**Project Implementation tasks**

**Task 1:**

=====================================================================================

Move file from windows Desktop/Shared to Cloudera Desktop/Shared using Shared folder

**Step1: -**

Create directory project\_input\_data

*hadoop fs -mkdir project\_input\_data*

**Step2:-**

Remove header from card\_transactions.csv file before transferring to HDFS

*sed -i '1d'  Desktop/Shared/card\_transactions.csv*

**Step3: -**

Moving file from cloudera local to newly created HDFS folder project\_input\_data

*hadoop fs -put Desktop/card\_transactions.csv project\_input\_data*

**Step4: -**

Verifying the record count of the card\_transactions file loaded into HDFS

hadoop fs -cat project\_input\_data/card\_transactions.csv | wc -l



**Task 2:**

=====================================================================================

**Step1: -**

Login to mysql

*mysql -u root -p*

Creating database bigdataproject and using same

*create database bigdataproject;*

*use bigdataproject;*



**Step2: -**

Creating table card\_transactions in MySQL based on the card\_transactions.csv file structure.

*create table stg\_card\_transactions (*

*card\_id bigint,*

*member\_id bigint,*

*amount int,*

*postcode int,*

*pos\_id bigint,*

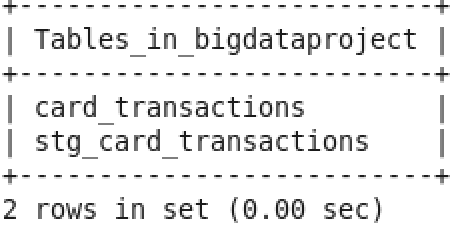
*transaction\_dt varchar(255),*

*status varchar(50),*

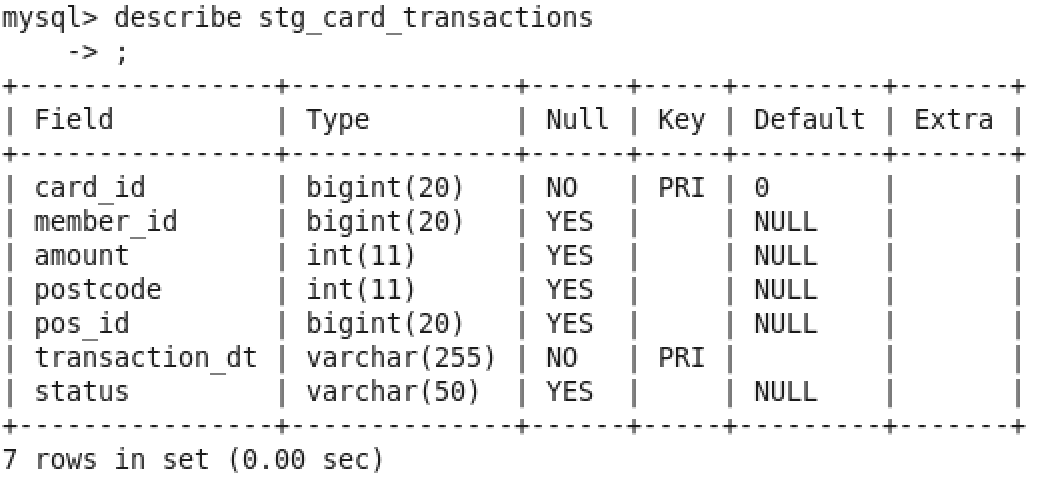
*PRIMARY KEY (card\_id ,transaction\_dt)*

*);*

*show tables;*



*Describe stg\_card\_transactions;*

**

*create table card\_transactions (*

*card\_id bigint,*

*member\_id bigint,*

*amount int,*

*postcode int,*

*pos\_id bigint,*

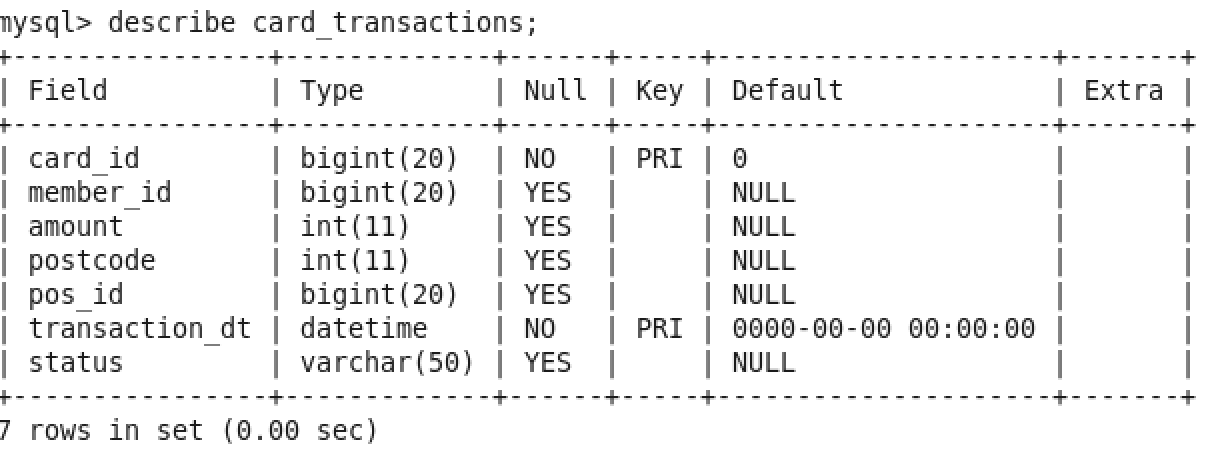
*transaction\_dt datetime,*

*status varchar(50),*

*PRIMARY KEY (card\_id ,transaction\_dt)*

*);*

*Describe card\_transactions;*

**

**Task 3:**

=====================================================================================

**Step1: -**

Doing sqoop export of card\_transactions data using password encryption from HDFS to MYSQL(staging table)

*hadoop credential create mysql.bigdataproject.password -provider jceks://hdfs/user/cloudera/mysql.dbpassword.jceks*

*sqoop export \*

*-Dhadoop.security.credential.provider.path=jceks://hdfs/user/cloudera/mysql.dbpassword.jceks \*

*--connect jdbc:mysql://quickstart.cloudera:3306/bigdataproject \*

*--username root \*

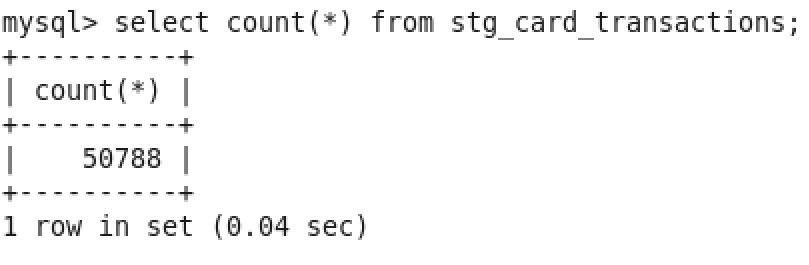
*--password-alias mysql.bigdataproject.password \*

*--table stg\_card\_transactions \*

*--export-dir project\_input\_data/card\_transactions.csv \*

*--fields-terminated-by ','*

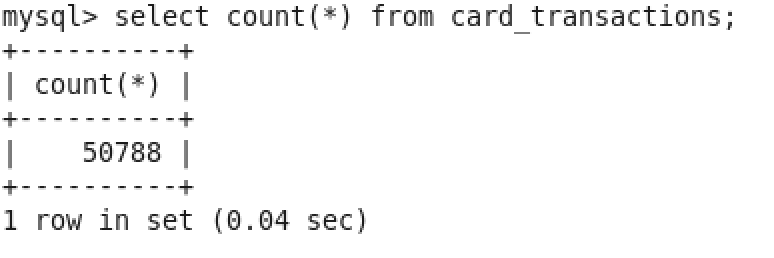
*select count(\*) from bigdataproject.stg\_card\_transactions;*



**step 2:**

**Transferring data from** *stg\_card\_transactions to card\_transactions*

**insert into card\_transactions select card\_id,member\_id,amount,postcode,pos\_id, STR\_TO\_DATE(transaction\_dt,'%d-%m-%Y %H:%i:%s') as transaction\_dt,status from stg\_card\_transactions**

****

**Step2: - (Need to change)**

Attached production ready parameterized Sqoop export Shell Script(password encrypted) from HDFS to MYSQL card\_transactions table



**Step3: -**

Providing permission and execution command for sqoop import shell script

*chmod +x sqoop\_export\_card\_transactions.sh*

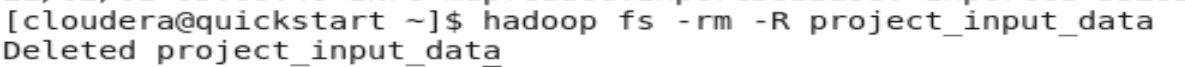
Shell execution command:-

*./sqoop\_export\_card\_transactions.sh quickstart.cloudera:3306 bigdataproject root card\_transactions*

**Step4: -**

Deleting the file from HDFS

hadoop fs -rm /project\_input\_data/card\_transactions.csv



**Task 4:**

=====================================================================================

**Step1: -**

Enter to Hive Beeline Terminal

*beeline -u jdbc:hive2://*

Create database in Hive

*create database bigdataproject;*

*use database bigdataproject;*

Create hive external table- member\_score

*create external table if not exists member\_score*

*(*

*member\_id string,*

*score float*

*)*

*row format delimited fields terminated by ','*

*stored as textfile*

*location '/project\_input\_data/member\_score/';*

**Step2: -**

Create hive external table- member\_details

*create external table if not exists member\_details*

*(*

*card\_id bigint,*

*member\_id bigint,*

*member\_joining\_dt timestamp ,*

*card\_purchase\_dt timestamp ,*

*country string,*

*city string,*

*score float*

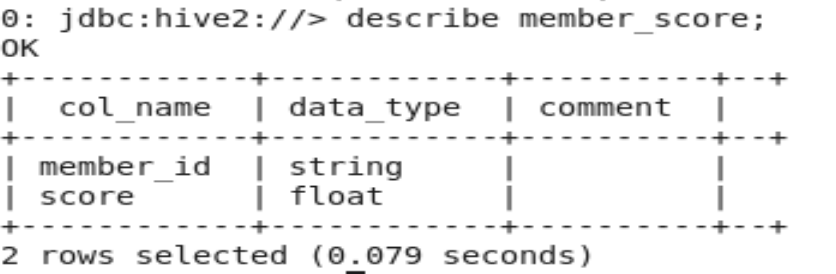
*)*

*row format delimited fields terminated by ','*

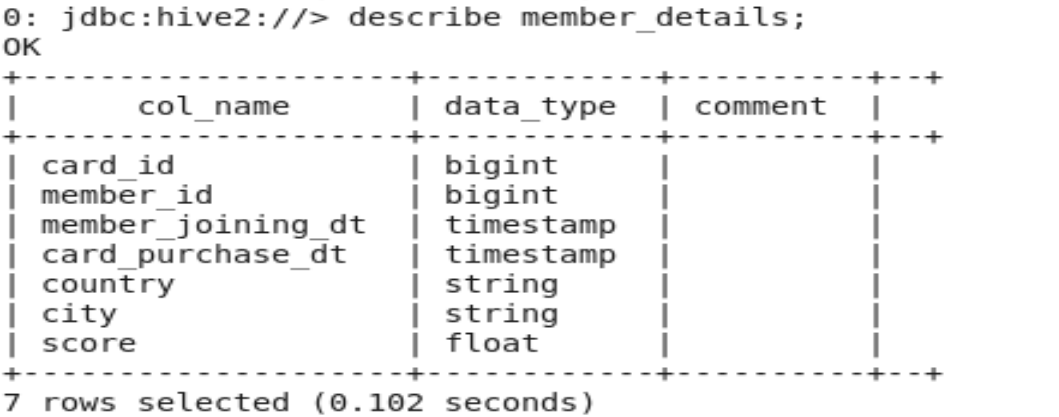
*stored as textfile*

*location '/project\_input\_data/member\_details/';*

Describe member\_score;



Describe member\_details;



**Step3: -**

Enabling bucketing:-

SET HIVE.ENFORCE.BUCKETING=TRUE;

Create hive bucketed table- member\_score with 4 Buckets on column:- member\_id since query/join with this table is based on card\_id

*create table if not exists member\_score\_bucketed*

*(*

*member\_id string,*

*score float*

*)*

*CLUSTERED BY (member\_id) into 4 buckets;*

**Step4: -**

Create hive bucketed table- member\_details with 4 Buckets on column:- card\_id since query/join with this table is based on card\_id

*create table if not exists member\_details\_bucketed*

*(*

*card\_id bigint,*

*member\_id bigint,*

*member\_joining\_dt timestamp ,*

*card\_purchase\_dt timestamp ,*

*country string,*

*city string,*

*score float*

*)*

*CLUSTERED BY (card\_id) into 4 buckets;*

**Step5: -**

Loading bucketed tables member\_score\_bucketed and member\_details\_bucketed from external tables

*insert into table member\_score\_bucketed*

*select \* from member\_score;*

*insert into table member\_details\_bucketed*

*select \* from member\_details;*

**Task 5:**

=====================================================================================

**Step1:-** Creating Hive-HBase table table card\_transactions with 8 buckets on column:-card\_id since query/join with this table is based on card\_id.

*create table card\_transactions*

*(*

*card\_id bigint,*

*member\_id bigint,*

*amount float,*

*postcode int,*

*pos\_id bigint,*

*transaction\_dt**timestamp*,

*status string*

*)*

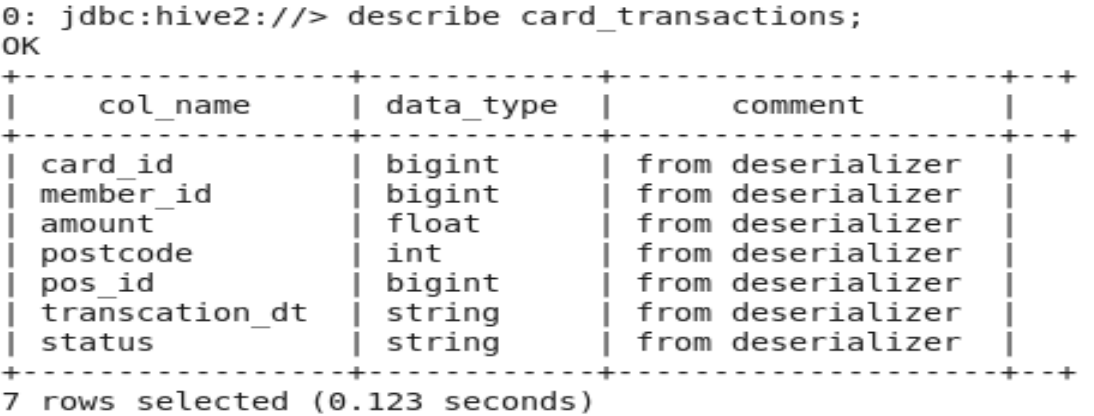
*CLUSTERED by (card\_id) into 8 buckets*

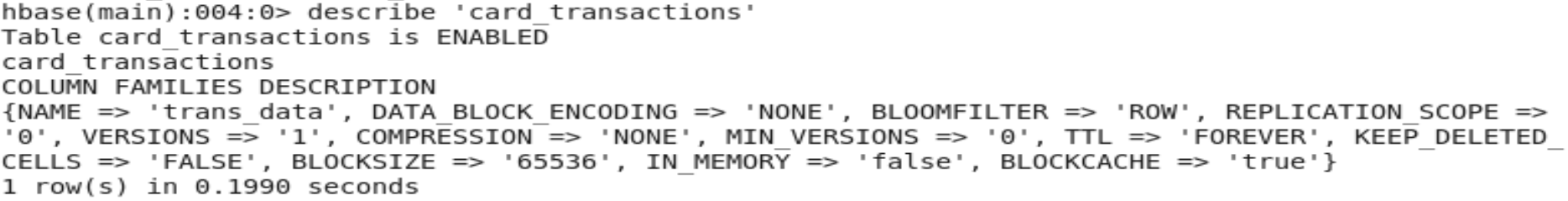
*STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'*

*WITH SERDEPROPERTIES("hbase.columns.mapping"=":key,trans\_data:member\_id,trans\_data:amount, trans\_data:postcode,trans\_data:pos\_id,trans\_data:transaction\_dt,trans\_data:Status")*

*TBLPROPERTIES ("hbase.table.name" = "card\_transactions");*

Describing card\_transactions table from hive



Describing card\_transactions table from HBase

**Task 6:**

=====================================================================================

**Step1:-** Creating Hive-HBase table table card\_lookup with 8 buckets on column:-card\_id since query/join with this table is based on card\_id.

*create table card\_lookup*

*(*

*card\_id bigint ,*

*member\_id bigint ,*

*ucl float ,*

*score float,*

*last\_txn\_time* timestamp,

*last\_txn\_zip string*

*)*

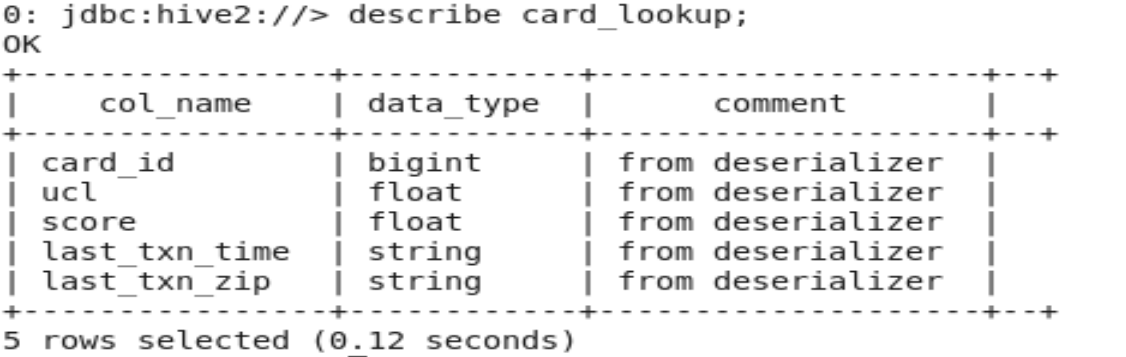
*CLUSTERED by (card\_id) into 8 buckets*

*STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'*

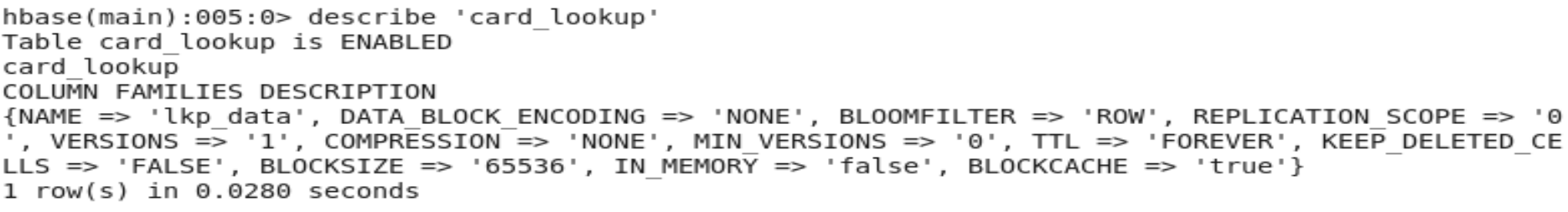
*WITH SERDEPROPERTIES("hbase.columns.mapping"=":key,lkp\_data:ucl,lkp\_data:score, lkp\_data:last\_txn\_time,lkp\_data:last\_txn\_zip")*

*TBLPROPERTIES ("hbase.table.name" = "card\_lookup");*

Describing card\_ lookup table from hive



Describing card\_transactions table from HBase



**Task 7:**

=====================================================================================

**Step1: -**

Doing sqoop import of member\_score data using password encryption from Amazon RDS to HDFS and used delete target directory since it full load.

*hadoop credential create amazonrds.bigdataproject.password -provider jceks://hdfs/user/cloudera/amazonrds.dbpassword.jceks*

*sqoop import \*

*-Dhadoop.security.credential.provider.path=jceks://hdfs/user/cloudera/amazonrds.dbpassword.jceks \*

*--connect jdbc:mysql://database-2.cl4c0rtglkdz.ap-south-1.rds.amazonaws.com/BankingPrj \*

*--username admin \*

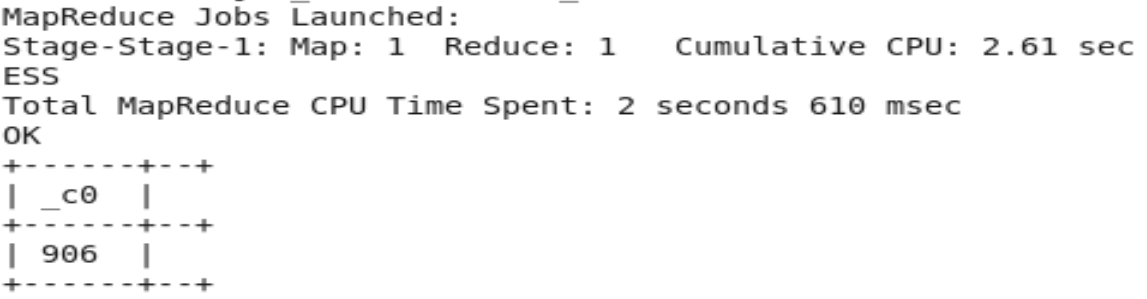
*--password-alias amazonrds.bigdataproject.password \*

*--table member\_score \*

*-- target -dir /project\_input\_data/member\_score \*

*--delete-target-dir*

select count(\*) from bigdataproject.member\_score;



**Step2: -**

Production ready parameterized Sqoop import Shell Script(password encrypted) for Amazon RDS member\_score table



**Step3: -**

Providing permission and execution command for sqoop import shell script

*chmod +x sqoop\_import\_member\_score.sh*

Shell execution command:-

*./sqoop\_import\_member\_score.sh database-2.cl4c0rtglkdz.ap-south-1.rds.amazonaws.com BankingPrj admin member\_score*

**Task 8:**

=====================================================================================

**Step1: -**

Doing sqoop import of member\_details data using password encryption from Amazon RDS to HDFS using incremental append mode based on column:- member\_id

We can reuse the password encryption file created for member\_score table for member\_details too

*sqoop import \*

*-Dhadoop.security.credential.provider.path=jceks://hdfs/user/cloudera/amazonrds.dbpassword.jceks \*

*--connect jdbc:mysql://database-2.cl4c0rtglkdz.ap-south-1.rds.amazonaws.com/BankingPrj \*

*--username admin \*

*--password-alias amazonrds.bigdataproject.password \*

*--table member\_details \*

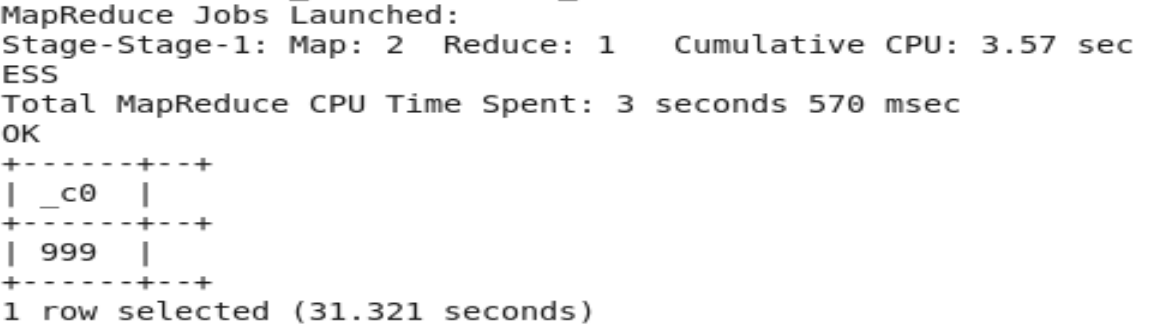
*--warehouse-dir /project\_input\_data \*

*--incremental append \*

*--check-column member\_id \*

*--last-value 0*

select count(\*) from bigdataproject.member\_details;



**Step2: -**

Production ready parameterized Sqoop import Shell Script(password encrypted) for Amazon RDS member\_details table



**Step3: -**

Providing permission and execution command for sqoop import shell script

*chmod +x sqoop\_import\_member\_details.sh*

Shell execution command:-

*./sqoop\_import\_member\_score.sh database-2.cl4c0rtglkdz.ap-south-1.rds.amazonaws.com BankingPrj admin member\_details*

**Task 9:**

=====================================================================================

**Step1: -**

Doing sqoop import of card\_transactions data using password encryption from MYSQL to HDFS

*sqoop import \*

*-Dhadoop.security.credential.provider.path=jceks://hdfs/user/cloudera/mysql.dbpassword.jceks \*

*--connect jdbc:mysql://quickstart.cloudera:3306/bigdataproject \*

*--username root \*

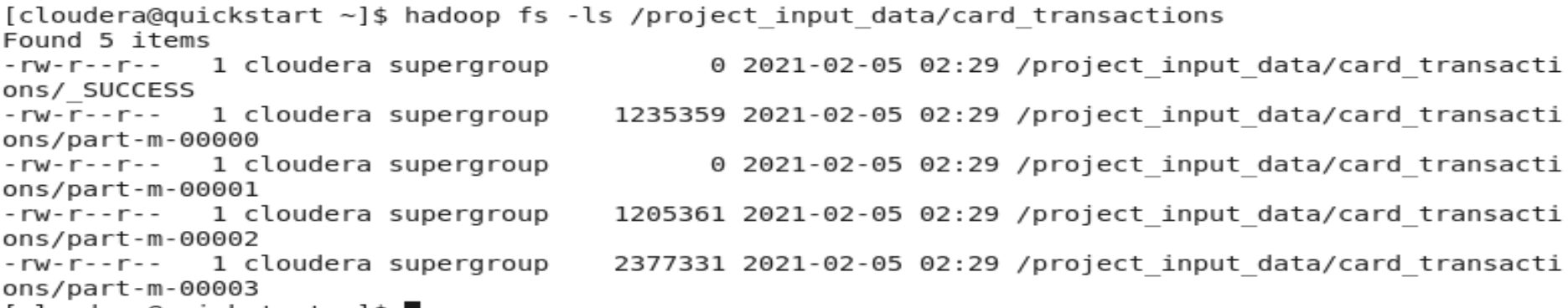
*--password-alias mysql.bigdataproject.password \*

--table card\_transactions \

*--split-by card\_id \*

*--warehouse-dir /project\_input\_data \*

*hadoop fs -ls /project\_input\_data/card\_transactions*



-------------------------------------------------------------------------------------

**Step2: - (Need to change)**

Production ready parameterized Sqoop import Shell Script(password encrypted) for MYSQL card\_transactions table



**Step3: -**

Providing permission and execution command for sqoop import shell script

*chmod +x sqoop\_import\_card\_txns.sh*

Shell execution command:-

*./sqoop\_import\_card\_txns.sh quickstart.cloudera:3306 bigdataproject root card\_transactions*

**Task 13 & 14 Combined:**

=====================================================================================

**Step1: -**

Access the hive tables using apache spark and calculate the UCL.

*import org.apache.spark.sql.{Row, SaveMode, SparkSession}*

*import org.apache.log4j.Logger*

*import org.apache.log4j.Level*

*import org.apache.spark.SparkConf*

*object HiveTest extends App{*

*//set logging level to error*

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

*// create spark config object*

*val sparkConf= new SparkConf()*

*sparkConf.setAppName("Credit\_Card\_Fraud\_Detection")*

*sparkConf.setMaster("local[2]")*

*sparkConf.set("hive.metastore.uris", "thrift://localhost:9083")*

*// use spark config object to create spark session*

*val spark = SparkSession*

*.builder()*

*.config(sparkConf)*

*.enableHiveSupport() //spark integration with hive*

*.getOrCreate()*

*import spark.implicits.\_*

*import spark.sql*

*// start writing the hive queries*

*val df\_ucl=sql("""*

*with cte\_rownum as*

*(*

*select card\_id,amount,member\_id,*

*first\_value(postcode) over(partition by card\_id order by transcation\_dt desc) as postcode,*

*row\_number() over(partition by card\_id order by transcation\_dt desc) rownum*

*from card\_transactions*

*)*

*select card\_id,member\_id,*

*avg(amount)+ 3\* max(std) as UCL ,*

*max(score) score,*

*max(transaction\_dt) as last\_txn\_time,*

*max(Postcode)as last\_txn\_zip*

*from*

*(*

*select*

*card\_id,amount,*

*c.member\_id,*

*m.score,*

*transaction\_dt,*

*Postcode,*

*STDDEV (amount) over(partition by card\_id order by (select 1) desc) std*

*from cte\_rownum c*

*inner join member\_score m on c.member\_id=m.member\_id*

*where rownum<=10*

*)a*

*group by card\_id,member\_id*

*""" )*

*val df=df\_ucl.select("card\_id","member\_id","UCL","score","last\_txn\_time","last\_txn\_zip")*

*df.show*

*}*

**Screenshot for the same:- (need a new screenshot)**

