Capstone Project
Logicmid
(Credit Card Fraud Detection System)

Devashish Kapadia

Content

DataIngestion

sqoop-job ingest from AWS RDS to HADOOP

--Incremental-card_member-job-Method followed.

```
sqoop job --create incremental_card_member --meta-connect
"jdbc:mysql://localhost/sqoop?user=sqoop&password=sqoop" -- import --connect
jdbc:mysql://upgradawsrds.cpclxrkdv wmz.us-east-
1.rds.amazonaws.com/cred_financials_data --username upgraduser --password upgraduser --table
card_member --incremental append --check-column member_joining_dt --last-value "1970-01-01
00:00:00" --warehouse-dir /user/sqoop_import/capstone_project
```

--member_score-job-Method followed.

```
sqoop job --create cred_member_score \
--meta-connect "jdbc:mysql://localhost/sqoop?user=sqoop&password=sqoop" \
-- import \
--connect jdbc:mysql://upgradawsrds.cpclxrkdv wmz.us-east-
1.rds.amazonaws.com/cred_financials_data \
--username upgraduser \
--password upgraduser \
--table member_score \
--warehouse-dir /user/sqoop_import/capstone_project
```

--Explanation

"sqoop job --create":- Sqoop job creates and saves the import and export commands

"meta-connect":- The Sqoop metastore is not a secure resource. Multiple users can access its contents. For this reason, Sqoop does not store passwords in the metastore. If you create a job that requires a password, you will be prompted for that password each time you execute the job. So this helps in creating job which will not prompt for password.

"import & connect" :- are used to tell sqoop it is an import command and connect is to tell sqoop to which type of system and database eg.ORACLE or MYSQL etc.

"username & password":- it is for authenticating the connection request.

"table":- this is to specify which table to import from the connected database.

"incremental & check-column & last-value":- Incremental imports are performed by comparing the values in a *check column* against a reference value for the most recent import

"warehouse-dir":- By default, Sqoop will import a table named foo to a directory named foo inside your home directory in HDFS. For example, if your username is someuser, then the import tool will write to /user/someuser/foo/(files). You can adjust the parent directory of the import with the --warehouse-dir argument

--Execute the job once to get the initial data load

sqoop job --exec incremental_card_member.:-

sqoop job --exec cred_member_score:-

"sqoop job --exec":-this command to execute the sqoop job

LoadCreateNoSQL.

--Hbase shell command for table creation

--Method Followed.

```
create 'card_transactions','member_detail','transaction_detail'
create 'lookup','limit','status'
create 'card_member','card_detail','location'
```

"create 'card_transactions','member_detail','transaction_detail'":- To create table in HBASE , first is table name "table name ", "column family"

--Create mapping table in hive for HBase card_transaction table

--Method Followed.

```
Create external table card_transations_hbase(key struct<member_id:bigint, transaction_dt:string,
amount:bigint>,
card_id bigint, post_code bigint,
pos_id bigint, status string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,member_detail:card_id,transaction_detail:post_code,transaction_det
ail:pos_id,transaction_detail:status")
tblproperties("hbase.table.name"="card_transactions");
```

--Explanation

created a hive and hbase integrated table to perform analytical queries.

--**Card transaction staging table in hive**
--Method-Followed.

```
create table if not exists transations_load (
card_id bigint,
member_id bigint,
amount bigint,
post_code bigint,
pos_id bigint,
transaction_dt string,
status string) row format delimited fields terminated by ','
tblproperties ("skip.header.line.count"="1");
```

--Explanation

created a staging table to load the data to hive and HBASE integrated table.

-- Load data from local path into staging table

--Method Followed

load data local inpath
'/home/cloudera/Desktop/card_transactions.csv'
overwrite into table transations_load ;

--Explanation

loading data from local file system using "LOCAL INPATH" and "OVERWRITE INTO TABLE" this is to overwrite the into the table if any prior data exist in the table.

-- Insert card transactions which has valid card id

--Method Followed

INSERT INTO TABLE card_transations_hbase
SELECT NAMED_STRUCT('member_id', tl.member_id,'transaction_dt', tl.transaction_dt, 'amount', tl.amount),
tl.card_id,post_code,pos_id,status FROM transations_load tl;

--Explanation

Inserting data into table card_transations_hbase by creating key as a STRUCT('member_id', tl.member_id,'transaction_dt', tl.transaction_dt, 'amount', tl.amount) with column families vales as (tl.card_id,post_code,pos_id,status) .

--droping staging table

drop table transations_load;

PreAnalysis

--Create mapping table in hive for Hbase card_member table

--Method followed

```
create external table card_member_hbase(card_id bigint,member_id bigint,joining_dt
string,purchase_dt string,country string,city string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
('hbase.columns.mapping'=':key,card_detail:member_id,card_detail:joining_dt,card_detail:purchase_dt ,l
ocation:country,location:city')
tblproperties('hbase.table.name'='card_member');
```

--Explanation

creating hive and hbase integrated table with card_members. the data in it contains card_ id as key and rest as column family.

-- Card member staging table in hive

--Method followed

```
create external table if not exists card_member_load (
card_id bigint,
member_id bigint,
joining_dt string,
purchase_dt string,
country string,
city string) row format delimited fields terminated by "," location
'/user/sqoop_import/capstone_project/card_member';
```

--Explanation

created a staging table to ingest the data in the staging table.

-- Insert data into HBase mapping table

--Method-followed

```
insert into table card_member_hbase select * from card_member_load;
```

--Explanation

loaded the full data from the staging table card_member_hbase

-- Create lookup UCL staging table

--Method followed

```
create table lookup_ucl (
card_id bigint,
member_id bigint,
ucl bigint);
```

--Explanation

created a staging table to store the values for lookup table

-- Cal UCL and insert into lookup UCL staging table
--Method followed
```
insert into table lookup_ucl
select cid, mid, (AVG(amt) + (3 * STDDEV_POP(amt))) as ucl from
(select card_id as cid, key.member_id as mid,key.amount as amt,
row_number() OVER (PARTITION BY card_id order by UNIX_TIMESTAMP(key.transaction_dt, 'dd-MM-yyyy
HH:mm:ss') desc) as rank
from card_transations_hbase where status = "GENUINE")a
where rank <= 10 group by cid, mid;
```

--Explanation

cid=> card_id, mid=> member_id and amt => amount.

Inserted the values directly in the staging table.

UCL calculated on the formula :- UCL=(Moving Average)+3×(Standard Deviation). first selected the **card_id**, **key.member_id** and **key.amount** from the **card_transations_hbase (card_id** from column family ,**key.member_id and key.amount** from the key in the strut data type)**.** To find UCL first we had to find last 10 transaction so for that we used **row_number() with PARTITION BY card_id**, **order by Transaction_date** (which had to be converted to UNIX_TIMESTAMP). to implement the formula took the average (**avg**) of the amount and Standard deviation (**STDDEV_POP)** by of last 10 transaction marked as genuine. and grouped the records by card_ id and member_id.

-- Create lookup full staging table for UCL & memscore
--Method followed
```
create table lookup_full (
card_id bigint,
member_id bigint,
ucl bigint,
score bigint);
```

--Explanation

created the **lookup_full** staging table to insert the joined value of the **member score** table and the **lookup_ucl** before creating the full **lookup table.**

-- Insert UCL and Score into lookup full staging table
--Method followed
```
insert into table lookup_full
select u.card_id, u.member_id, u.ucl, m.score from lookup_ucl u
LEFT OUTER JOIN member_score m
ON (u.member_id = m.member_id);
```

--Explanation

inserted the values directly by joining the tables lookup_ucl, member_score with the help of **left outer join.**

-- Create mapping table for HBase lookup table for loading limit attributes

--Method followed

```
create external table lookup_limit_hbase(card_id bigint,
ucl bigint, score bigint)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,limit:ucl,limit:score")
tblproperties("hbase.table.name"="lookup");
```

--Explanation

created table hive and hbase integrated table lookup_limit_hbase mapped to the lookup table. To insert the values of **ucl** and **score** into the **limit** column family of the lookup table in hbase.

-- Insert UCL and Score into mapping lookup from staging table

--Method followed

```
insert into table lookup_limit_hbase
select card_id, ucl, score from lookup_full;
```

--Explanation

inserted the values in lookup_limit_hbase table created in the above step by selecting card_id , ucl and score from lookup_full table. here card_id is the key.

-- Create mapping table for HBase lookup table for loading status attributes

--Method followed

```
create external table lookup_status_hbase(cid bigint,
pc bigint, tdt string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,status:pc,status:tdt")
tblproperties("hbase.table.name"="lookup");
```

--Explanation

created table hive and hbase integrated table lookup_limit_hbase mapped to the lookup table.To insert the values of post_code(pc) and transaction_date(td) into the status column family of the lookup table in hbase.

-- Insert postal code and transaction date into mapping lookup from card_transations_hbase

--Method followed

```
insert into table lookup_status_hbase
select card_id , post_code, td from
(select card_id ,key.transaction_dt as td, post_code,
row_number() OVER (PARTITION BY card_id order by UNIX_TIMESTAMP(key.transaction_dt, 'dd-MM-yyyy
HH:mm:ss') desc) as rank
from card_transations_hbase where status = 'GENUINE') a
where rank == 1;
```

--Explanation

Inserted the Transaction_date and -post_codes in the above table created here i have used the rank method to find the last transaction_date and post_code as the ucl value can only be associate with on transaction date. rest is the same method as above in lookup_limit_hbase table.

-- Create mapping table for Hbase lookup table

--Method followed

```
create external table lookup_full_hbase(card_id bigint,
ucl bigint, score bigint, post_code bigint, transaction_dt string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
with serdeproperties
("hbase.columns.mapping"=":key,limit:ucl,limit:score, status:pc, status:tdt")
tblproperties("hbase.table.name"="lookup");
```

--Explanation

Created the lookup_full_hbase with all the value (card_id ,
ucl , score , post_code and transaction_dt). all the values get updated in this table as it is mapped with the lookup table in hbase

Final step

-- Dropping all the staging tables

```
drop table lookup_ucl;
drop table lookup_full;
drop lookup_limit_hbase
drop lookup_limit_hbase
drop table lookup_status_hbase;
```