\_table)

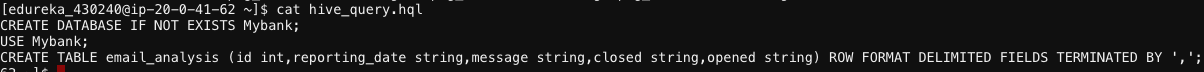
hive\_file.close()

status, output = commands.getstatusoutput("hive -f " + hive\_file.name)

Save this fine and run this script from command line.



You will see a file with name hive\_query.hql would have been created in the terminal.



And, in hive, tables will be created under given Database.

**1. Which is the longest running email?**

SELECT id FROM

(

SELECT id, RANK() OVER (ORDER BY datediff(closed\_date, opened\_date) DESC) AS rank

FROM

(

SELECT id,

MIN(IF(opened="YES",reporting\_date,NULL)) AS opened\_date,

MIN(IF(closed="YES",reporting\_date,NULL)) AS closed\_date

FROM email\_analysis

GROUP BY id

) inner\_1

WHERE opened\_date IS NOT NULL AND closed\_date IS NOT NULL

) inner\_2

WHERE rank = 1;

**2. Find out the list of emails which were unanswered.**

SELECT id

FROM

(

SELECT id,

MIN(IF(opened="YES",reporting\_date,NULL)) AS opened\_date,

MIN(IF(closed="YES",reporting\_date,NULL)) AS closed\_date

FROM email\_analysis

GROUP BY id

) inne

WHERE opened\_date IS NULL AND closed\_date IS NOT NULL;