**CREDIT CARD MANAGEMENT SYSTEM**

Mahshooq K

2061693

**CREATING THE FACT AND DIMENSION TABLE**

1. **Move the data files to cloudera folder using WinSCP**
2. **Creating tables for storing data using hive**

**//create database for the project**

CREATE DATABASE PROJECT;

USE PROJECT;

**//Create command for customer table**

CREATE EXTERNAL TABLE CDW\_SAPP\_CUSTOMER\_2061693 (

CUST\_ID BIGINT,CUST\_F\_NAME STRING, CUST\_M\_NAME STRING,CUST\_L\_NAME STRING,CUST\_CC\_NO BIGINT,CARD\_TYPE STRING,CARD\_LIMIT INT,CARD\_END\_DATE STRING,CUST\_SSN BIGINT,CUST\_STREET STRING,CUST\_CITY STRING,CUST\_STATE STRING,CUST\_COUNTRY STRING,CUST\_ZIP INT,CUST\_PHONE STRING,CUST\_EMAIL STRING)

ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

STORED AS TEXTFILE

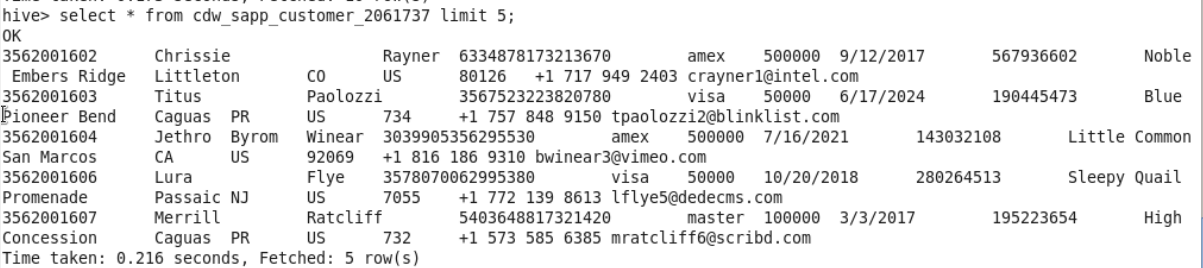
TBLPROPERTIES("SKIP.HEADER.LINE.COUNT"="1");

**//loading data from local folder to hive cdw\_sapp\_customer\_2061693 table**

LOAD DATA LOCAL INPATH 'Customer.csv' overwrite into table CDW\_SAPP\_CUSTOMER\_2061693;

// **showing cdw\_sapp\_customer\_2061693 table**

SELECT \* FROM CDW\_SAPP\_CUSTOMER\_2061693;



**//Create command for branch table**

CREATE EXTERNAL TABLE CDW\_SAPP\_BRANCH\_2061693 (

BRANCH\_CODE INT,

BRANCH\_NAME STRING,

BRANCH\_STREET STRING,

BRANCH\_CITY STRING,

BRANCH\_STATE STRING,

BRANCH\_ZIP INT,

BRANCH\_PHONE STRING)

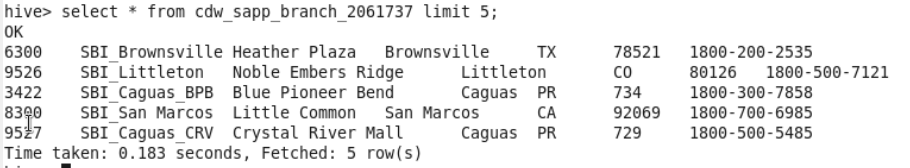
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE TBLPROPERTIES("SKIP.HEADER.LINE.COUNT"="1");

**//loading data from local folder to hive table**

LOAD DATA LOCAL INPATH 'BRANCH.CSV' OVERWRITE INTO TABLE CDW\_SAPP\_BRANCH\_2061693;

// **showing cdw\_sapp\_branch\_2061693 table**

SELECT \* FROM CDW\_SAPP\_BRANCH\_2061693;



**//Create command for creditcard fact table**

CREATE EXTERNAL TABLE CDW\_SAPP\_CREDITCARD\_2061693 (

TRAN\_ID INT,

TRAN\_DATE STRING,

CUST\_ID BIGINT,

CUST\_CC\_NO BIGINT,

CUST\_SSN BIGINT,

BRANCH\_CODE INT,

TRAN\_TYPE STRING,

TRAN\_VALUE INT)

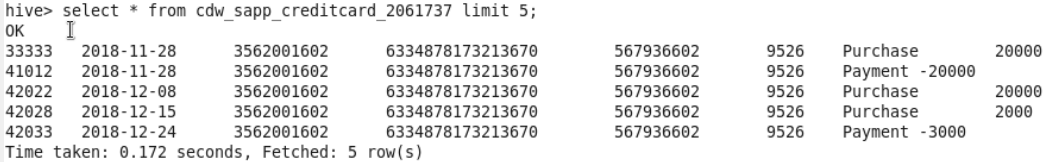
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE TBLPROPERTIES("SKIP.HEADER.LINE.COUNT"="1");

**//loading data from local folder to hive table**

LOAD DATA LOCAL INPATH 'Tran\_creditcard.csv' overwrite into table CDW\_SAPP\_creditcard\_2061693;

// **showing cdw\_sapp\_creditcard\_2061693 table**

SELECT \* FROM CDW\_SAPP\_CREDITCARD\_2061693;



**//Create command for time table**

CREATE EXTERNAL TABLE CDW\_SAPP\_TIME\_2061693 (

TIME\_ID STRING,

DAY INT,

MONTH INT,

QUARTER INT,

YEAR INT)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE;

**//creating data file ‘Time’ for time table using data from cdw\_sapp\_creditcard\_2061693**

INSERT OVERWRITE LOCAL DIRECTORY '/HOME/CLOUDERA/TIME' ROW FORMAT DELIMITED FIELDS TERMINATED BY ','

SELECT DISTINCT TRAN\_DATE AS TIMEID,

SUBSTR(TRAN\_DATE,9,2) AS DAY,

SUBSTR(TRAN\_DATE,6,2) AS MONTH,

(CASE

WHEN SUBSTR(TRAN\_DATE,6,2) >= 1 AND SUBSTR(TRAN\_DATE,6,2) <= 3 THEN 1

WHEN SUBSTR(TRAN\_DATE,6,2) >= 4 AND SUBSTR(TRAN\_DATE,6,2) <= 6 THEN 2

WHEN SUBSTR(TRAN\_DATE,6,2) >= 7 AND SUBSTR(TRAN\_DATE,6,2) <= 9 THEN 3

WHEN SUBSTR(TRAN\_DATE,6,2) >= 10 AND SUBSTR(TRAN\_DATE,6,2) <= 12 THEN 4

END) AS QUARTER,

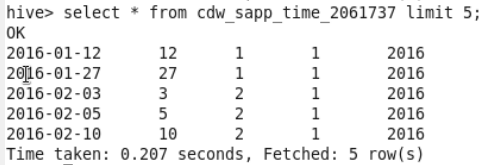
SUBSTR(TRAN\_DATE,1,4) AS YEAR FROM CDW\_SAPP\_CREDITCARD\_2061693;

**//loading data from local folder to hive table**

LOAD DATA LOCAL INPATH 'Time' overwrite into table CDW\_SAPP\_time\_2061693;

// **showing cdw\_sapp\_time\_2061693 table**

SELECT \* FROM CDW\_SAPP\_TIME\_2061693;



**FUNCTIONAL REQUIREMENTS**

CUSTOMER MODULE

1. **To store credit card details of customers.**

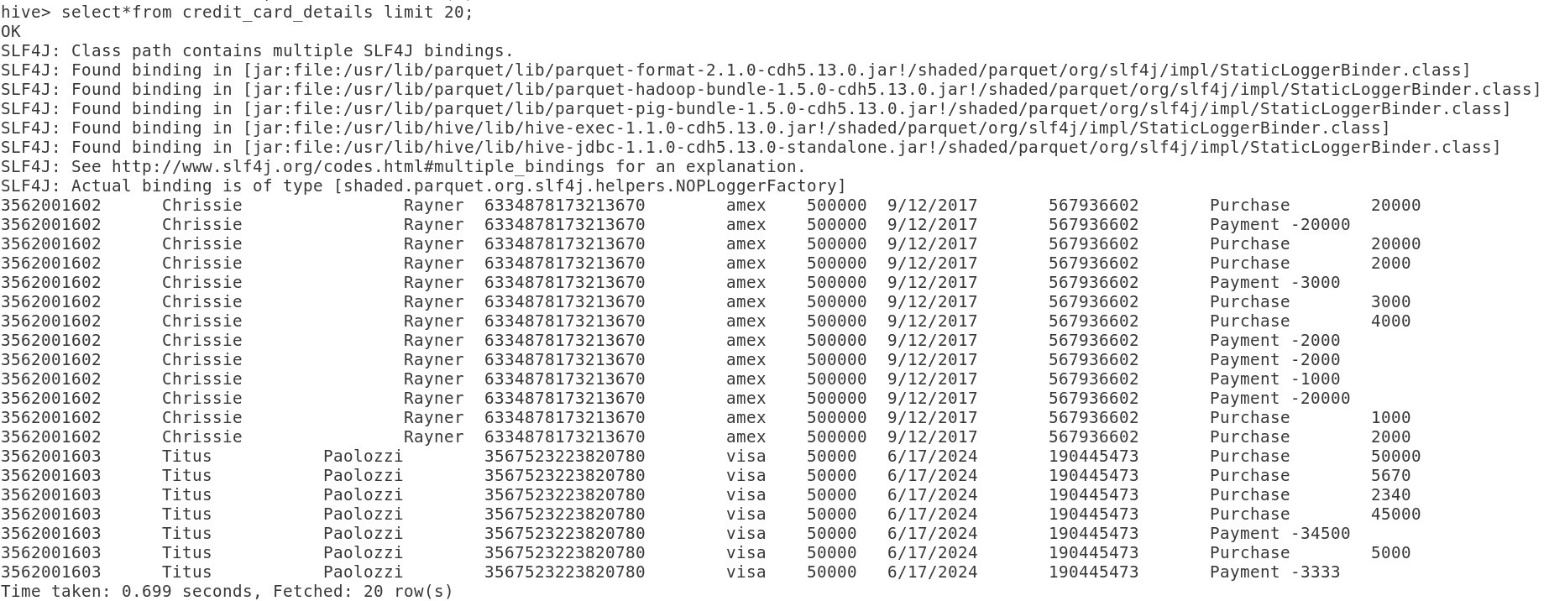
**//in spark shell**

**//to store data in a RDD**

val cc\_details = sqlContext.sql(" select t.cust\_id, t.cust\_f\_name, t.cust\_m\_name, t.cust\_l\_name, t.cust\_cc\_no, t.card\_type, t.card\_limit, t.card\_end\_date, t.cust\_ssn, c.tran\_type, c.tran\_value from project.cdw\_sapp\_creditcard\_2061693 c, project.cdw\_sapp\_customer\_2061693 t where c.cust\_id=t.cust\_id")

**//to save output back to hive**

cc\_details.write.mode("overwrite").saveAsTable("project.credit\_card\_details");



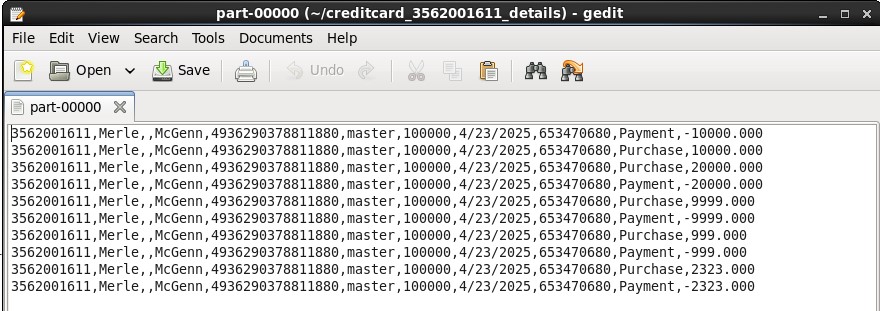
1. **To fetch the card details of a customer.**

**//to store data in a RDD**

val cust\_ccdetails = sqlContext.sql(" select\*from project.credit\_card\_details where cust\_id=3562001611");

**//to save output as textfile in the local system**

val op= cust\_ccdetails.rdd.map(\_.toString().replace("[","").replace("]", "")).saveAsTextFile("file:/home/cloudera/creditcard\_3562001611\_details")



**//Creating temporary table in RDD using data in hive table**

**//customer table**

val customers = sqlContext.sql("select \* from project.cdw\_sapp\_customer\_2061693")

customers.registerTempTable("customer")

**//creditcard table**

val creditcard = sqlContext.sql("select \* from project.cdw\_sapp\_creditcard\_2061693")

creditcard.registerTempTable("credit\_card")

**//time table**

val time= sqlContext.sql("select \* from project.cdw\_sapp\_time\_2061693")

time.registerTempTable("time")

**//branch table**

val branch = sqlContext.sql("select \* from project.cdw\_sapp\_branch\_2061693")

branch.registerTempTable("branch")

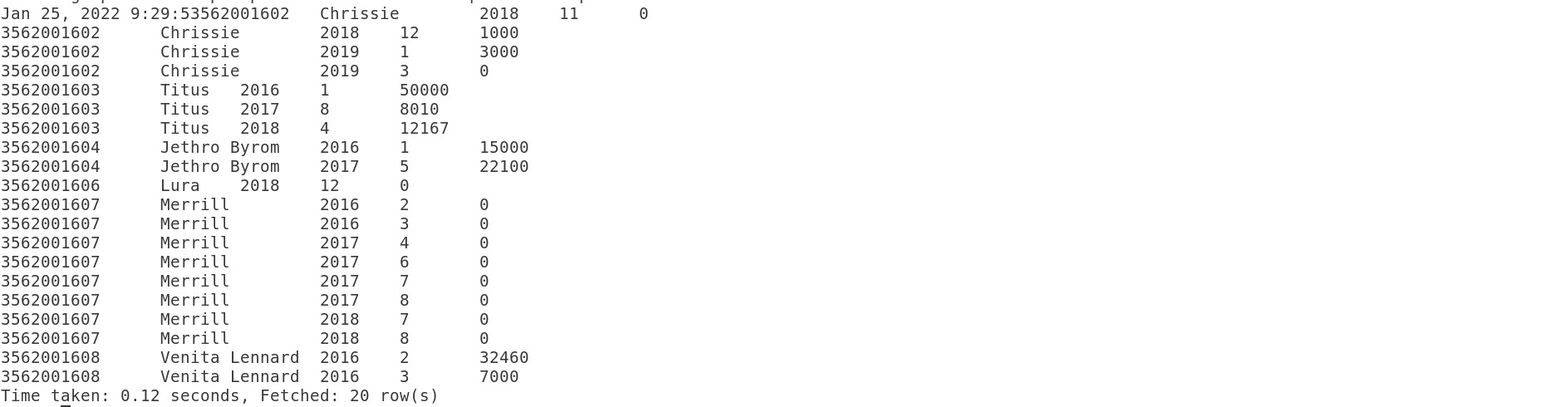
1. **To generate monthly bill for a credit card number.**

**//to store data in a RDD**

val bill = sqlContext.sql("select c.cust\_id customer\_id,name,year,month,sum(tran\_value) bill from credit\_card c join time t on c.tran\_date=t.time\_id join (select cust\_id,concat(cust\_f\_name,' ',cust\_l\_name) as name from customer) as cs1 on c.cust\_id=cs1.cust\_id group by c.cust\_id,name,year,month order by customer\_id,year,month")

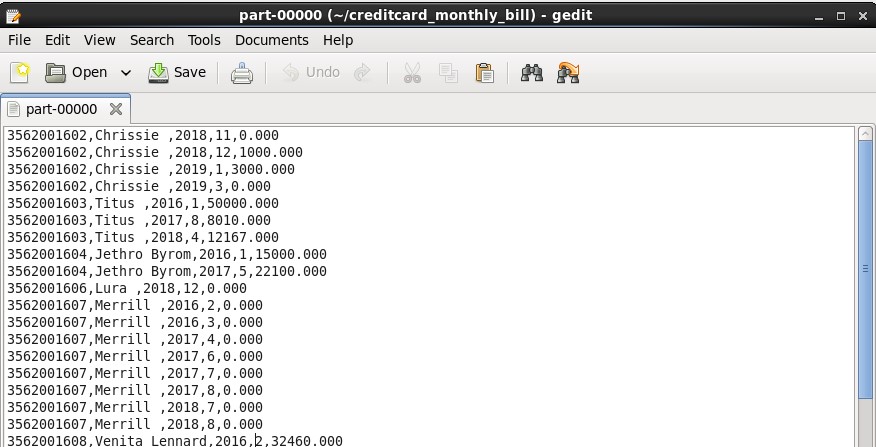
**//to save output back to hive**

bill.write.mode("overwrite").saveAsTable("project.monthly\_bill");



**//to save output as textfile in the local system**

val billoutput= bill.rdd.map(\_.toString().replace("[","").replace("]", "")).coalesce(1).saveAsTextFile("file:/home/cloudera/creditcard\_monthly\_bill")



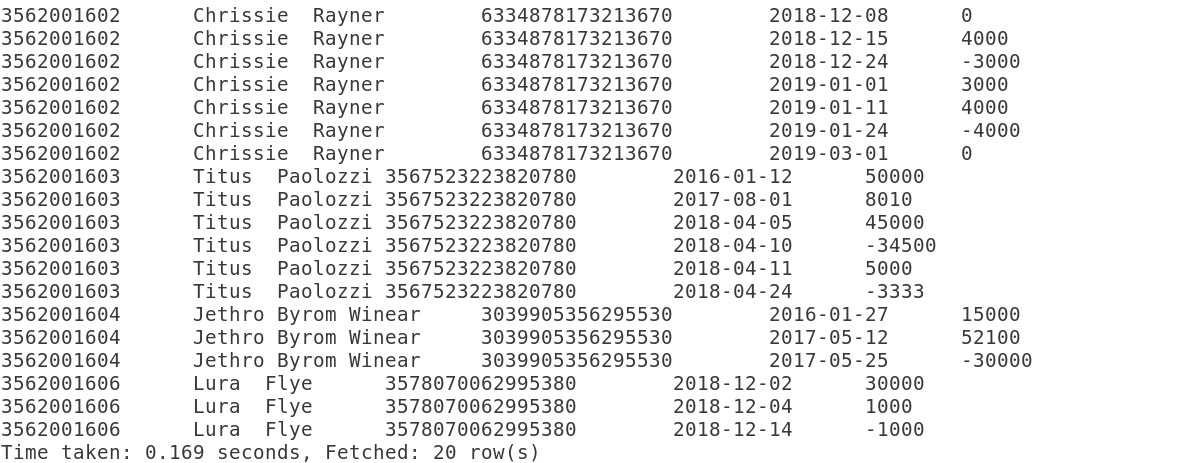
1. **To display the transactions made by a customer (daily, weekly, monthly basis).**

**//transaction on daily basis**

VAL DAILY= SQLCONTEXT.SQL("SELECT C.CUST\_ID, CONCAT(C.CUST\_F\_NAME,' ', C.CUST\_M\_NAME,' ', C.CUST\_L\_NAME) AS NAME, T.CUST\_CC\_NO, T.TRAN\_DATE, SUM(T.TRAN\_VALUE) AS TOTAL\_AMOUNT FROM CUSTOMER C, CREDIT\_CARD T WHERE T.CUST\_CC\_NO=C.CUST\_CC\_NO GROUP BY C.CUST\_ID, T.CUST\_CC\_NO, T.TRAN\_DATE, C.CUST\_F\_NAME, C.CUST\_M\_NAME, C.CUST\_L\_NAME ORDER BY C.CUST\_ID, T.TRAN\_DATE")

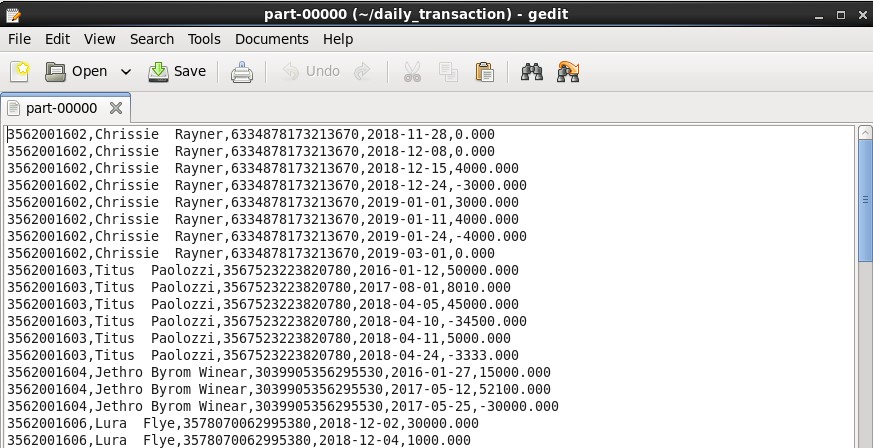
**//to save output back to hive**

daily.write.mode("overwrite").saveAsTable("project.daily\_transaction");



**//to save output as textfile in the local system**

val sa= daily.rdd.map(\_.toString().replace("[","").replace("]", "")).coalesce(1).saveAsTextFile("file:/home/cloudera/daily\_transaction")

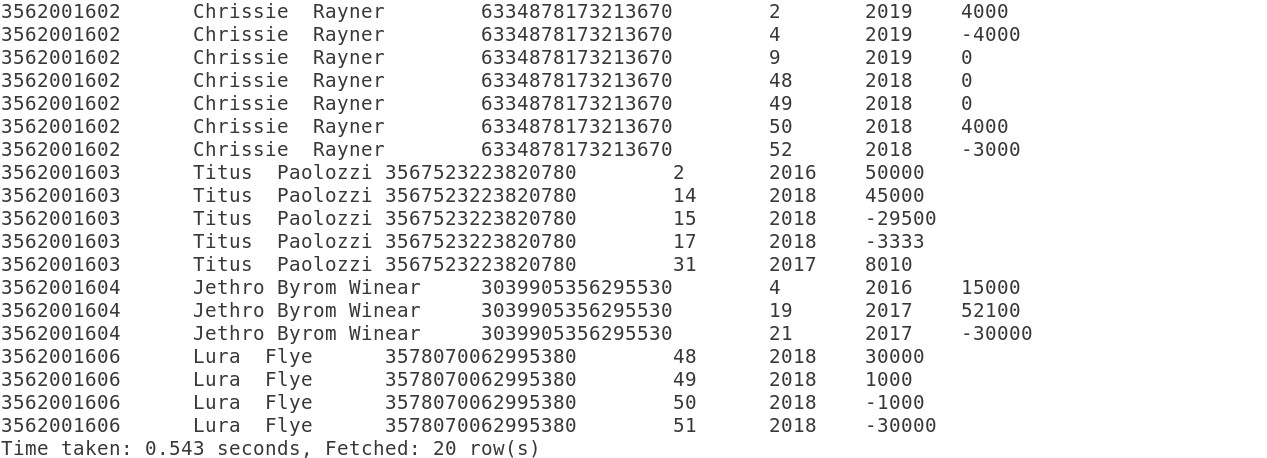


**//transaction on weekly basis**

val week= sqlContext.sql("select c.cust\_id, concat(c.CUST\_F\_NAME,' ', c.CUST\_M\_NAME,' ', c.CUST\_L\_NAME) as NAME, t.CUST\_CC\_NO, weekofyear(t.tran\_date) as week, year(t.tran\_date) as year, sum(t.TRAN\_VALUE) as TOTAL\_AMOUNT from customer c, credit\_card t where t.cust\_cc\_no=c.cust\_cc\_no group by c.cust\_id, t.CUST\_CC\_NO, weekofyear(t.tran\_date), year(tran\_date), c.CUST\_F\_NAME, c.CUST\_M\_NAME, c.CUST\_L\_NAME order by c.cust\_id, weekofyear(t.tran\_date), year(t.tran\_date)")

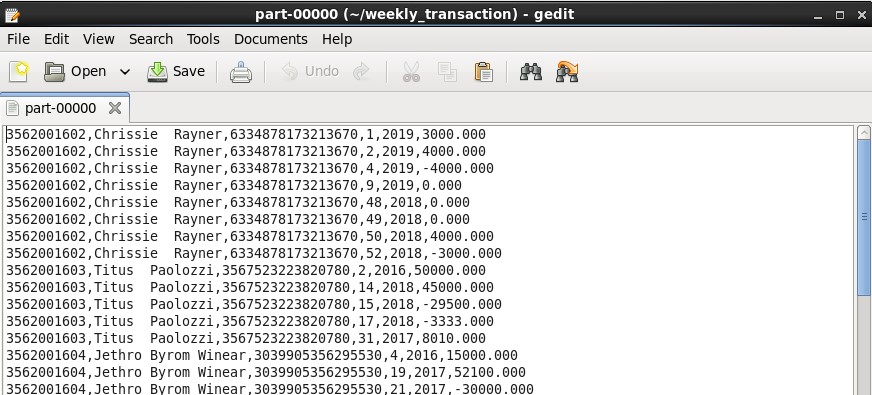
**//to save output back to hive**

week.write.mode("overwrite").saveAsTable("project.weekly\_transaction");



**//to save output as textfile in the local system**

val sa= week.rdd.map(\_.toString().replace("[","").replace("]", "")).coalesce(1).saveAsTextFile("file:/home/cloudera/weekly\_transaction")

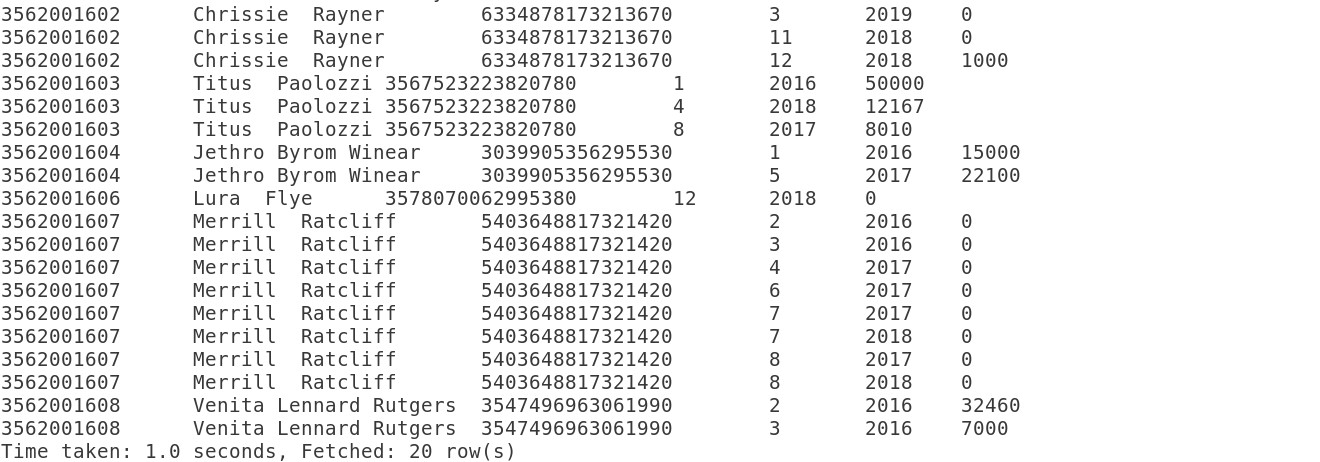


**//transaction on monthly basis**

val month= sqlContext.sql("select c.cust\_id, concat(c.CUST\_F\_NAME,' ', c.CUST\_M\_NAME,' ', c.CUST\_L\_NAME) as NAME, t.CUST\_CC\_NO, month(t.tran\_date) as month, year(t.tran\_date) as year, sum(t.tran\_VALUE) as TOTAL\_AMOUNT from customer c, credit\_card t where t.cust\_cc\_no=c.cust\_cc\_no group by c.cust\_id, t.CUST\_CC\_NO, month(t.tran\_date), year(tran\_date), c.CUST\_F\_NAME, c.CUST\_M\_NAME, c.CUST\_L\_NAME order by c.cust\_id, month(t.tran\_date), year(t.tran\_date)")

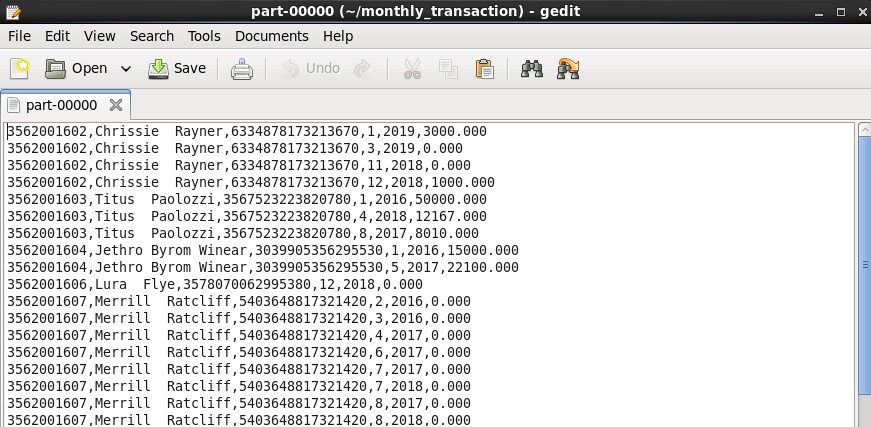
**//to save output back to hive**

month.write.mode("overwrite").saveAsTable("project.monthly\_transaction");



**//to save output as textfile in the local system**

val sa= month.rdd.map(\_.toString().replace("[","").replace("]", "")).coalesce(1).saveAsTextFile("file:/home/cloudera/monthly\_transaction")



1. **To fetch customer details who have overdue.**

val overdue= sqlContext.sql("select id,name,mon,year,tot from (select cr.cust\_id as id,cust\_f\_name as name,mon,year,sum(tran\_value) as tot from credit\_card cr join customer cs on cr.cust\_id=cs.cust\_id join time tm on cr.tran\_date=tm.tran\_date group by cr.cust\_id,cust\_f\_name,mon,year order by id,year,mon) as sample where tot < 0")

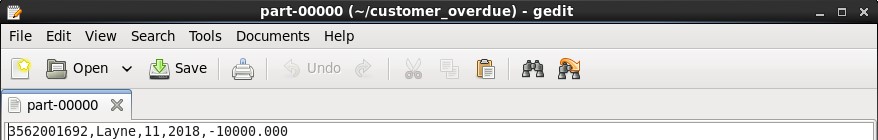
**//to save output back to hive**

overdue.write.mode("overwrite").saveAsTable("project.customer\_overdue");



**//to save output as textfile in the local system**

val sa= overdue.rdd.map(\_.toString().replace("[","").replace("]", "")).coalesce(1).saveAsTextFile("file:/home/cloudera/customer\_overdue")



1. **To display customer details based on**

* **Card type**

val card\_type= sqlContext.sql("select cust\_id,concat(cust\_f\_name,' ',cust\_m\_name,' ',cust\_L\_name) as NAME, cust\_cc\_no, cust\_ssn, cust\_street, cust\_email from customer where card\_type like 'visa'")

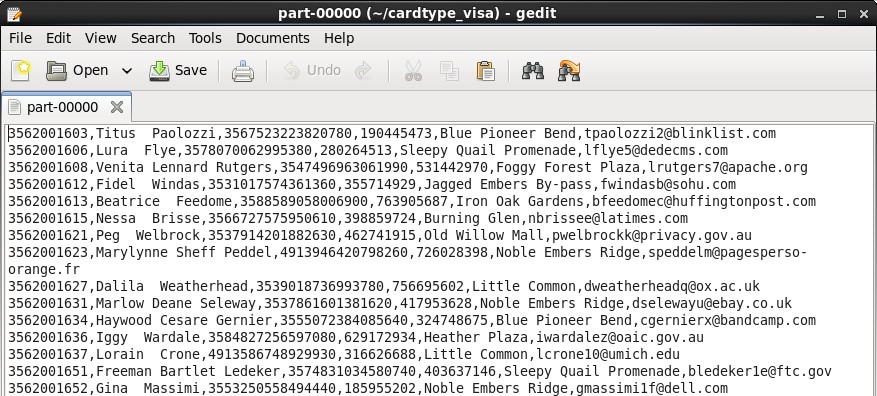
**//to save output back to hive**

card\_type.write.mode("overwrite").saveAsTable("project.cardtype\_visa");



**//to save output as textfile in the local system**

val sa= card\_type.rdd.map(\_.toString().replace("[","").replace("]", "")).coalesce(1).saveAsTextFile("file:/home/cloudera/cardtype\_visa")

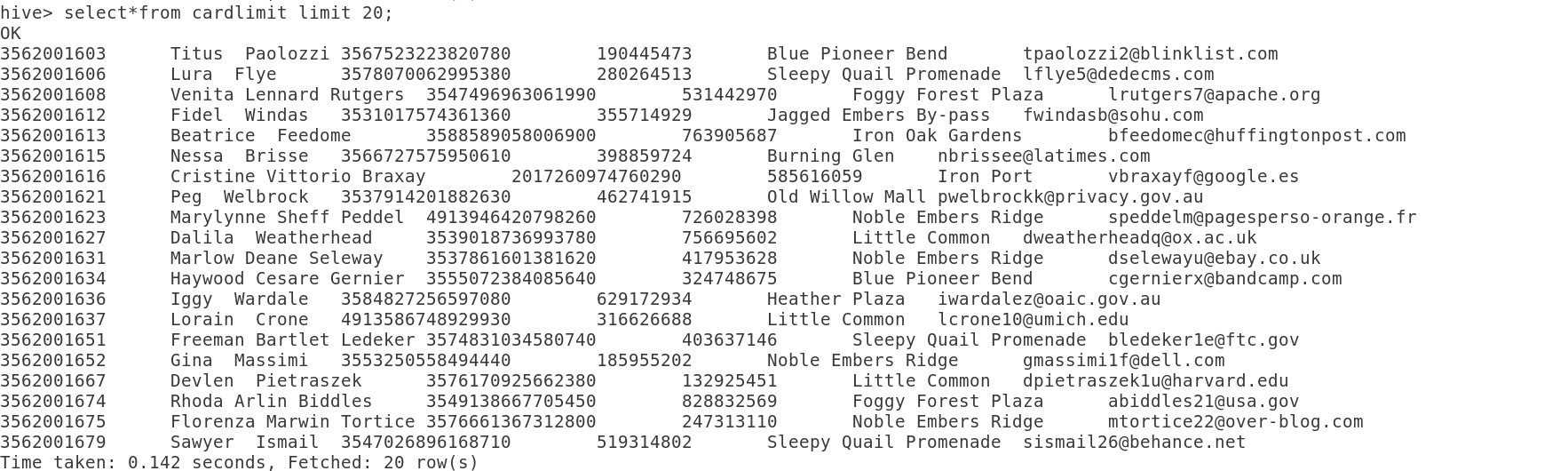


* **Credit limit range**

val card\_limit= sqlContext.sql("select cust\_id,concat(cust\_f\_name,' ',cust\_m\_name,' ',cust\_L\_name) as NAME, cust\_cc\_no, cust\_ssn, cust\_street, cust\_email from customer where card\_limit <= 50000")

**//to save output back to hive**

card\_limit.write.mode("overwrite").saveAsTable("project.cardlimit");

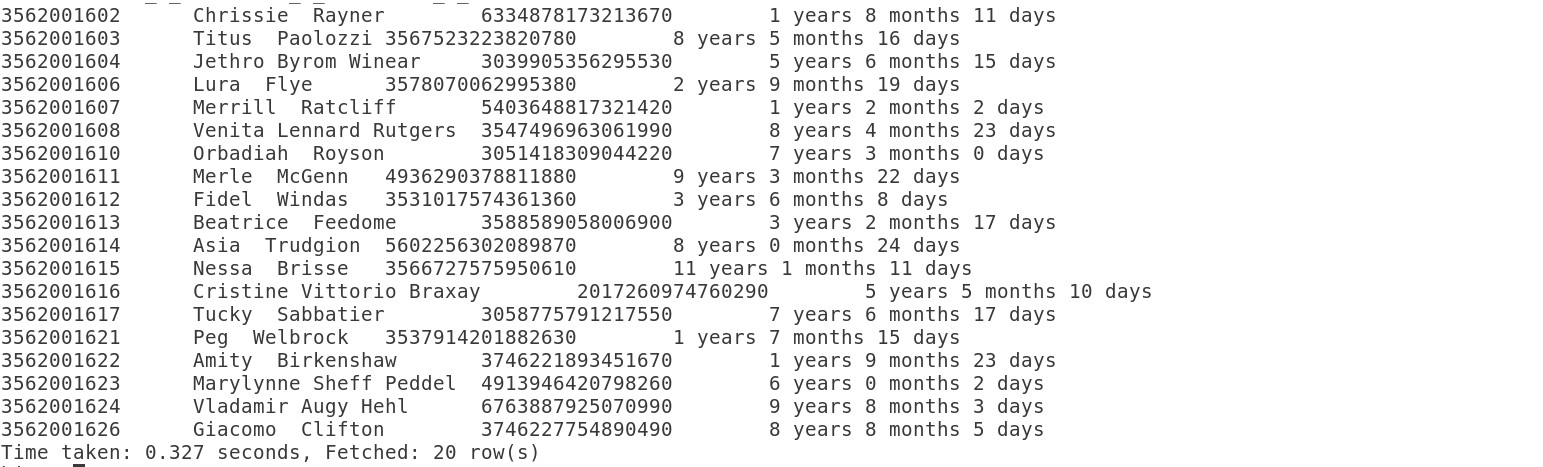


1. **To check the credit card validity.**

val validity =sqlContext.sql("select id,name,cc,concat(yr,' years ',mn,' months ',dy,' days') from (select cust\_id as id,concat(cust\_f\_name,' ',cust\_m\_name,' ',cust\_L\_name) as name,cust\_cc\_no as cc, (cast(year(from\_unixtime(unix\_timestamp(card\_end\_date, 'mm/dd/yyyy'),'yyyy-mm-dd')) as int)-cast(year('2016-01-01') as int)) as yr, (cast(month(from\_unixtime(unix\_timestamp(card\_end\_date, 'mm/dd/yyyy'),'yyyy-mm-dd')) as int)-cast(month('2016-01-01') as int)) as mn, (cast(day(from\_unixtime(unix\_timestamp(card\_end\_date, 'mm/dd/yyyy'),'yyyy-mm-dd')) as int)-cast(day('2016-01-01') as int)) as dy from customer) as enddate")

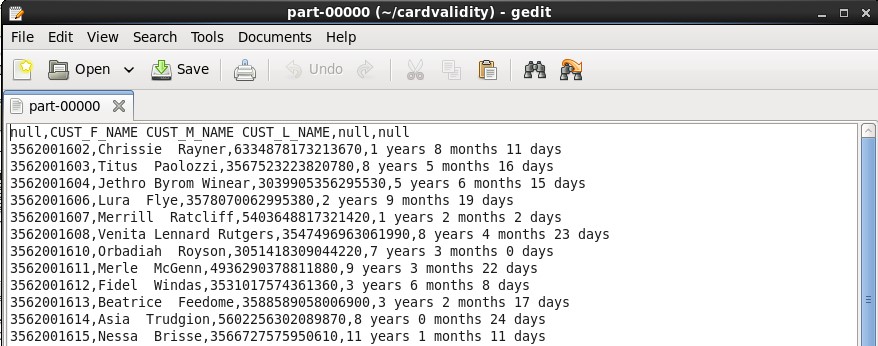
**//to save output back to hive**

validity.write.mode("overwrite").saveAsTable("project.cardvalidity");



**//to save output as textfile in the local system**

val sa= validity.rdd.map(\_.toString().replace("[","").replace("]", "")).coalesce(1).saveAsTextFile("file:/home/cloudera/cardvalidity")



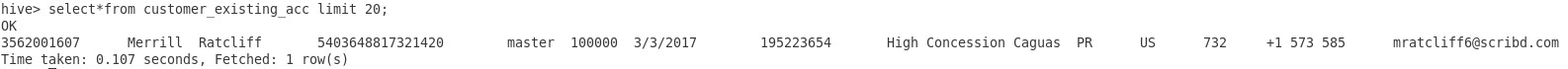
BANK MODULE

1. **To check for existing account details of a customer.**

val existing\_acc=sqlContext.sql("select cust\_id,concat(cust\_f\_name,' ',cust\_m\_name,' ',cust\_L\_name) as cust\_name,CUST\_CC\_NO,card\_type,card\_limit,CARD\_END\_DATE,CUST\_SSN,CUST\_STREET,CUST\_CITY,CUST\_STATE,CUST\_COUNTRY,CUST\_ZIP,CUST\_PHONE,CUST\_EMAIL from customer where cust\_ssn=195223654 order by cust\_id")

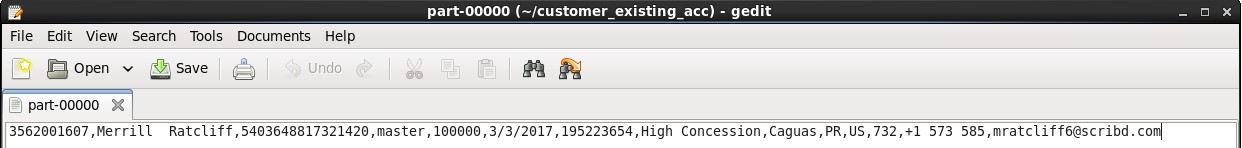
**//to save output back to hive**

existing\_acc.write.mode("overwrite").saveAsTable("project.customer\_existing\_acc")



**//to save output as textfile in the local system**

val sa= existing\_acc.rdd.map(\_.toString().replace("[","").replace("]", "")).coalesce(1).saveAsTextFile("file:/home/cloudera/customer\_existing\_acc")

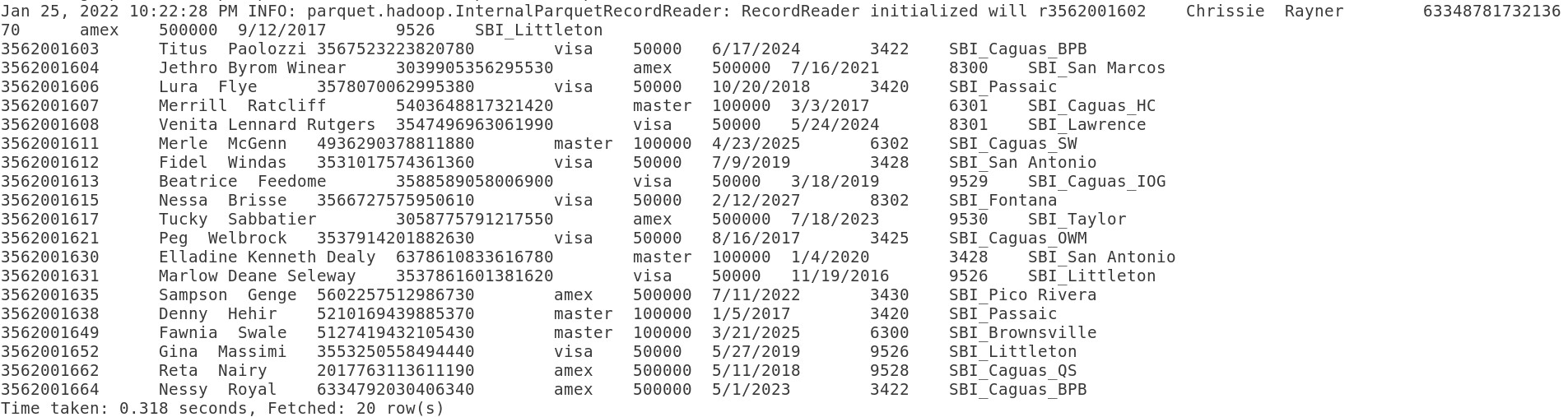


1. **To display Account Information for a customer.**

val acc=sqlContext.sql("select cs.cust\_id,concat(cust\_f\_name,' ',cust\_m\_name,' ',cust\_L\_name) as cust\_name,cs.CUST\_CC\_NO,card\_type,card\_limit,CARD\_END\_DATE,cr.branch\_code,br.branch\_name from credit\_card cr join customer cs on cr.cust\_id=cs.cust\_id join branch br on br.branch\_code=cr.branch\_code order by cs.cust\_id").distinct

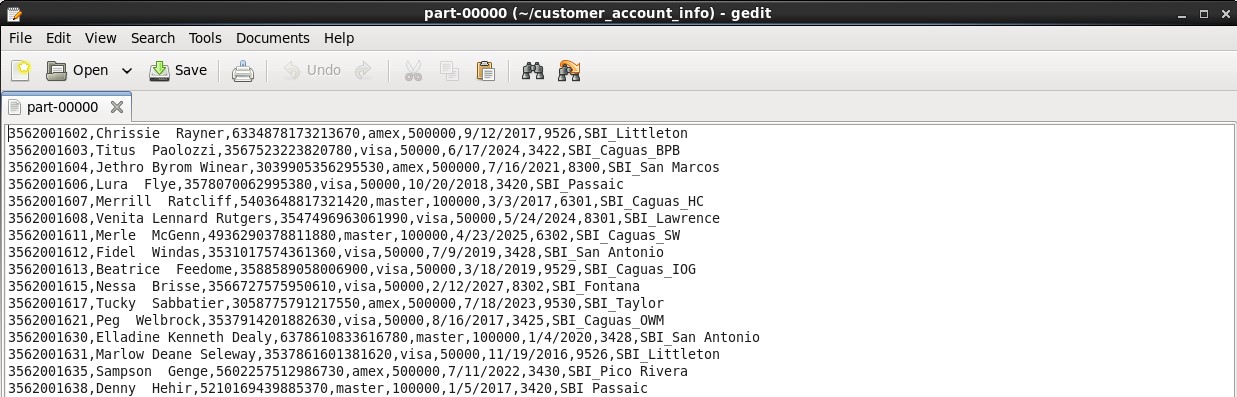
**//to save output back to hive**

acc.write.mode("overwrite").saveAsTable("project.customer\_account\_info")



**//to save output as textfile in the local system**

val sa=acc.rdd.map(\_.toString().replace("[","").replace("]","")).coalesce(1).saveAsTextFile("file:/home/cloudera/customer\_account\_info")



**EXTRAS**

1. **Total credit limit for each customer**

val cr\_limit= sqlContext.sql("SELECT CUST\_ID, CONCAT(CUST\_F\_NAME,' ', C.CUST\_M\_NAME,' ', C.CUST\_L\_NAME) AS NAME,card\_limit from project.cdw\_sapp\_customer\_2061693”)

cr\_limit.show()

1. **Available credit limit for each customer**

val overdue= sqlContext.sql("select id,name,os from (select cr.cust\_id as id,cust\_f\_name as name,

sum(tran\_value) as os from project.cdw\_sapp\_creditcard\_2061693 cr join project.cdw\_sapp\_customer\_2061693 cs on cr.cust\_id=cs.cust\_id group by cr.cust\_id,cust\_f\_name) as sample")

overdue.registerTempTable("overdue");

cr\_limit.registerTempTable("cr\_limit");

overdue.write.mode("overwrite").saveAsTable("project.customer\_overdue");

cr\_limit.write.mode("overwrite").saveAsTable("project.customer\_cr\_limit");

val avl\_limit= sqlContext.sql("select o.id, l.NAME, (l.card\_limit-o.os) as AVL\_LIMIT from overdue o join cr\_limit l on o.id = l.CUST\_ID")

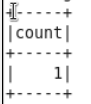
**//to save output back to hive**

avl\_limit.write.mode("overwrite").saveAsTable("project.cardvalidity");

**3.How many customers have invalid or null email id**

val invalid\_cnt=sqlContext.sql("select count (cust\_email) as count from project.cdw\_sapp\_customer\_2061693 where cust\_email not like '%\_@\_%.\_%' or cust\_email is null")

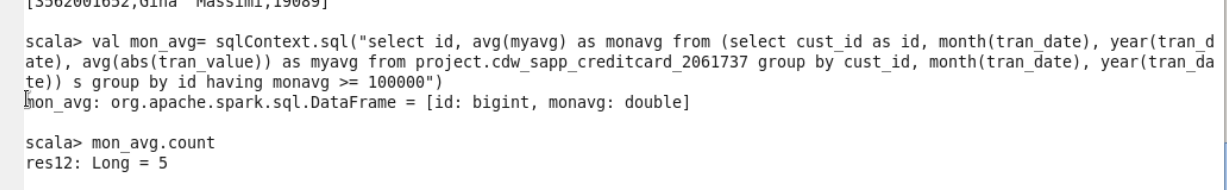
invalid\_cnt.show()



**4. Total customers who do average 1L transaction/purchase per month**

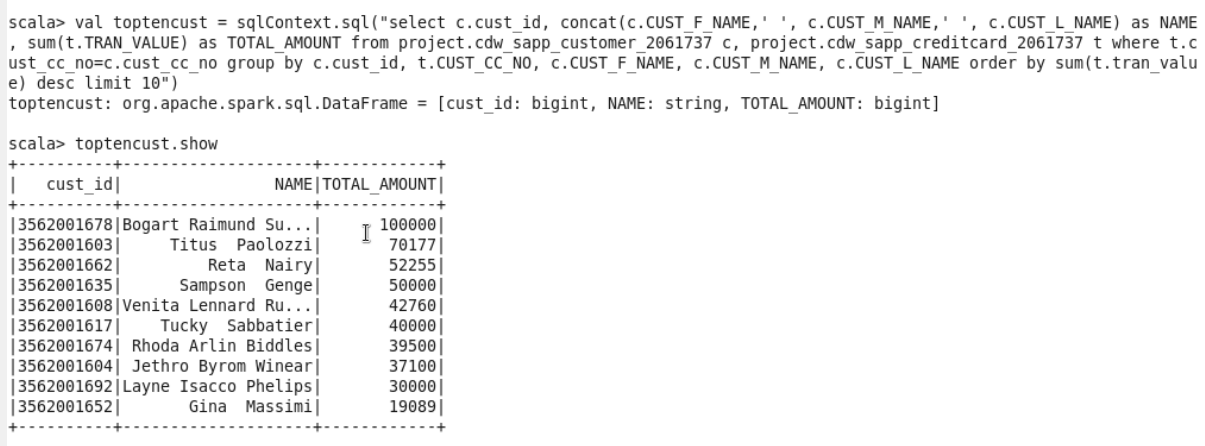
val mon\_avg= sqlContext.sql("select id, avg(myavg) as monavg from (select cust\_id as id, month(tran\_date), year(tran\_date), avg(abs(tran\_value)) as myavg from project.cdw\_sapp\_creditcard\_2061693 group by cust\_id, month(tran\_date), year(tran\_date)) s group by id having monavg >= 100000")

mon\_avg.count()



**5. Fetch top 10 customers who is doing more transactions**

val toptencust = sqlContext.sql("select c.cust\_id, concat(c.CUST\_F\_NAME,' ', c.CUST\_M\_NAME,' ', c.CUST\_L\_NAME) as NAME, sum(t.TRAN\_VALUE) as TOTAL\_AMOUNT from project.cdw\_sapp\_customer\_2061693 c, project.cdw\_sapp\_creditcard\_2061693 t where t.cust\_cc\_no=c.cust\_cc\_no group by c.cust\_id, t.CUST\_CC\_NO, c.CUST\_F\_NAME, c.CUST\_M\_NAME, c.CUST\_L\_NAME order by sum(t.tran\_value) desc limit 10")

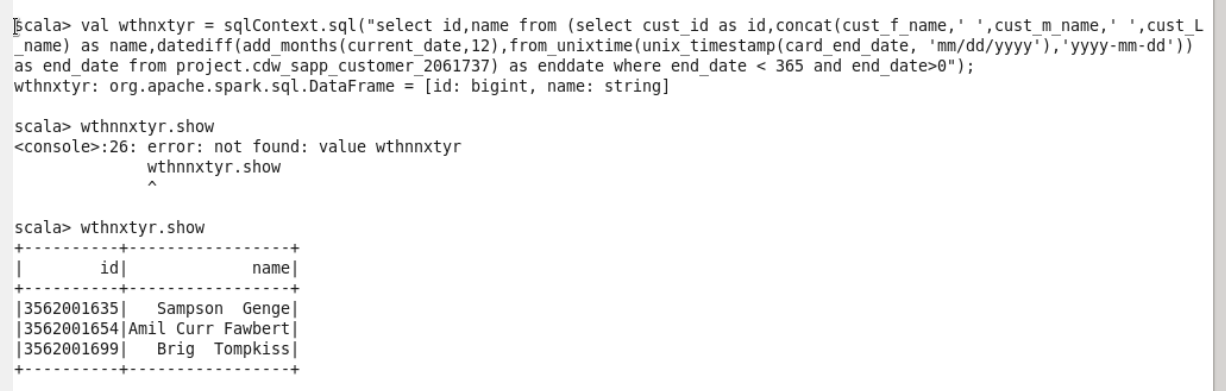


**//to save output back to hive**

toptencust.write.mode("overwrite").saveAsTable("project.cardvalidity");

**6. List of customers whose credit card needs to be renewed within next 1 year**

val wthnxtyr = sqlContext.sql("select id,name from (select cust\_id as id,concat(cust\_f\_name,' ',cust\_m\_name,' ',cust\_L\_name) as name,datediff(add\_months(current\_date,12),from\_unixtime(unix\_timestamp(card\_end\_date, 'mm/dd/yyyy'),'yyyy-mm-dd')) as end\_date from project.cdw\_sapp\_customer\_2061693) as enddate where end\_date < 365 and end\_date>0");



**//to save output back to hive**

wthnxtyr.write.mode("overwrite").saveAsTable("project.cardvalidity");