Setting up the Environment -

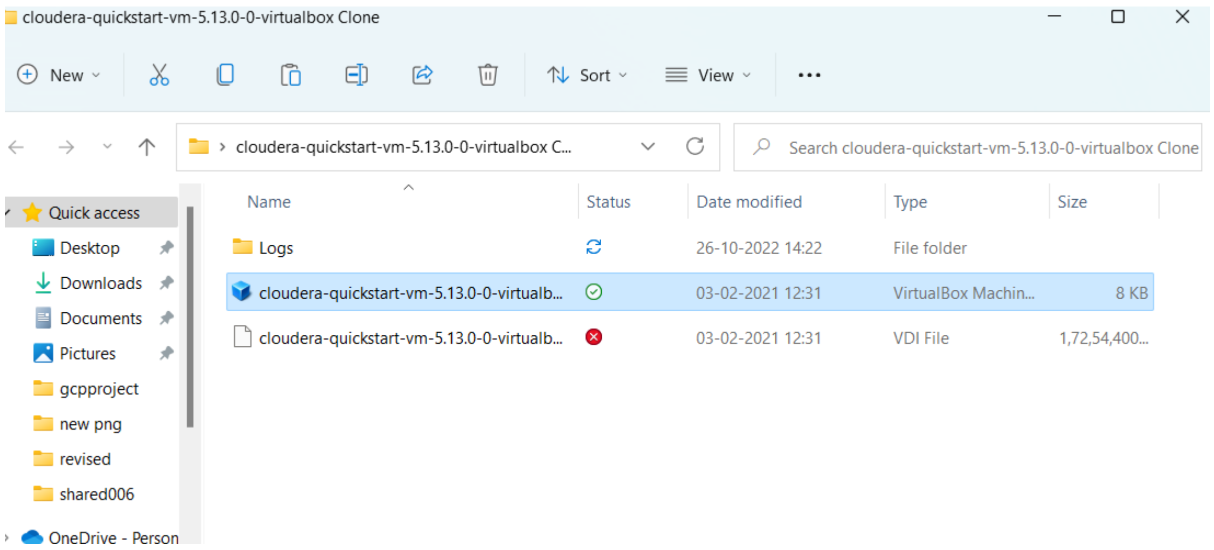
Airflow Download and Setup Steps:

-Download Virtual Box and install
https://www.virtualbox.org/wiki/Downloads
Select the respective host Operating System and download the file and install.

-We are providing a VM which has airflow installed called a clone VM. **Just download , import the VM and start using it as steps given below**

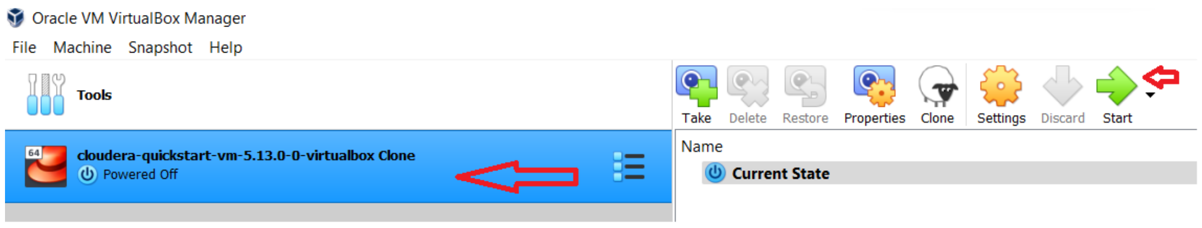
https://drive.google.com/drive/folders/1fXViofZOqdvX2bKY3RVwyTpCbgvFS7iI?usp=sharing

-Extract the files from the zip. Open the cloudera-quickstart-vm folder that gets created after extracting the zip. Double click on the cloudera-quickstart-vm-5.13.0-0-virtualb… as highlighted in the screenshot below.


cloudera-quickstart-vm-5.13.0-0-virtualbox Clone
New
Sort
View
cloudera-quickstart-vm-5.13.0-0-virtualbox C...
Search cloudera-quickstart-vm-5.13.0-0-virtualbox Clone
Quick access
Desktop
Downloads
Documents
Pictures
gcpproject
new png
revised
shared006
OneDrive - Person

Name	Status	Date modified	Type	Size
Logs		26-10-2022 14:22	File folder	
cloudera-quickstart-vm-5.13.0-0-virtualb...	✓	03-02-2021 12:31	VirtualBox Machin...	8 KB
cloudera-quickstart-vm-5.13.0-0-virtualb...	✗	03-02-2021 12:31	VDI File	1,72,54,400...

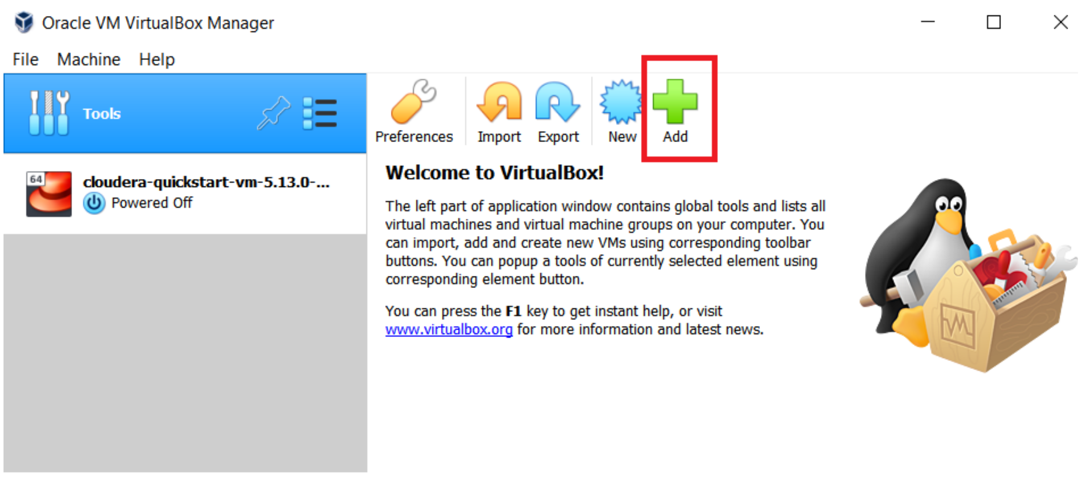
This will open the file in the virtual box. Select the important VM and click on **Start** as shown below.



[**ERRORS & SOLUTIONS**
If you are unable to import the VM and if you see a .vdi instead of .ovf after extracting, follow the steps below to import the VM to Virtual box

Either right click on the .vdi file and open with Virtual box

OR



Welcome to VirtualBox!

The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for more information and latest news.

Click on the **+ADD** option in the Virtual Box and select the .vdi file. This should start the import with default values.
]

Additional Background Tasks:

1. Create a **shared folder** as explained in the course to transfer the files from host to cloudera VM. You could refer to the 3 General Things section of Week1 for shared folder creation steps.

2. Drive Link to Download BigData_Project_Demo Folder. This contains all the scripts used in the project.
https://drive.google.com/drive/folders/1eWgaOZN54aePfTbtIm8oeawvjtFkEY-s?usp=sharing
[Note: If you downloaded the BigData_Project_Demo on your host windows or mac machine, transfer it to cloudera VM using Shared folder]

Follow below steps to setup airflow

1. **su** (pwd: cloudera)
2. **cd airflow**
3. **source venv/bin/activate**
4. **export AIRFLOW_HOME=$PWD**
5. Create dag folder if not exists
[mkdir dags]
6. Create python file inside dag folder(for the python file, copy paste the export_card_txns.py from BigData_Project_Demo folder, present under Airflow_Scripts)

7. Run the command “**airflow webserver -p 8080**”
8. Open another terminal — do steps 1-4 — and Run the command “**airflow scheduler**”
9. Go to Browser — localhost:8080 — you can see your dag with dag id not with python file name

[**ERRORS & SOLUTIONS:**

1.If you run into errors while starting the airflow server, run the following on the same terminal -

```
pip install wheel
pip install --upgrade pip
pip install sshtunnel
```

2.If you run into PARAMIKO error, execute below commands:

```
export CRYPTOGRAPHY_DONT_BUILD_RUST=1
pip3 install --upgrade homeassistant
pip install paramiko (or pip3 install paramiko)
```

After executing this, if you encounter error of ssh tunnel, run below command:

```
pip install sshtunnel
```

3.If you encounter error: Already running on PID , execute below command:

sudo lsof -i tcp:8080   (so you will get list of running pid’s, now execute below kill command to kill all the mentioned pid’s)

```
kill -9 <pid>
```

Also remove the airflow-webserver.pid by executing below command:

```
rm airflow-webserver.pid
```

]

Tasks:

The following Datasets required for the project are present in the Project Datasets folder in the above google drive link.

1. Card Transactions History Data - card_transactions.csv
2. Member Score Dataset
3. Member Details Dataset

Task1: Copy “card_transactions.cvs” file from local system to HDFS

```
hadoop fs -mkdir project_input_data
hadoop fs -put Desktop/card_transactions.csv project_input_data/
hadoop fs -cat project_input_data/card_transactions.csv | wc -l
```

Connect to mysql

```
mysql -u root -p
Password :cloudera
```

Task2: Creating tables in MySQL (Run in MySQL terminal)

```
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)
);


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)
);
```

Task3: Sqoop export to the card_transactions table in MySQL database for card_transactions.csv(Using Airflow) and delete the file from HDFS.

Navigate to -> BigData_Project_Demo -> Airflow_Scripts
copy : export_card_txns.py

Navigate to Home Directory -> airflow -> dags
paste : export_card_txns.py

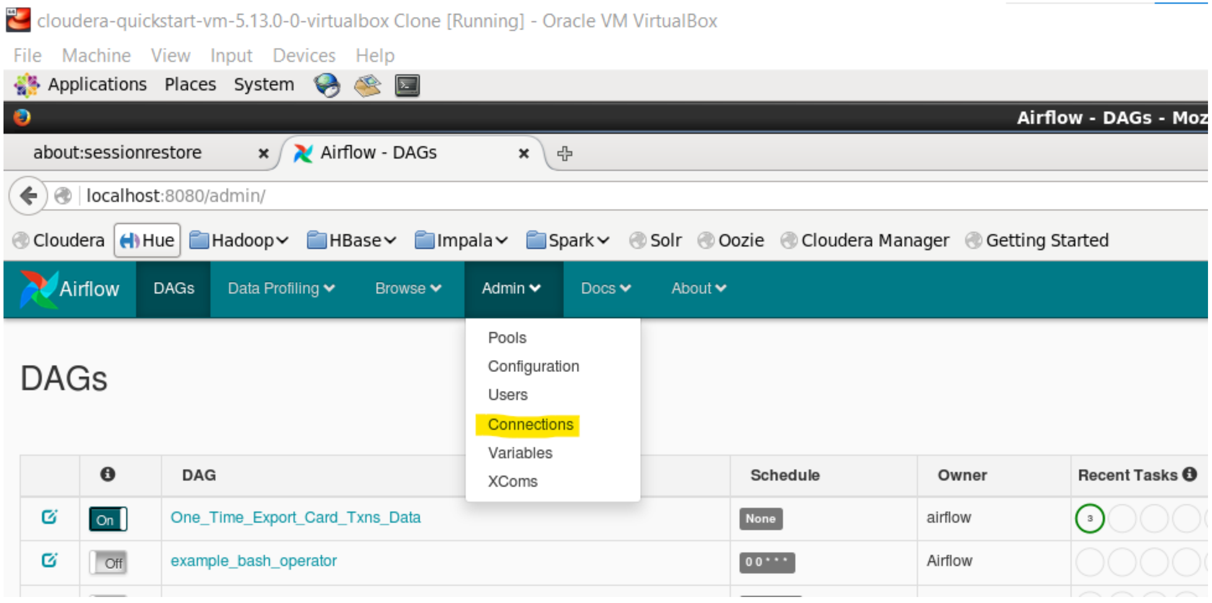
Encrypting MySQL Password - (Run in the terminal)

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

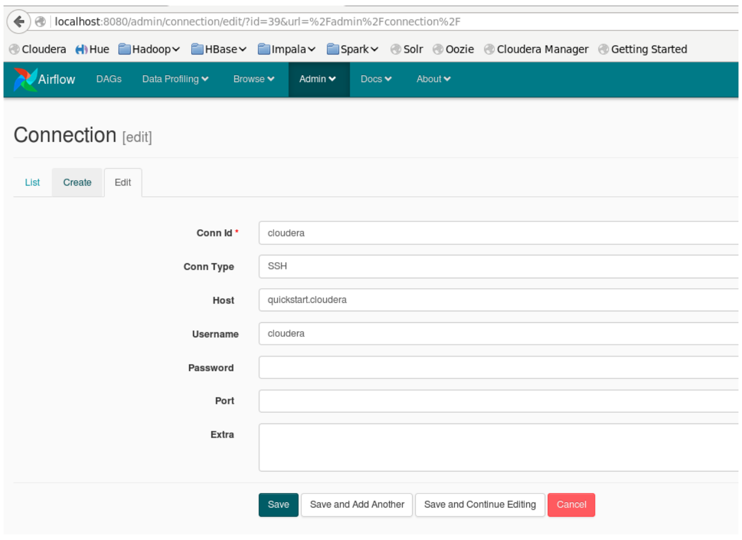
Copy the sh file (sqoop_export_card_txns.sh file to Desktop and give permission to this file :
cd Desktop
chmod 777 sqoop_export_card_txns.sh)

Creating Variables and Connections
After having started the Airflow server, When you traverse to localhost:8080 , the DAGS will be visible if the server started successfully

Under the **Admin** tab ->


cloudera-quickstart-vm-5.13.0-0-virtualbox Clone [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Airflow - DAGs - Moz
about:sessionrestore
Airflow - DAGs
localhost:8080/admin/
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Airflow DAGs Data Profiling Browse Admin Docs About
Pools
Configuration
Users
Connections
Variables
XComs
DAGs
DAG
Schedule
Owner
Recent Tasks
On
One_Time_Export_Card_Txns_Data
None
airflow
Off
example_bash_operator
Airflow

Setting up Connection
click on **Connections:**
Under **create** tab, fill in the following details and save the connection
Conn id : cloudera
Conn Type : SSH
Host : quickstart.cloudera
Username : cloudera
Password: cloudera

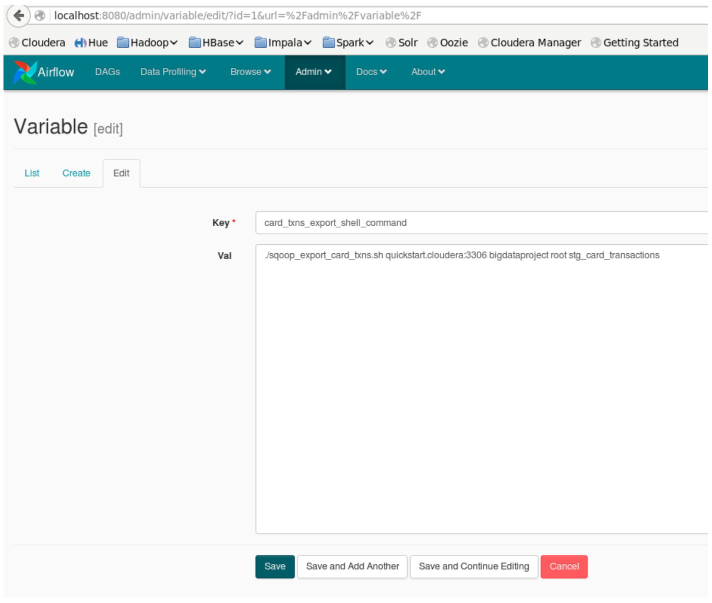

localhost:8080/admin/connection/edit/?id=39&url=%2Fadmin%2Fconnection%2F
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Airflow DAGs Data Profiling Browse Admin Docs About
List Create Edit
Save Save and Add Another Save and Continue Editing Cancel

Connection [edit]

Conn Id *	cloudera
Conn Type	SSH
Host	quickstart.cloudera
Username	cloudera
Password	
Port	
Extra	

And Save

Setting up Variable
click on **Variables:**
Under **create** tab, fill in the following details and save the connection
Key : card_txns_export_shell_command
Val : ./sqoop_export_card_txns.sh quickstart.cloudera:3306 bigdataproject root stg_card_transactions


localhost:8080/admin/variable/edit/?id=1&url=%2Fadmin%2Fvariable%2F
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
Airflow DAGs Data Profiling Browse Admin Docs About
List Create Edit
Save Save and Add Another Save and Continue Editing Cancel

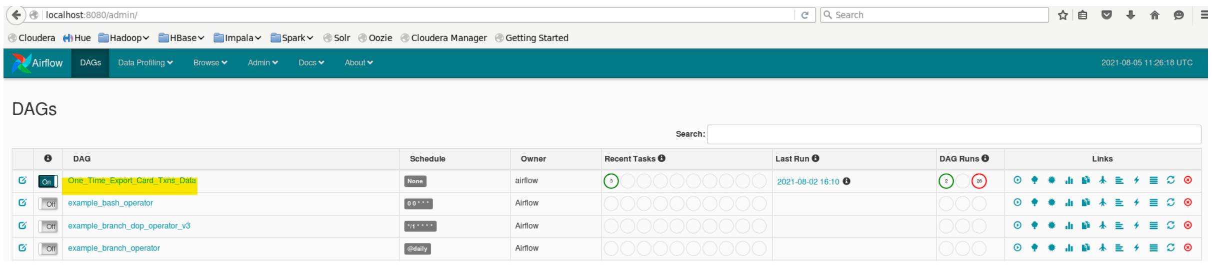
Variable [edit]

Key *	card_txns_export_shell_command
Val	./sqoop_export_card_txns.sh quickstart.cloudera:3306 bigdataproject root stg_card_transactions

And Save

Triggering the DAG

Select **One_Time_Export_Card_Txns_Data** dag


localhost:8080/admin/

Cloudera Hue Hadoop Impala Spark Solr Oozie Cloudera Manager Getting Started

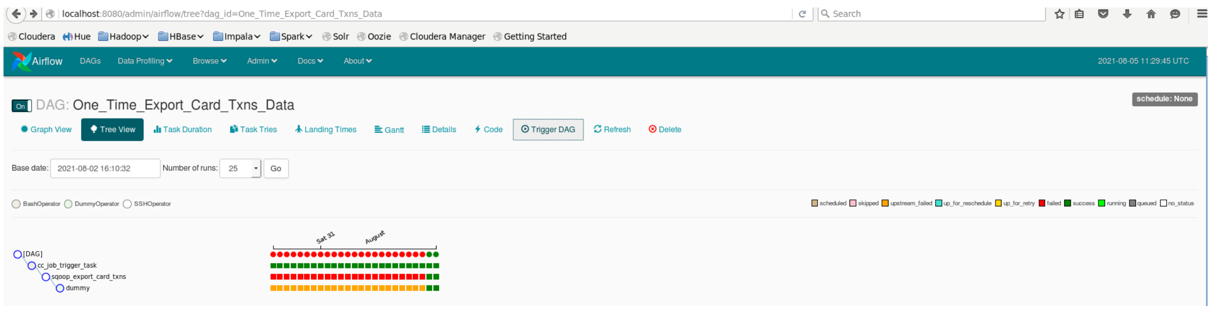
Airflow DAGs Data Profiling Browse Admin Docs About 2021-08-05 11:26:18 UTC

DAGs

Search:

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	On	One_Time_Export_Card_Txns_Data	None	airflow	3	2021-08-02 16:10	2 28	
	Off	example_bash_operator	0 0 * * *	Airflow				
	Off	example_branch_dop_operator_v3	*/1 * * * *	Airflow				
	Off	example_branch_operator	@daily	Airflow				

and click on the Trigger dag option


localhost:8080/admin/airflow/tree?dag_id=One_Time_Export_Card_Txns_Data

Cloudera Hue Hadoop Impala Spark Solr Oozie Cloudera Manager Getting Started

Airflow DAGs Data Profiling Browse Admin Docs About 2021-08-05 11:29:45 UTC

On DAG: One_Time_Export_Card_Txns_Data

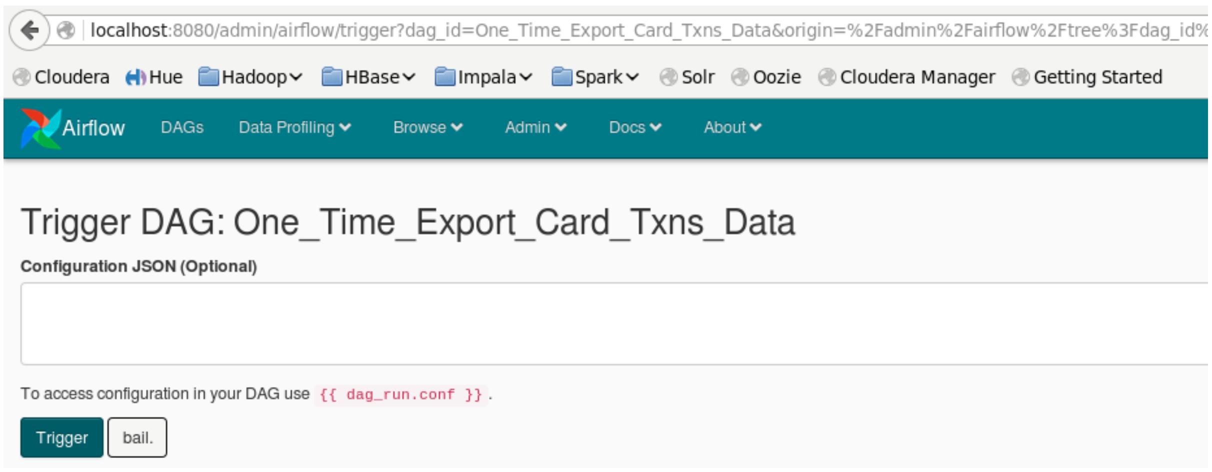
schedule: None

Graph View | Tree View | Task Duration | Task Tries | Landing Times | Gantt | Details | Code | Trigger DAG | Refresh | Delete

Base date: 2021-08-02 16:10:32 Number of runs: 25 Go

BashOperator DummyOperator SSHOperator

scheduled skipped upstream_failed up_for_reschedule up_for_retry failed success running queued no_status


localhost:8080/admin/airflow/trigger?dag_id=One_Time_Export_Card_Txns_Data&origin=%2Fadmin%2Fairflow%2Ftree%3Fdag_id%

Cloudera Hue Hadoop Impala Spark Solr Oozie Cloudera Manager Getting Started

Airflow DAGs Data Profiling Browse Admin Docs About

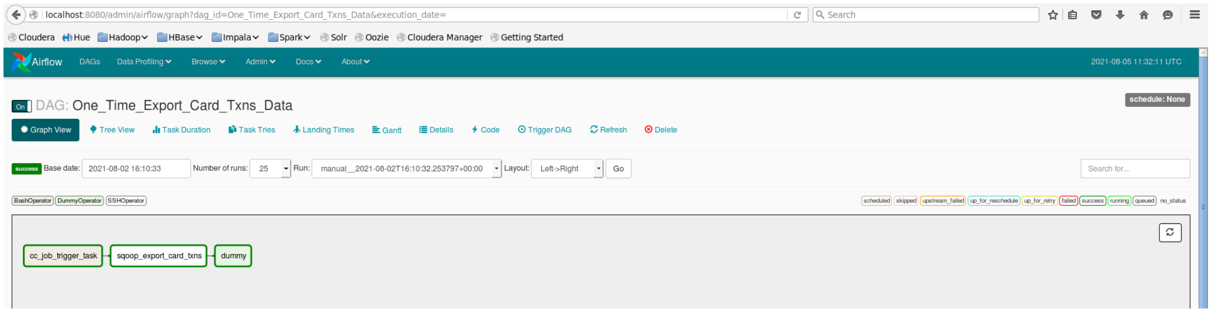
Trigger DAG: One_Time_Export_Card_Txns_Data

Configuration JSON (Optional)

To access configuration in your DAG use `{{ dag_run.conf }}`.

Trigger bail.

The graph view after the sqoop job has run successfully


localhost:8080/admin/airflow/graph?dag_id=One_Time_Export_Card_Txns_Data&execution_date=

Cloudera Hue Hadoop Impala Spark Solr Oozie Cloudera Manager Getting Started

Airflow DAGs Data Profiling Browse Admin Docs About 2021-08-05 11:32:11 UTC

On DAG: One_Time_Export_Card_Txns_Data

schedule: None

Graph View | Tree View | Task Duration | Task Tries | Landing Times | Gantt | Details | Code | Trigger DAG | Refresh | Delete

success Base date: 2021-08-02 16:10:33 Number of runs: 25 Run: manual__2021-08-02T16:10:32.253797+00:00 Layout: Left->Right Go Search for...

BashOperator DummyOperator SSHOperator

scheduled skipped upstream_failed up_for_reschedule up_for_retry failed success running queued no_status

[ERROR SOLUTIONS:

1.If you run into errors while triggering the DAG like -> no such file / directory
Run the following to get rid of any special characters in the script

```
cat sqoop_export_card_txns.sh | tr -d '\r' > a.sh.ne
```

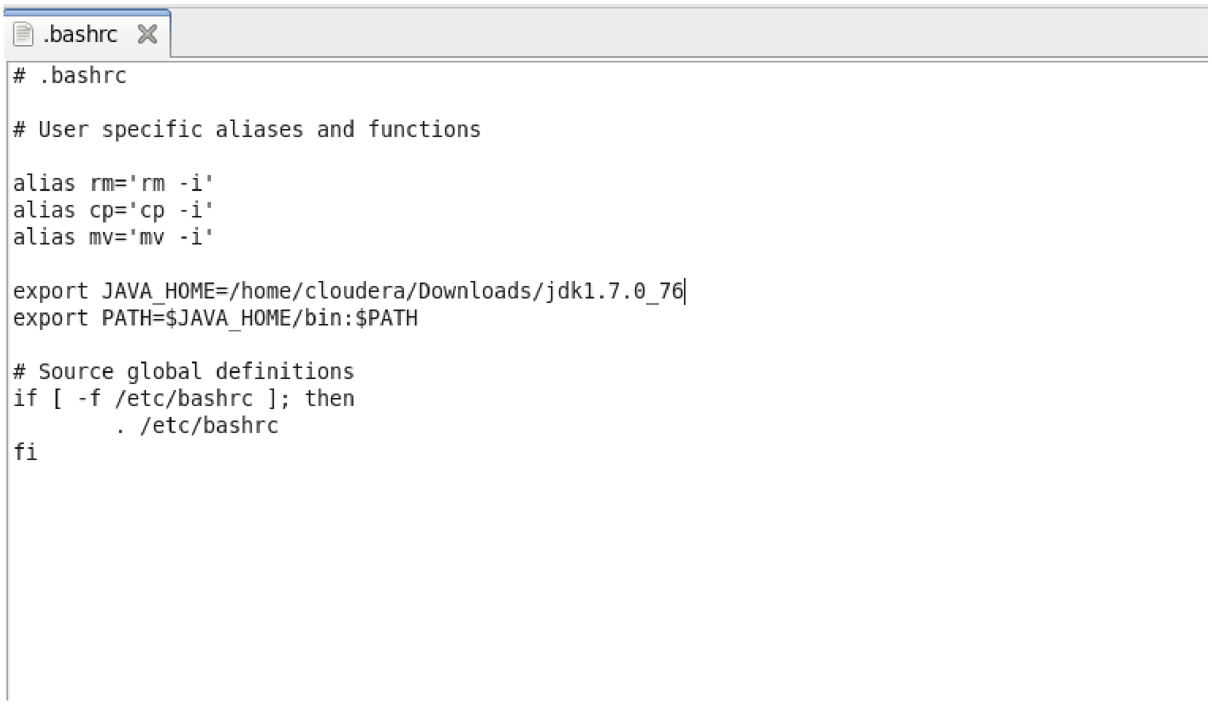
Then, rename a.sh.ne to sqoop_export_card_txns.sh back again:

```
mv a.sh.ne sqoop_export_card_txns.sh
```

2.If -> ERROR - SSH operator error: error running cmd: cd Desktop && ./sqoop_import_card_txns.sh quickstart.cloudera:3306 bigdataproject root card_transactions, error: 21/08/14 02:27:21 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0:

It is due to java version, so install java 7 from link: https://www.oracle.com/ae/java/technologies/javase/javase7-archive-downloads.html
and edit the bashrc file. Refer below ss:
-Execute below command in terminal to edit .bashrc file

```
gedit .bashrc
```



```
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

export JAVA_HOME=/home/cloudera/Downloads/jdk1.7.0_76
export PATH=$JAVA_HOME/bin:$PATH

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
```

-Execute below command to bring the changes to effect:

```
source .bashrc
```

And now check java version, it should be java 1.7
]
--Verify count (mysql terminal)
select count(*) from stg_card_transactions;

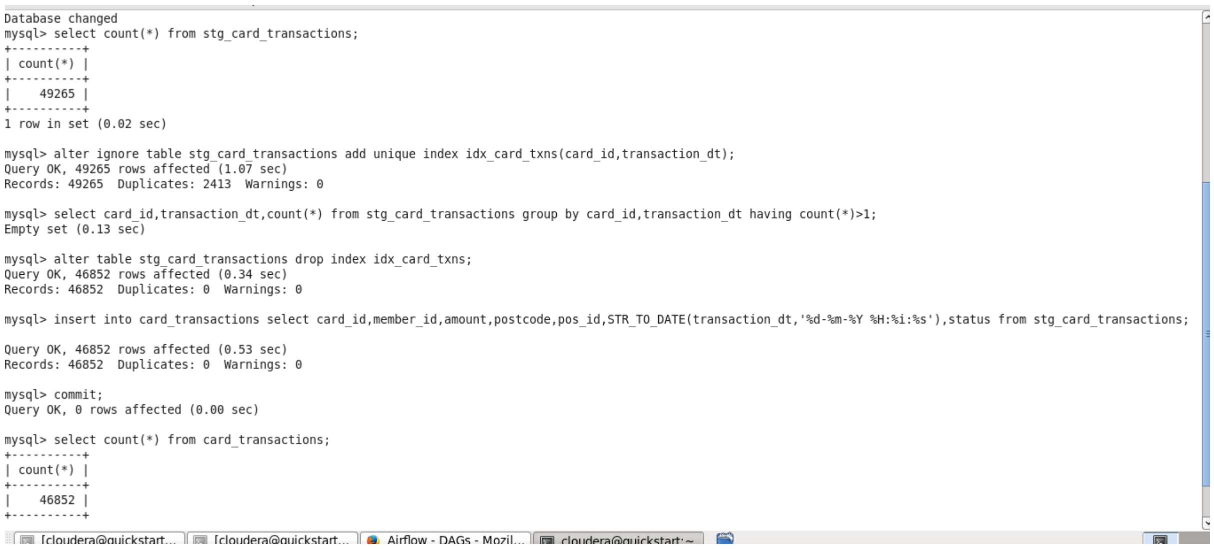
--Remove Dups from Stg Table (mysql terminal)
Alter ignore table stg_card_transactions
Add unique index idx_card_txns(card_id,transaction_dt);

--Verify no dups
select card_id,transaction_dt,count(*) from stg_card_transactions group by card_id,transaction_dt having count(*) >1;

--Dropping index used for removing dups
alter table stg_card_transactions drop index idx_card_txns;

--Loading main table
insert into card_transactions
select
card_id,member_id,amount,postcode,pos_id,STR_TO_DATE(transaction_dt,'%d-%m-%Y %H:%i:%s'),status from stg_card_transactions;
commit;

--Verify the count
select count(*) from card_transactions;



```
Database changed
mysql> select count(*) from stg_card_transactions;
+----------+
| count(*) |
+----------+
|    49265 |
+----------+
1 row in set (0.02 sec)

mysql> alter ignore table stg_card_transactions add unique index idx_card_txns(card_id,transaction_dt);
Query OK, 49265 rows affected (1.07 sec)
Records: 49265  Duplicates: 2413  Warnings: 0

mysql> select card_id,transaction_dt,count(*) from stg_card_transactions group by card_id,transaction_dt having count(*)>1;
Empty set (0.13 sec)

mysql> alter table stg_card_transactions drop index idx_card_txns;
Query OK, 46852 rows affected (0.34 sec)
Records: 46852  Duplicates: 0  Warnings: 0

mysql> insert into card_transactions select card_id,member_id,amount,postcode,pos_id,STR_TO_DATE(transaction_dt,'%d-%m-%Y %H:%i:%s'),status from stg_card_transactions;

Query OK, 46852 rows affected (0.53 sec)
Records: 46852  Duplicates: 0  Warnings: 0

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> select count(*) from card_transactions;
+----------+
| count(*) |
+----------+
|    46852 |
+----------+
```

--Deleting the card_transactions file from HDFS
hadoop fs -rm /user/cloudera/project_input_data/card_transactions.csv

–Create member_score and member_details directory in HDFS

hadoop fs -mkdir project_input_data/member_score
hadoop fs -mkdir project_input_data/member_details

Task4: Creating HIVE Tables (Run in hive terminal)

Connect to Hive using -> $hive
And enable bucketing using the following in the hive terminal-
SET HIVE.ENFORCE.BUCKETING=TRUE;

```
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/';

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/';
```

--Member score bucketed table(8 buckets)

```
create table if not exists member_score_bucketed
(
 member_id string,
 score float
)
CLUSTERED BY (member_id) into 8 buckets;
```

--Member details bucketed table(8 buckets)

```
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 8 buckets;
```

Hive-Hbase card_transactions table creation : External & Bucketed tables

```
create external table if not exists card_transactions (
card_id bigint,
member_id bigint,
amount float,
postcode int,
pos_id bigint,
transaction_dt timestamp,
status string
)
row format delimited fields terminated by ','
stored as textfile
location '/project_input_data/card_transactions/';

create table card_transactions_bucketed
(
cardid_txnts string,
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:card_id,trans_data:mem
ber_id,trans_data:amount,trans_data:postcode,trans_data:pos_id,trans_data:transaction_dt,
trans_data:Status") TBLPROPERTIES ("hbase.table.name" = "card_transactions");
```

Hive-Hbase card_lookup table creation : Bucketed tables

```
create table card_lookup
(
member_id bigint,
card_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:member_id,lkp_data:ucl,lkp
_data:score, lkp_data:last_txn_time,lkp_data:last_txn_zip")
TBLPROPERTIES ("hbase.table.name" = "card_lookup");
```

Task5: Inserting Data into optimised hive tables

Run member_score & member_details Airflow Job to load data from AWS to hive tables

Script for connecting to AWS RDS and load data to hive member score external table file path
[member_score.py present in the Big Data Project Demo -> airflow scripts folder in the google drive link]

```
--Inserting into member_score_bucketed table(Run from hive terminal)
insert into table member_score_bucketed;
select * from member_score;
```

Script for connecting to AWS RDS and load data to hive member details external table file path
[member_details.py present in the Big Data Project Demo -> airflow scripts folder in the google drive link]

```
--Inserting into member_details_bucketed table(Run from hive terminal)
insert into table member_details_bucketed select * from member_details;
```

OR

--Manually add memberscore.csv and memberdetails.csv to HDFS using the put command as shown below.

[**cardmember.csv and memberscore.csv file will be provided in the google drive link mentioned at the beginning of this document under Project Dataset folder**] hadoop fs -put Desktop/memberscore.csv project_input_data/member_score/

hadoop fs -put Desktop/cardmembers.csv project_input_data/member_details/

--Load the data from hdfs to hive table (In Hive terminal)
load data inpath 'project_input_data/member_score/memberscore.csv' overwrite into table member_score;
load data inpath 'project_input_data/member_details/cardmembers.csv' overwrite into table member_details;

--To check if the data is loaded
select count(*) from member_score;

select count(*) from member_details;

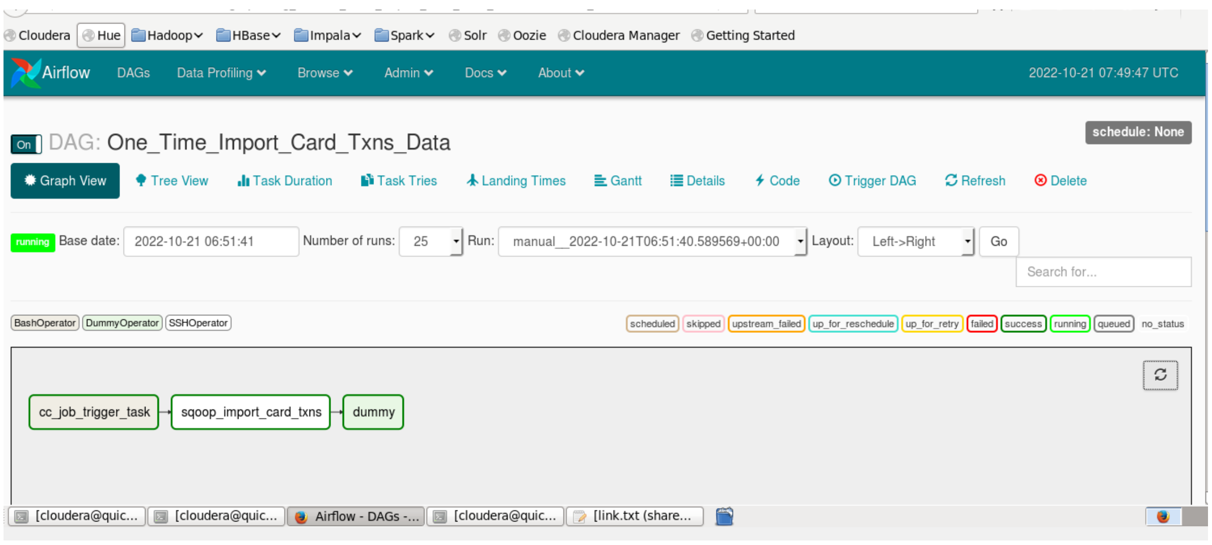
--Inserting data into bucketed tables
insert into table member_score_bucketed
select * from member_score;

insert into table member_details_bucketed
select * from member_details;

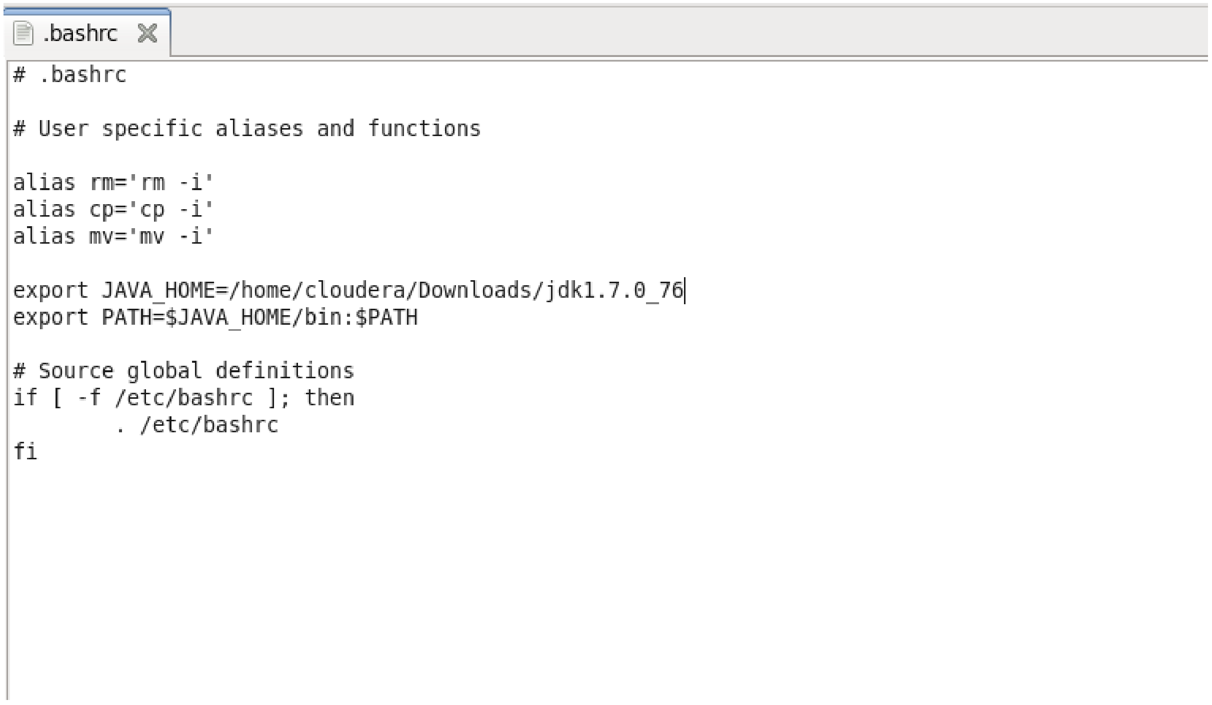
Airflow script for importing card transactions data to HDFS card_transactions external table file path from MySQL

-Run the import_card_txns.py (copy sqoop_import_card_txns.sh file to Desktop and give permission to this file: chmod 777 sqoop_import_card_txns.sh)

-**Setting up Variable for import**
click on **Variables:**
Under **create** tab, fill in the following details and save the connection
Key : card_txns_import_shell_command
Val : ./sqoop_import_card_txns.sh quickstart.cloudera:3306 bigdataproject root card_transactions



[**ERROR SOLUTIONS**:
if -> ERROR - SSH operator error: error running cmd: cd Desktop && ./sqoop_import_card_txns.sh quickstart.cloudera:3306 bigdataproject root card_transactions, error: 21/08/14 02:27:21 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6-cdh5.13.0
Download java 7 and edit .bashrc file



```
# .bashrc

# User specific aliases and functions

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

export JAVA_HOME=/home/cloudera/Downloads/jdk1.7.0_76
export PATH=$JAVA_HOME/bin:$PATH

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
```

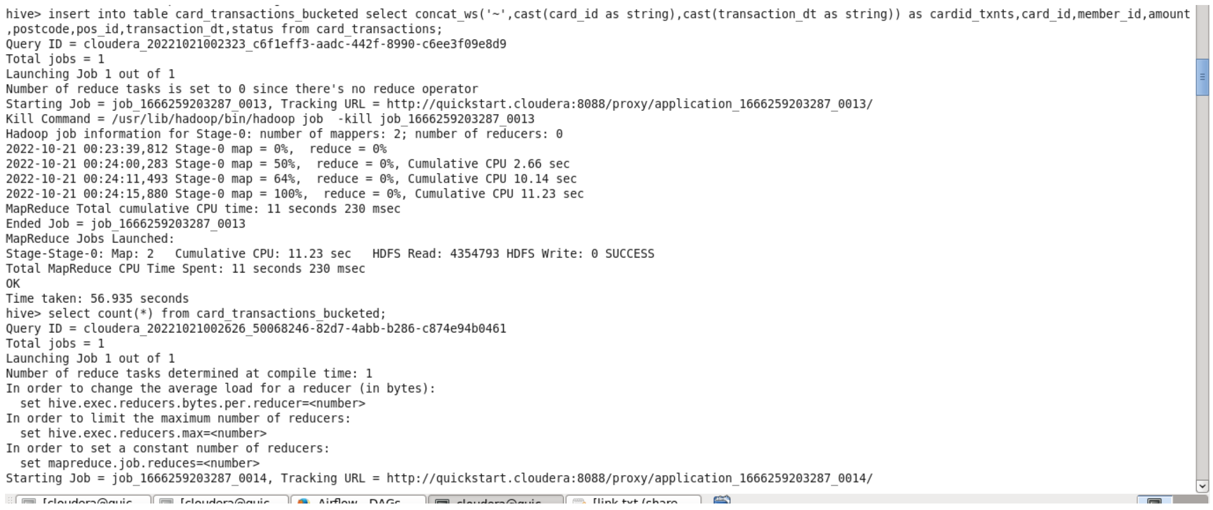
and execute below command in terminal:

```
source .bashrc
```

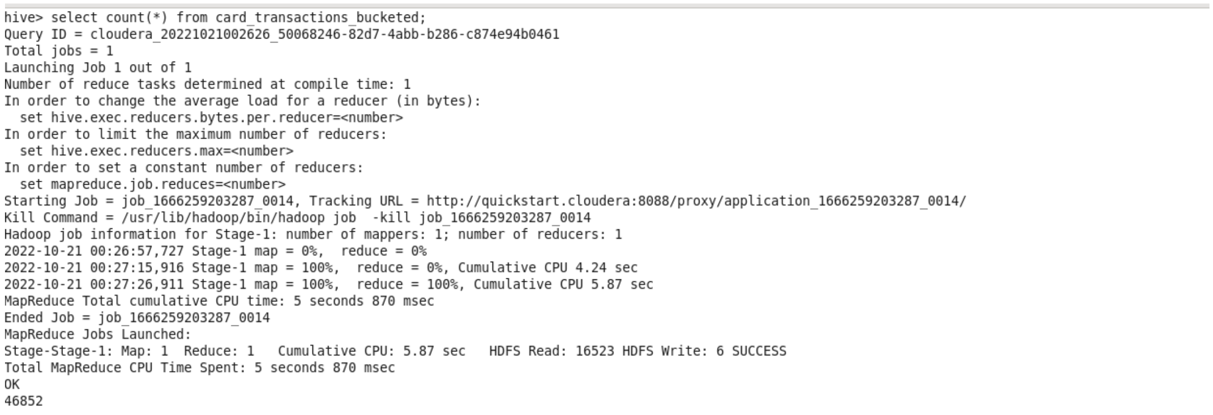
And now check java version, it should be java 1.7
]

--Load card_txns_bucketed table with concatenated row key (In the Hive terminal)
insert into table card_transactions_bucketed
select concat_ws('~',cast(card_id as string),cast(transaction_dt as string)) as cardid_txnts,card_id,member_id,amount,postcode,pos_id,transaction_dt,status
from card_transactions;

select count(*) from card_transactions_bucketed;



```
hive> insert into table card_transactions_bucketed select concat_ws('~',cast(card_id as string),cast(transaction_dt as string)) as cardid_txnts,card_id,member_id,amount
,postcode,pos_id,transaction_dt,status from card_transactions;
Query ID = cloudera_20221021002323_c6f1eff3-aadc-442f-8990-c6ee3f09e8d9
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1666259203287_0013, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1666259203287_0013/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1666259203287_0013
Hadoop job information for Stage-0: number of mappers: 2; number of reducers: 0
2022-10-21 00:23:39,812 Stage-0 map = 0%,  reduce = 0%
2022-10-21 00:24:00,283 Stage-0 map = 50%,  reduce = 0%, Cumulative CPU 2.66 sec
2022-10-21 00:24:11,493 Stage-0 map = 64%,  reduce = 0%, Cumulative CPU 10.14 sec
2022-10-21 00:24:15,880 Stage-0 map = 100%,  reduce = 0%, Cumulative CPU 11.23 sec
MapReduce Total cumulative CPU time: 11 seconds 230 msec
Ended Job = job_1666259203287_0013
MapReduce Jobs Launched:
Stage-Stage-0: Map: 2   Cumulative CPU: 11.23 sec   HDFS Read: 4354793 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 230 msec
OK
Time taken: 56.935 seconds
hive> select count(*) from card_transactions_bucketed;
Query ID = cloudera_20221021002626_50068246-82d7-4abb-b286-c874e94b0461
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1666259203287_0014, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1666259203287_0014/
```



```
hive> select count(*) from card_transactions_bucketed;
Query ID = cloudera_20221021002626_50068246-82d7-4abb-b286-c874e94b0461
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1666259203287_0014, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1666259203287_0014/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1666259203287_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2022-10-21 00:26:57,727 Stage-1 map = 0%,  reduce = 0%
2022-10-21 00:27:15,916 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.24 sec
2022-10-21 00:27:26,911 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.87 sec
MapReduce Total cumulative CPU time: 5 seconds 870 msec
Ended Job = job_1666259203287_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.87 sec   HDFS Read: 16523 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 870 msec
OK
46852
```

[<u>ERROR SOLUTIONS:</u>
Hbase connection refused error ->
Check if the Hbase server failed to start by
sudo service --status-all

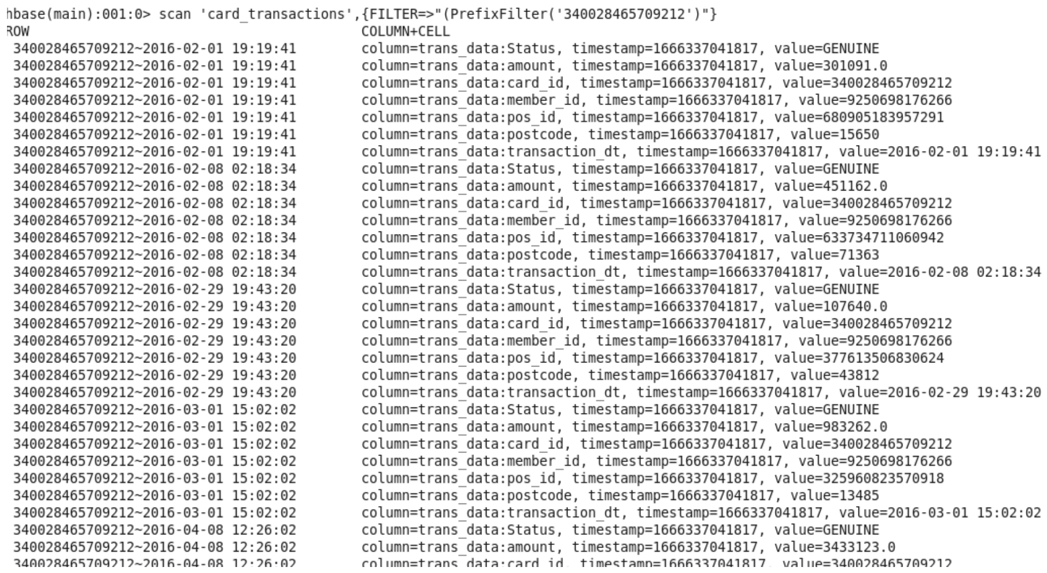
To restart the Hbase server
sudo service hbase-master start

sudo service hbase-regionserver start]

Task6: Batch Processing

--Hbase Search functionality based on rowkey

scan 'card_transactions', {FILTER => "(PrefixFilter('340028465709212')"}



```
hbase(main):001:0> scan 'card_transactions',{FILTER=>"(PrefixFilter('340028465709212')"}
ROW                                         COLUMN+CELL
 340028465709212~2016-02-01 19:19:41        column=trans_data:Status, timestamp=1666337041817, value=GENUINE
 340028465709212~2016-02-01 19:19:41        column=trans_data:amount, timestamp=1666337041817, value=301091.0
 340028465709212~2016-02-01 19:19:41        column=trans_data:card_id, timestamp=1666337041817, value=340028465709212
 340028465709212~2016-02-01 19:19:41        column=trans_data:member_id, timestamp=1666337041817, value=9250698176266
 340028465709212~2016-02-01 19:19:41        column=trans_data:pos_id, timestamp=1666337041817, value=680905183957291
 340028465709212~2016-02-01 19:19:41        column=trans_data:postcode, timestamp=1666337041817, value=15650
 340028465709212~2016-02-01 19:19:41        column=trans_data:transaction_dt, timestamp=1666337041817, value=2016-02-01 19:19:41
 340028465709212~2016-02-08 02:18:34        column=trans_data:Status, timestamp=1666337041817, value=GENUINE
 340028465709212~2016-02-08 02:18:34        column=trans_data:amount, timestamp=1666337041817, value=451162.0
 340028465709212~2016-02-08 02:18:34        column=trans_data:card_id, timestamp=1666337041817, value=340028465709212
 340028465709212~2016-02-08 02:18:34        column=trans_data:member_id, timestamp=1666337041817, value=9250698176266
 340028465709212~2016-02-08 02:18:34        column=trans_data:pos_id, timestamp=1666337041817, value=633734711060942
 340028465709212~2016-02-08 02:18:34        column=trans_data:postcode, timestamp=1666337041817, value=71363
 340028465709212~2016-02-08 02:18:34        column=trans_data:transaction_dt, timestamp=1666337041817, value=2016-02-08 02:18:34
 340028465709212~2016-02-29 19:43:20        column=trans_data:Status, timestamp=1666337041817, value=GENUINE
 340028465709212~2016-02-29 19:43:20        column=trans_data:amount, timestamp=1666337041817, value=107640.0
 340028465709212~2016-02-29 19:43:20        column=trans_data:card_id, timestamp=1666337041817, value=340028465709212
 340028465709212~2016-02-29 19:43:20        column=trans_data:member_id, timestamp=1666337041817, value=9250698176266
 340028465709212~2016-02-29 19:43:20        column=trans_data:pos_id, timestamp=1666337041817, value=377613506830624
 340028465709212~2016-02-29 19:43:20        column=trans_data:postcode, timestamp=1666337041817, value=43812
 340028465709212~2016-02-29 19:43:20        column=trans_data:transaction_dt, timestamp=1666337041817, value=2016-02-29 19:43:20
 340028465709212~2016-03-01 15:02:02        column=trans_data:Status, timestamp=1666337041817, value=GENUINE
 340028465709212~2016-03-01 15:02:02        column=trans_data:amount, timestamp=1666337041817, value=983262.0
 340028465709212~2016-03-01 15:02:02        column=trans_data:card_id, timestamp=1666337041817, value=340028465709212
 340028465709212~2016-03-01 15:02:02        column=trans_data:member_id, timestamp=1666337041817, value=9250698176266
 340028465709212~2016-03-01 15:02:02        column=trans_data:pos_id, timestamp=1666337041817, value=325960823570918
 340028465709212~2016-03-01 15:02:02        column=trans_data:postcode, timestamp=1666337041817, value=13485
 340028465709212~2016-03-01 15:02:02        column=trans_data:transaction_dt, timestamp=1666337041817, value=2016-03-01 15:02:02
 340028465709212~2016-04-08 12:26:02        column=trans_data:Status, timestamp=1666337041817, value=GENUINE
 340028465709212~2016-04-08 12:26:02        column=trans_data:amount, timestamp=1666337041817, value=3433123.0
 340028465709212~2016-04-08 12:26:02        column=trans_data:card_id, timestamp=1666337041817, value=340028465709212
```

-Setting up Scala IDE

In your .bashrc file have the below 2 lines

==

export JAVA_HOME=/home/cloudera/Downloads/jdk1.8.0_271
export PATH=$JAVA_HOME/bin:$PATH

Check java version using

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

java -version
it should show java 1.8

Download the scala ide

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

http://downloads.typesafe.com/scalaide-pack/4.7.0-vfinal-oxygen-212-20170929/scala-SDK-4.7.0-vfinal-2.12-linux.gtk.x86_64.tar.gz and then extract this.

In your eclipse.ini file put the below line at the end.

===

-Dosgi.bundles=org.eclipse.equinox.simpleconfigurator@1:start,org.eclipse.equinox.common@2:start,org.eclipse.equinox.ds@2:start,org.eclipse.equinox.event@2:start,org.eclipse.update.configurator@3:start,org.eclipse.core.runtime@start

Steps to start Scala IDE

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

cd /home/cloudera/Downloads/eclipse

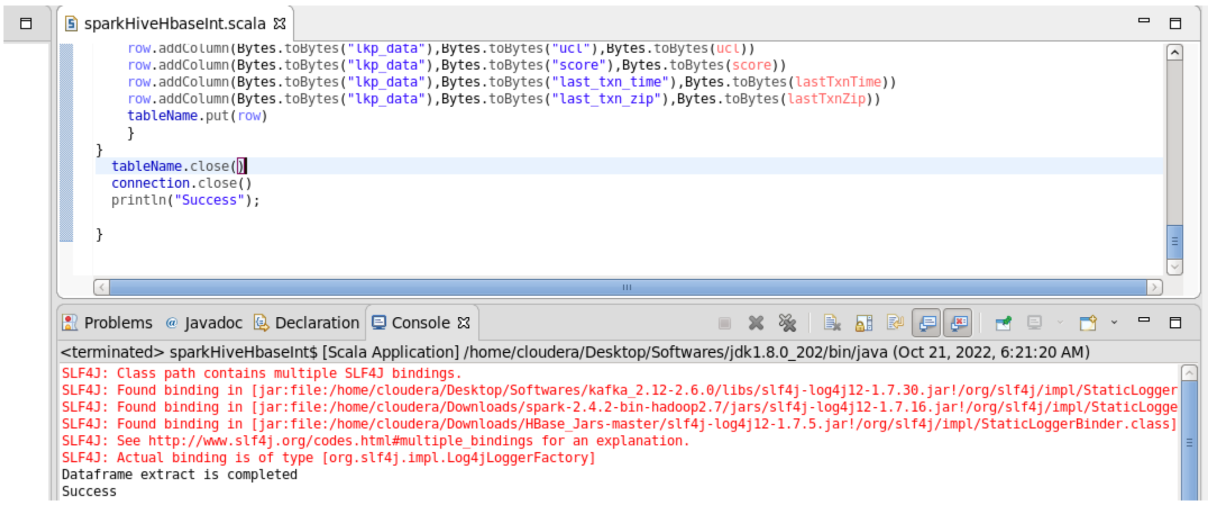
./eclipse -clean

Download spark 3.2.2 binaries in your cloudera using below link:
https://archive.apache.org/dist/spark/

Download Hbase Dependencies in your cloudera using below link:
http://codeload.github.com/gvreddy1210/Hbase_jars/zip/master

[if you download spark 3.2.2
you need to use scala 2.12 compiler]

-Create a new Scala Project (name:bigdataproject)
-Create new Scala Object(name:sparkHiveHbaseInt.scala) [Download the file "sparkHiveHbaseInt.scala" from Code and scripts>Batch processing>IntegrationCodes>sparkHiveHbaseInt]
-Add the required spark dependencies - "spark-3.2.2-bin-hadoop2.7" and "Hbase master". (link is given above)
-Run as scala application sparkHiveHbaseInt.scala optimized scala code from Eclipse

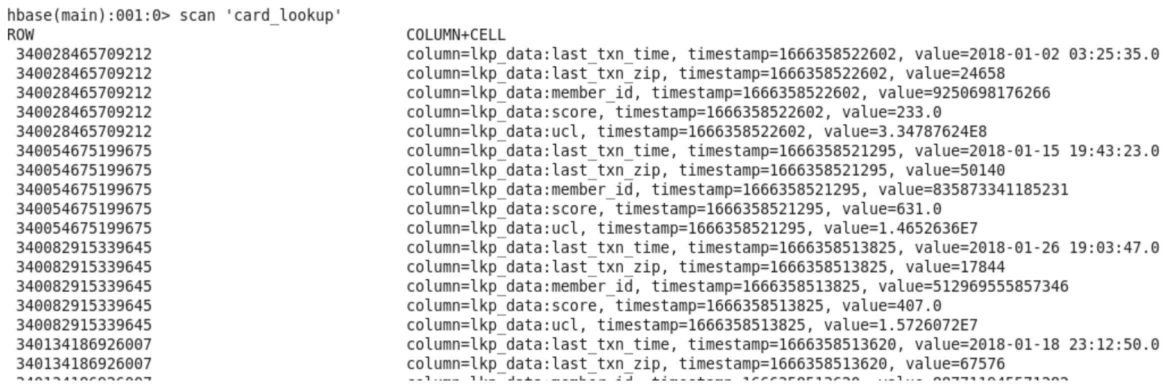


```
        row.addColumn(Bytes.toBytes("lkp_data"),Bytes.toBytes("ucl"),Bytes.toBytes(ucl))
        row.addColumn(Bytes.toBytes("lkp_data"),Bytes.toBytes("score"),Bytes.toBytes(score))
        row.addColumn(Bytes.toBytes("lkp_data"),Bytes.toBytes("last_txn_time"),Bytes.toBytes(lastTxnTime))
        row.addColumn(Bytes.toBytes("lkp_data"),Bytes.toBytes("last_txn_zip"),Bytes.toBytes(lastTxnZip))
        tableName.put(row)
        }
  }
    tableName.close()
    connection.close()
    println("Success");

  }
```

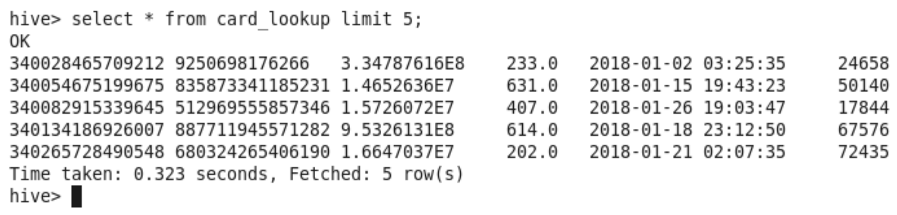
```
<terminated> sparkHiveHbaseInt$ [Scala Application] /home/cloudera/Desktop/Softwares/jdk1.8.0_202/bin/java (Oct 21, 2022, 6:21:20 AM)
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/cloudera/Desktop/Softwares/kafka_2.12-2.6.0/libs/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLogger
SLF4J: Found binding in [jar:file:/home/cloudera/Downloads/spark-2.4.2-bin-hadoop2.7/jars/slf4j-log4j12-1.7.16.jar!/org/slf4j/impl/StaticLogge
SLF4J: Found binding in [jar:file:/home/cloudera/Downloads/HBase_Jars-master/slf4j-log4j12-1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Dataframe extract is completed
Success
```

--Verify lookup table data from HBase
scan 'card_lookup'



```
hbase(main):001:0> scan 'card_lookup'
ROW                                              COLUMN+CELL
 340028465709212                                 column=lkp_data:last_txn_time, timestamp=1666358522602, value=2018-01-02 03:25:35.0
 340028465709212                                 column=lkp_data:last_txn_zip, timestamp=1666358522602, value=24658
 340028465709212                                 column=lkp_data:member_id, timestamp=1666358522602, value=9250698176266
 340028465709212                                 column=lkp_data:score, timestamp=1666358522602, value=233.0
 340028465709212                                 column=lkp_data:ucl, timestamp=1666358522602, value=3.34787624E8
 340054675199675                                 column=lkp_data:last_txn_time, timestamp=1666358521295, value=2018-01-15 19:43:23.0
 340054675199675                                 column=lkp_data:last_txn_zip, timestamp=1666358521295, value=50140
 340054675199675                                 column=lkp_data:member_id, timestamp=1666358521295, value=835873341185231
 340054675199675                                 column=lkp_data:score, timestamp=1666358521295, value=631.0
 340054675199675                                 column=lkp_data:ucl, timestamp=1666358521295, value=1.4652636E7
 340082915339645                                 column=lkp_data:last_txn_time, timestamp=1666358513825, value=2018-01-26 19:03:47.0
 340082915339645                                 column=lkp_data:last_txn_zip, timestamp=1666358513825, value=17844
 340082915339645                                 column=lkp_data:member_id, timestamp=1666358513825, value=512969555857346
 340082915339645                                 column=lkp_data:score, timestamp=1666358513825, value=407.0
 340082915339645                                 column=lkp_data:ucl, timestamp=1666358513825, value=1.5726072E7
 340134186926007                                 column=lkp_data:last_txn_time, timestamp=1666358513620, value=2018-01-18 23:12:50.0
 340134186926007                                 column=lkp_data:last_txn_zip, timestamp=1666358513620, value=67576
```

--Verify lookup table data from Hive
select * from card_lookup limit 5;



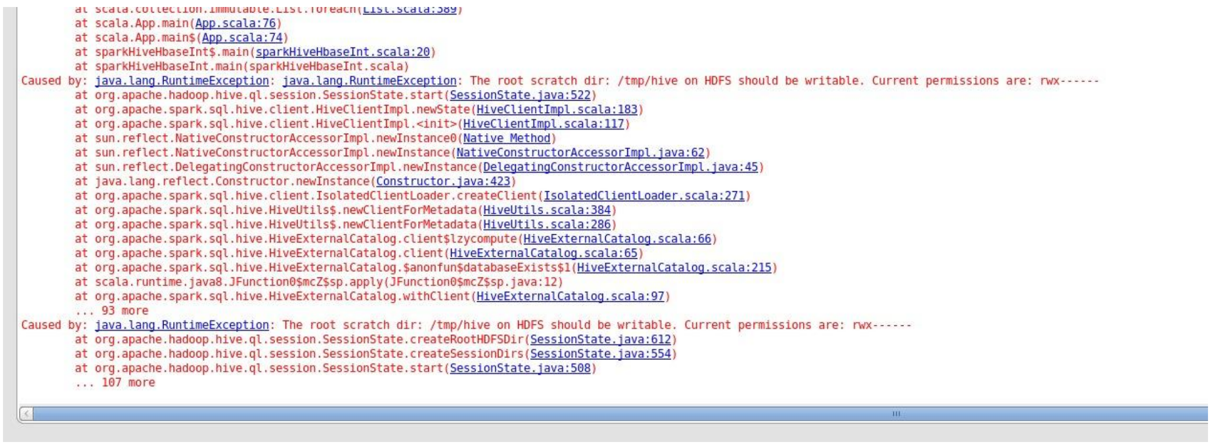
```
hive> select * from card_lookup limit 5;
OK
340028465709212 9250698176266   3.34787616E8    233.0   2018-01-02 03:25:35     24658
340054675199675 835873341185231 1.4652636E7     631.0   2018-01-15 19:43:23     50140
340082915339645 512969555857346 1.5726072E7     407.0   2018-01-26 19:03:47     17844
340134186926007 887711945571282 9.5326131E8     614.0   2018-01-18 23:12:50     67576
340265728490548 680324265406190 1.6647037E7     202.0   2018-01-21 02:07:35     72435
Time taken: 0.323 seconds, Fetched: 5 row(s)
hive> 
```

Or

You can also trigger the same job through AIRFLOW as well
Airflow script to RUN the above JAR, batch_job
“cc_batch_job.py”
(Downloadable from drive) -> copy this file to the DAGS folder of Airflow and trigger the DAG once it is listed in the DAGS list

[Errors & Solutions

If error is - The root scratch dir: /tmp/hive on HDFS should be writable.



```
        at scala.collection.immutable.List.foreach(List.scala:389)
        at scala.App.main(App.scala:76)
        at scala.App.main$(App.scala:74)
        at sparkHiveHbaseInt$.main(sparkHiveHbaseInt.scala:20)
        at sparkHiveHbaseInt.main(sparkHiveHbaseInt.scala)
Caused by: java.lang.RuntimeException: java.lang.RuntimeException: The root scratch dir: /tmp/hive on HDFS should be writable. Current permissions are: rwx------
        at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:522)
        at org.apache.spark.sql.hive.client.HiveClientImpl.newState(HiveClientImpl.scala:183)
        at org.apache.spark.sql.hive.client.HiveClientImpl.<init>(HiveClientImpl.scala:117)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
        at org.apache.spark.sql.hive.client.IsolatedClientLoader.createClient(IsolatedClientLoader.scala:271)
        at org.apache.spark.sql.hive.HiveUtils$.newClientForMetadata(HiveUtils.scala:384)
        at org.apache.spark.sql.hive.HiveUtils$.newClientForMetadata(HiveUtils.scala:286)
        at org.apache.spark.sql.hive.HiveExternalCatalog.client$lzycompute(HiveExternalCatalog.scala:66)
        at org.apache.spark.sql.hive.HiveExternalCatalog.client(HiveExternalCatalog.scala:65)
        at org.apache.spark.sql.hive.HiveExternalCatalog.$anonfun$databaseExists$1(HiveExternalCatalog.scala:215)
        at scala.runtime.java8.JFunction0$mcZ$sp.apply(JFunction0$mcZ$sp.java:12)
        at org.apache.spark.sql.hive.HiveExternalCatalog.withClient(HiveExternalCatalog.scala:97)
        ... 93 more
Caused by: java.lang.RuntimeException: The root scratch dir: /tmp/hive on HDFS should be writable. Current permissions are: rwx------
        at org.apache.hadoop.hive.ql.session.SessionState.createRootHDFSDir(SessionState.java:612)
        at org.apache.hadoop.hive.ql.session.SessionState.createSessionDirs(SessionState.java:554)
        at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:508)
        ... 107 more
```

Solution - Rebuild the project by adding the spark binaries - spark-3.2.2-bin-hadoop2.7]

Task7: Stream Processing

-Download the file “classReadFromKafka.scala” (from Code and Scripts>Streaming>project_repo_streaming>streamjobClasses)
-Download the file “classCardValidation.scala” (from Code and Scripts>Streaming>project_repo_streaming>streamjobClasses)
-Create new scala project (name:bigdatastreaming)
-Add the above 2 scala files under src folder of the “bigdatastreaming”

Download spark 2.4.3 binaries in your cloudera using below link:
https://archive.apache.org/dist/spark/

Here, set the scala compiler: 2.11

-Add the below jars:

1. **“kafka-spark(dependencies)” folder (from Code and Scripts>Streaming>project_repo_streaming>kafka-spark(dependencies))**
2. **Add the external jars from the below link for Hbase: http://codeload.github.com/gvreddy1210/Hbase_jars/zip/master**
3. **Add distanceFindjar(from Code and Scripts>Streaming>project_repo_streaming>distanceFindjar)**

[Note: Download the zipCodePosId.csv from downloadables and keep it in your Desktop, and change the path of this csv file in the code as per your path in class CardValidation.scala code]

-Starting Kafka Producer and Consumer and creating the KafkaTopic

--Start Kafka
```
cd /home/cloudera/Desktop/Softwares/kafka_2.12-2.6.0/bin

./kafka-server-start.sh ../config/server.properties
```

--Create Topic(In a new terminal)
```
cd /home/cloudera/Desktop/Softwares/kafka_2.12-2.6.0/bin

./kafka-topics.sh --create --topic cctxnstopic --bootstrap-server localhost:9092 --partitions 1
```

[Note: Use the topic name created here itself in your code also]

--Kafka Producer
```
./kafka-console-producer.sh --broker-list localhost:9092 --topic cctxnstopic
```

--Sample records to pass

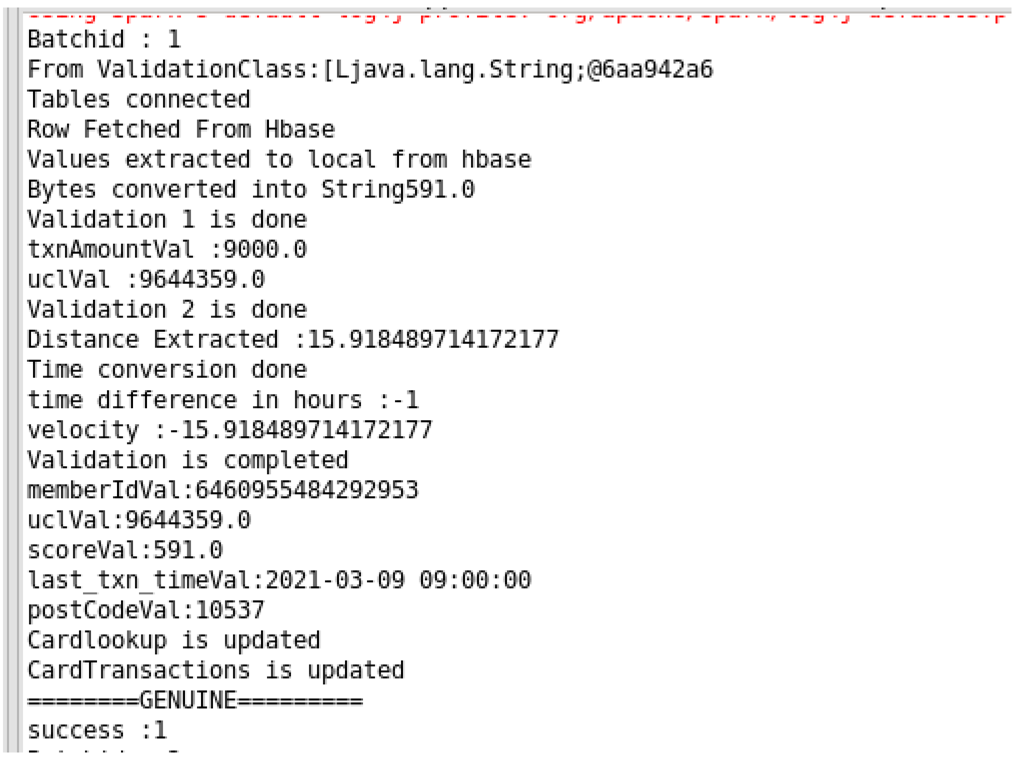
//genuine
{"card_id":555059812846420,"member_id":6460955484292953,"amount":9000,"pos_id":4444,"post_code":10537,"transc_dt":"2021-03-09 07:28:43"}

//fraud
{"card_id":555059812846420,"member_id":6460955484292953,"amount":9000000000,"pos_id":4444,"post_code":10537,"transc_dt":"2021-03-09 07:28:43"}

Note : the amount and transac_dt has to be changed to test for Fraud transactions.

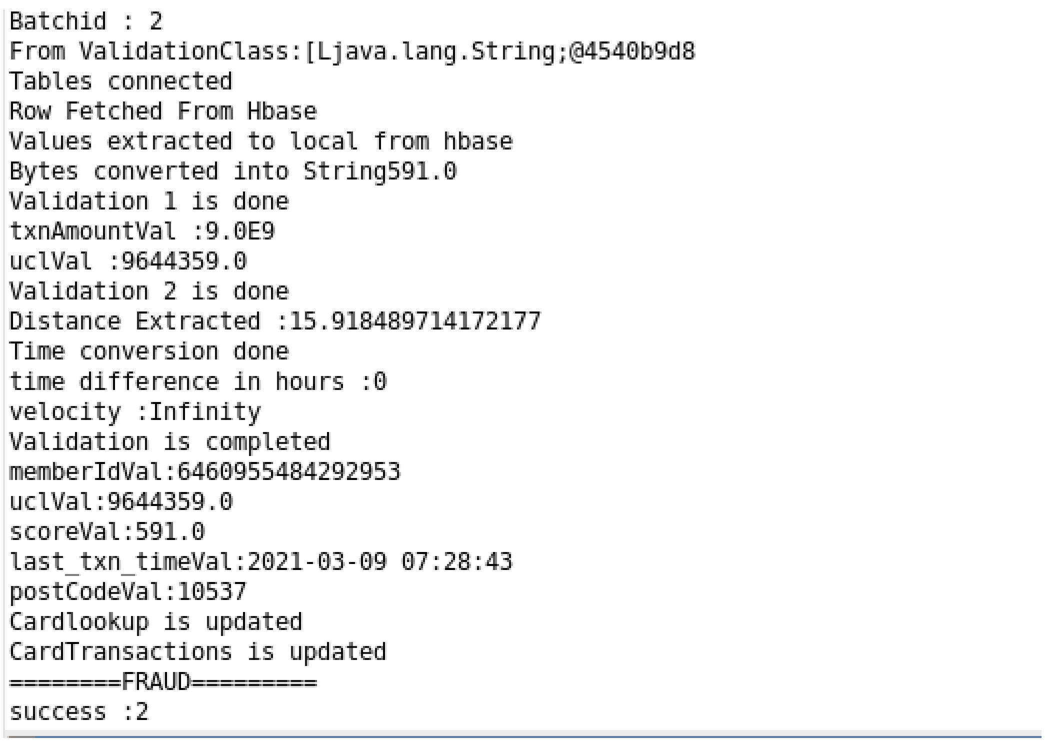
-Run as scala application classReadFromKafka.scala optimised scala code from Eclipse

Output after producing genuine transaction:



```
Batchid : 1
From ValidationClass:[Ljava.lang.String;@6aa942a6
Tables connected
Row Fetched From Hbase
Values extracted to local from hbase
Bytes converted into String591.0
Validation 1 is done
txnAmountVal :9000.0
uclVal :9644359.0
Validation 2 is done
Distance Extracted :15.918489714172177
Time conversion done
time difference in hours :-1
velocity :-15.918489714172177
Validation is completed
memberIdVal:6460955484292953
uclVal:9644359.0
scoreVal:591.0
last_txn_timeVal:2021-03-09 09:00:00
postCodeVal:10537
Cardlookup is updated
CardTransactions is updated
========GENUINE=========
success :1
```

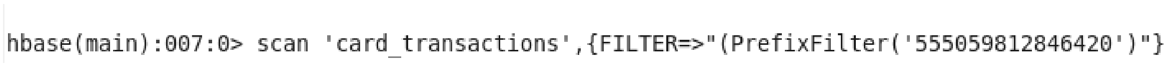
Output after producing fraud transaction:



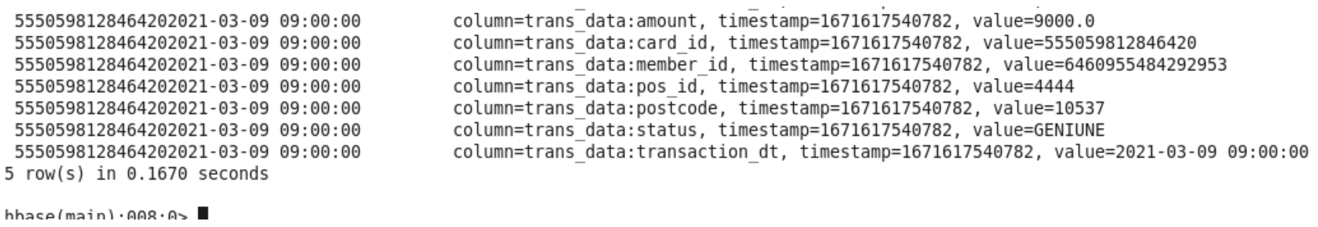
```
Batchid : 2
From ValidationClass:[Ljava.lang.String;@4540b9d8
Tables connected
Row Fetched From Hbase
Values extracted to local from hbase
Bytes converted into String591.0
Validation 1 is done
txnAmountVal :9.0E9
uclVal :9644359.0
Validation 2 is done
Distance Extracted :15.918489714172177
Time conversion done
time difference in hours :0
velocity :Infinity
Validation is completed
memberIdVal:6460955484292953
uclVal:9644359.0
scoreVal:591.0
last_txn_timeVal:2021-03-09 07:28:43
postCodeVal:10537
Cardlookup is updated
CardTransactions is updated
========FRAUD=========
success :2
```

Verification in card_transaction table to check if its updated:

Run following command in Hbase shell:

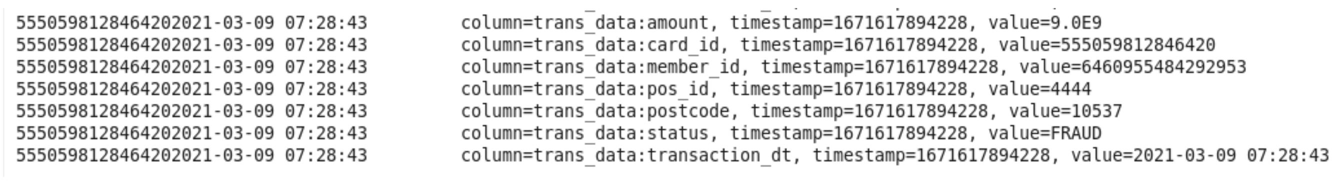


```
hbase(main):007:0> scan 'card_transactions',{FILTER=>"(PrefixFilter('555059812846420')"}
```



```
 5550598128464202021-03-09 09:00:00        column=trans_data:amount, timestamp=1671617540782, value=9000.0
 5550598128464202021-03-09 09:00:00        column=trans_data:card_id, timestamp=1671617540782, value=555059812846420
 5550598128464202021-03-09 09:00:00        column=trans_data:member_id, timestamp=1671617540782, value=6460955484292953
 5550598128464202021-03-09 09:00:00        column=trans_data:pos_id, timestamp=1671617540782, value=4444
 5550598128464202021-03-09 09:00:00        column=trans_data:postcode, timestamp=1671617540782, value=10537
 5550598128464202021-03-09 09:00:00        column=trans_data:status, timestamp=1671617540782, value=GENIUNE
 5550598128464202021-03-09 09:00:00        column=trans_data:transaction_dt, timestamp=1671617540782, value=2021-03-09 09:00:00
5 row(s) in 0.1670 seconds

hbase(main):008:0>
```



```
 5550598128464202021-03-09 07:28:43        column=trans_data:amount, timestamp=1671617894228, value=9.0E9
 5550598128464202021-03-09 07:28:43        column=trans_data:card_id, timestamp=1671617894228, value=555059812846420
 5550598128464202021-03-09 07:28:43        column=trans_data:member_id, timestamp=1671617894228, value=6460955484292953
 5550598128464202021-03-09 07:28:43        column=trans_data:pos_id, timestamp=1671617894228, value=4444
 5550598128464202021-03-09 07:28:43        column=trans_data:postcode, timestamp=1671617894228, value=10537
 5550598128464202021-03-09 07:28:43        column=trans_data:status, timestamp=1671617894228, value=FRAUD
 5550598128464202021-03-09 07:28:43        column=trans_data:transaction_dt, timestamp=1671617894228, value=2021-03-09 07:28:43
```

card_transactions table is updated!!

==