Hadoop and Spark Developer - CCA 175

Problem Scenario 5 [SQOOP]

CCA 175 Hadoop and Spark Developer Exam Preparation - Problem Scenario 5

PLEASE READ THE INTRODUCTION TO THIS SERIES. CLICK ON HOME LINK AND READ THE INTRO BEFORE ATTEMPTING TO SOLVE THE PROBLEMS

Video walkthrough of this problem is available at *[PART 1 CLICK HERE]* AND *[PART 2 CLICK HERE]*

Click here for the video version of this series. This takes you to the youtube playlist of videos.

Sqoop is one of the important topics for the exam. Based on generally reported exam pattern from anonymous internet bloggers, you can expect 2 out of 10 questions on this topic related to Data Ingest and Data Export using Sqoop. Hence, 20% of the exam score can be obtained just by practicing simple Sqoop concepts. Sqoop can be mastered easily (i.e in a few hours) at the skill level that CCA 175 exam is expecting you to demonstrate. I created this problem focusing on Sqoop alone, if you are able to perform this exercise on your own or at worst using just the sqoop user guide then there is a very very high chance that you can score the 20% easily.

Pre-Work: Please perform these steps before solving the problem

1. Login to MySQL using below commands on a fresh terminal window
   mysql -u retail_dba -p
   Password = cloudera
2. Create a replica product table and name it products_replica
   create table products_replica as select * from products
3. Add primary key to the newly created table
   alter table products_replica add primary key (product_id);
4. Add two more columns
   alter table products_replica add column (product_grade int, product_sentiment varchar(100))
5. Run below two update statements to modify the data
   update products_replica set product_grade = 1 where product_price > 500;
   update products_replica set product_sentiment = 'WEAK' where product_price between 300 and 500;

Problem 5: Above steps are important so please complete them successfully before attempting to solve the problem

1. Using sqoop, import products_replica table from MYSQL into hdfs such that fields are separated by a '|' and lines are separated by '\n'. Null values are represented as -1 for numbers and "NOT-AVAILABLE" for strings. Only records with product id greater than or equal to 1 and less than or equal to 1000 should be imported and use 3 mappers for importing. The destination file should be stored as a text file to directory **/user/cloudera/problem5/products-text**.
2. Using sqoop, import products_replica table from MYSQL into hdfs such that fields are separated by a '*' and lines are separated by '\n'. Null values are represented as -1000 for numbers and "NA" for strings. Only records with product id less than or equal to 1111 should be imported and use 2 mappers for importing. The destination file should be stored as a text file to directory **/user/cloudera/problem5/products-text-part1**.
3. Using sqoop, import products_replica table from MYSQL into hdfs such that fields are separated by a '*' and lines are separated by '\n'. Null values are represented as -1000 for numbers and "NA" for strings. Only records with product id greater than 1111 should be imported and use 5 mappers for importing. The destination file should be stored as a text file to directory **/user/cloudera/problem5/products-text-part2.**
4. Using sqoop merge data available in **/user/cloudera/problem5/products-text-part1** and **/user/cloudera/problem5/products-text-part2** to produce a new set of files in **/user/cloudera/problem5/products-text-both-parts**
5. Using sqoop do the following. Read the entire steps before you create the sqoop job.
   - create a sqoop job Import Products_replica table as text file to directory **/user/cloudera/problem5/products-incremental**. Import all the records.
   - insert three more records to Products_replica from mysql
   - run the sqoop job again so that only newly added records can be pulled from mysql
   - insert 2 more records to Products_replica from mysql
   - run the sqoop job again so that only newly added records can be pulled from mysql
   - Validate to make sure the records have not be duplicated in **HDFS**
6. Using sqoop do the following. Read the entire steps before you create the sqoop job.
   - create a hive table in database named **problem5** using below command
   - create table **products_hive** (product_id int, product_category_id int, product_name string, product_description string, product_price float, product_imaage string,product_grade int, product_sentiment string);
   - create a sqoop job Import Products_replica table as hive table to database named **problem5**. name the table as **products_hive**.
   - insert three more records to Products_replica from mysql
   - run the sqoop job again so that only newly added records can be pulled from mysql
   - insert 2 more records to Products_replica from mysql
   - run the sqoop job again so that only newly added records can be pulled from mysql
   - Validate to make sure the records have not been duplicated in **Hive** table
7. Using sqoop do the following. .
   - insert 2 more records into **products_hive** table using hive.
   - create table in mysql using below command
   - create table products_external (product_id int(11) primary Key, product_grade int(11), product_category_id int(11), product_name varchar(100), product_description varchar(100), product_price float, product_impage varchar(500), product_sentiment varchar(100));
   - export data from products_hive (hive) table to (mysql) products_external table.
   - insert 2 more records to Products_hive table from hive
   - export data from products_hive table to products_external table.
   - Validate to make sure the records have not be duplicated in **mysql** table

Solution:

Try your best to solve the above scenario without going through the solution below. If you could then use the solution to compare your result. If you could not then I strongly recommend that you go through the concepts again (this time in more depth). Each step below provides a solution to the points mentioned in the Problem Scenario.

Step 1:

Search This Blog

Search

CCA 175 Hadoop and Spark Developer Preparation

- Home
- CCA 175 Prep Plan
- Problem Scenario 1
- Problem Scenario 2
- Problem Scenario 3
- Problem Scenario 4
- **Problem Scenario 5 [SQOOP]**
- Problem Scenario 6 [Data Analysis]
- Problem Scenario 7 [FLUME]
- File Formats
- Youtube Playlist

A leader with a unique blend of deep technology expertise and strong management skil

Arun Kumar Pasuparthi

Follow 212

View my complete profile

Report Abuse

Blog Archive

April 2017 (1)

```
sqoop import \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
--username retail_dba \
--password cloudera \
--table products_replica \
--target-dir /user/cloudera/problem5/products-text \
--fields-terminated-by '|' \
--lines-terminated-by '\n' \
--null-non-string -1 \
--null-string "NOT-AVAILABLE" \
-m 3 \
--where "product_id between 1 and 1000" \
--outdir /home/cloudera/sqoop1 \
--boundary-query "select min(product_id), max(product_id) from products_replica where product_id between 1 and 1000";
```

Step 2:

```
sqoop import \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
--username retail_dba \
--password cloudera \
--table products_replica \
--target-dir  /user/cloudera/problem5/products-text-part1  \
--fields-terminated-by '*' \
--lines-terminated-by '\n' \
--null-non-string -1000 \
--null-string "NA" \
-m 2 \
--where "product_id <= 1111 " \
--outdir /home/cloudera/sqoop2 \
--boundary-query "select min(product_id), max(product_id) from products_replica where product_id <= 1111";
```

Step 3:

```
sqoop import \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
--username retail_dba \
--password cloudera \
--table products_replica \
--target-dir  /user/cloudera/problem5/products-text-part2  \
--fields-terminated-by '*' \
--lines-terminated-by '\n' \
--null-non-string -1000 \
--null-string "NA" \
-m 5 \
--where "product_id > 1111 " \
--outdir /home/cloudera/sqoop3 \
--boundary-query "select min(product_id), max(product_id) from products_replica where product_id > 1111"
```

Step 4:

```
sqoop merge \
--class-name products_replica \
--jar-file mp/sqoop-cloudera/compile/66b4f23796be7625138f2171a7331cd3/products_replica.jar \
--new-data /user/cloudera/problem5/products-text-part2 \
--onto /user/cloudera/problem5/products-text-part1 \
--target-dir /user/cloudera/problem5/products-text-both-parts \
--merge-key product_id;
```

Step 5:

On terminal -

```
sqoop job --create first_sqoop_job \
-- import \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
--username "retail_dba" \
--password "cloudera" \
--table products_replica \
--target-dir /user/cloudera/problem5/products-incremental \
--check-column product_id \
--incremental append \
--last-value 0;

sqoop job --exec first_sqoop_job
```

On MySQL command line -

```
mysql> insert into products_replica values (1346,2,'something 1','something 2',300.00,'not avaialble',3,'STRONG');
mysql> insert into products_replica values (1347,5,'something 787','something 2',356.00,'not avaialble',3,'STRONG');
```

On terminal -

```
sqoop job --exec first_sqoop_job
```

On MYSQL Command Line

```
insert into products_replica values (1376,4,'something 1376','something 2',1.00,'not avaialble',3,'WEAK');
insert into products_replica values (1365,4,'something 1376','something 2',10.00,'not avaialble',null,'NOT APPLICABLE');
```

On terminal -

```
sqoop job --exec first_sqoop_job
```

Step 6:

On Terminal window-

```
sqoop job \
--create hive_sqoop_job \
-- import \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
--username "retail_dba" \
--password "cloudera" \
--table products_replica  \
--check-column product_id \
```

```
--incremental append \
--last-value 0 \
--hive-import \
--hive-table products_hive \
--hive-database problem5;
```

On Hive window:

```
create database problem5;

use problem5;

create table products_hive  (product_id int, product_category_id int, product_name string, product_description string, product_price float, product_imaage string,product_grade int,  product_sentiment string);
```

On Terminal window

```
sqoop job --exec hive_sqoop_job
```

On MySQL window

```
insert into products_replica values (1378,4,'something 1376','something 2',10.00,'not avaialble',null,'NOT APPLICABLE');
insert into products_replica values (1379,4,'something 1376','something 2',10.00,'not avaialble',null,'NOT APPLICABLE');
```

On Terminal Window

```
sqoop job --exec hive_sqoop_job
```

On Hive Window

```
select * from products_hive
```

Step 7:

On Hive Window

```
use problem5;

insert into table products_hive values (1380,4,'something 1380','something 2',8.00,'not avaialble',3,'NOT APPLICABLE');

insert into table products_hive values (1381,4,'something 1380','something 2',8.00,'not avaialble',3,'NOT APPLICABLE');
```

On MYSQL window

```
create table products_external  (product_id int(11) primary Key, product_grade int(11), product_category_id int(11), product_name varchar(100), product_description varchar(100), product_price float, product_impage varchar(500), product_sentiment varchar(100));
```

On Terminal

```
sqoop export \
--username "retail_dba" \
--password "cloudera" \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
 --export-dir /user/hive/warehouse/problem5.db/products_hive/ \
--input-fields-terminated-by '\001' \
--input-null-non-string "null" \
--input-null-string "null" \
--update-mode allowinsert \
--update-key product_id \
--columns
"product_id,product_category_id,product_name,product_description,product_price,product_impage,product_grade,product_sentiment" --table products_external;
```

On Hive Window

```
insert into table products_hive values (1382,4,'something 1380','something 2',8.00,'not avaialble',3,'NOT APPLICABLE');

insert into table products_hive values (1383,4,'something 1380','something 2',8.00,'not avaialble',3,'NOT APPLICABLE');
```

On Terminal Window:

```
sqoop export \
--username "retail_dba" \
--password "cloudera" \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
--export-dir /user/hive/warehouse/problem5.db/products_hive/ \
--input-fields-terminated-by '\001' \
--input-null-non-string "null" \
--input-null-string "null" \
--update-mode allowinsert \
--update-key product_id \
--columns
"product_id,product_category_id,product_name,product_description,product_price,product_impage,product_grade,product_sentiment" --table products_external;
```

To Validate
On Hive

```
select count(*) from problem5.products_hive;
```

on MySQL

```
select count(*) from products_replica;
```

25 comments:

bloggie May 9, 2017 at 11:16 AM

Great job. These exercises gives confidence of Clearing the Certification

Reply

Deven May 23, 2017 at 1:02 PM

Excellent problem scenario!!! Please keep up the great work.
I came across a trivial point. In the 1st part of the problem it says "Only records with product id greater than 1 and less than 1000" - i think the where clause and boundary-query needs a slight change "product_id between 2 and 999"

Reply

Arun Kumar Pasuparthi May 23, 2017 at 1:57 PM

Thank you Daven for identifying an error. Given that I cannot update the video, i updated the question to suit the video. Thanks for the complement as well.

Reply

Unknown May 30, 2017 at 4:20 AM

why you have used boundary query.

Reply

Replies

Arun Kumar Pasuparthi June 4, 2017 at 5:14 PM

please identify yourself.

Reply

vishvas patel June 8, 2017 at 3:27 PM

Hey Arun can you please tell me why you used boundary query ?
How can I get the product_replica.jar file which you have mentioned here
--jar-file mp/sqoop-cloudera/compile/66b4f23796be7625138f2171a7331cd3/products_replica.jar
I tack hadoop MR job I found that it produced product_replica.jar file not able to access it.

Reply

hi June 8, 2017 at 3:42 PM

Look at the video buddy. Arun explained where to find the jar file

Reply

Deven June 17, 2017 at 1:20 PM

Arun Sir thanks for the awesome content It helped me clear the CCA175 certification. Thanks!!!

Reply

Sohail June 23, 2017 at 11:06 PM

Another way to do question 2 using --query, to avoid where and boundary-query

split-by is then mandatory.

```
sqoop import \
--connect "jdbc:mysql://quickstart.cloudera:3306/retail_db" \
--username retail_dba \
--password cloudera \
--query "select * from products_replica where product_id<=1111 and \$CONDITIONS" \
--target-dir /user/cloudera/problem5/products-text-part1 \
--fields-terminated-by '*' \
--lines-terminated-by '\n' \
--null-non-string -1000 \
--null-string "NA" \
--split-by product_id \
-m 2
```

Reply

Varun Mishra June 24, 2017 at 9:36 AM

Hi Arun,

In question 5, If every time we will incremental append by last value 0, It will create duplicate value in it.
Can you please explain?

last value should be change for each and every sqoop job.??

Reply

Arun Kumar Pasuparthi June 24, 2017 at 10:41 AM

you will have a better understanding if you can go through the video. to answer your question, Sqoop job remembers the last value and starts from where it left off. So you dont have to recreate the sqoop job with new sqoop import command everytime. Please watch video for better understanding. Good luck.

Reply

Varun Mishra June 24, 2017 at 10:44 AM

What is difference between input-fields-terminated-by and fields-terminated-by in sqoop export?

Reply

Arun Kumar Pasuparthi June 24, 2017 at 11:06 AM

input-fields-terminated by is used in sqoop export and describes the source files field terminator. .whereas fields-terminated-by is used in sqoop import and describes the destination files field terminator.

I corrected the typo in the solution. Thanks for bringing it to my notice. All the best for your exam.

Reply

Replies

Balaji K G September 1, 2017 at 12:15 AM

I used --fields-terminated-by option while exporting tab delimiter file. It's exported all the records to mysql table.

Can you please tell me the exact difference between these 2 options?

Reply

Murali Rachakonda July 19, 2017 at 7:47 PM

Hi Arun, TFGW (Thanks For Great Work). In the step 7 is there any specific reason why you mentioned columns in the export job --columns "product_id,product_category_id,product_name,product_description,product_price,product_impage,product_grade,product_sentiment" --table products_external;

Reply

Replies

Arun Kumar Pasuparthi July 19, 2017 at 8:02 PM

due to the difference in source and destination order. please go through the accompanying video. you can forward to that step if you dont want to watch the entire video.

Reply

Balaji K G September 1, 2017 at 12:11 AM

Today, I cleared CCA - 175 with 7/9..Thanks arun sir for giving such good examples.

Reply

Lakshmi Thiagarajan September 27, 2017 at 12:47 PM

Hi all
In the sqoop command for Step 1 ,

u either need the where condn

--where "product_id between 1 and 1000" \

or the boundary-query

--boundary-query "select min(product_id), max(product_id) from products_replica where product_id between 1 and 1000";

We dont need both . I tried with just the boundary-query and it worked as expected . Pls correct me if am wrong.

Reply

Replies

Lakshmi Thiagarajan September 27, 2017 at 12:58 PM

Again for Step 2, using boundary-query will suffice , no need of --where.

And for step 3 , we can just use --where "product_id > 1111 . Boundary query is mostly meant to be used when ur primary key values are skewed much apart from each other ..for eg , the primary key values range from 6000 to 8000 and then start from 11000 thru 15000 . Just a gentle suggestion.

Reply

Doubts Several November 8, 2017 at 3:30 AM

Hi Arun,
I've WATCHED YOUR YOUTUBE VIDEO but I still DON'T UNDERSTAND THIS part of 4c exercise. I'm totally lost with the combineByKeyResult:

combineByKey((x:(Float, String))=>(x._1,Set(x._2)),
(x:(Float,Set[String]),y:(Float,String))=>(x._1 + y._1,x._2+y._2),
(x:(Float,Set[String]),y:(Float,Set[String]))=>(x._1+y._1,x._2++y._2)).
map(x=> (x._1._1,x._1._2,x._2._1,x._2._2.size))

Completeley lost with the combineByKey. Could you please explain me what you are doing here?
I've got my exam in two days.

Thank you

Reply

Replies

GodfreyDeK November 19, 2017 at 4:37 AM

Hi @Doubts Several, you can have a look at this blog, explaining aggregateByKey:
http://codingjunkie.net/spark-agr-by-key/

Then have a look at the follow up post on combineByKey:
http://codingjunkie.net/spark-combine-by-key/

I hope it helps, i struggled too.

Doubts Several November 26, 2017 at 1:39 AM

Thank you!

Reply

GodfreyDeK November 19, 2017 at 5:22 AM

Greetings Arun,

I've used the jar location as shown in the video but i keep getting either of the following errors (depending on how i reorder the merge parameters):
" bash: --merge-key: command not found " or
" bash: --jar-file: command not found "

Is there any pre-work i should have done apart from what you've mentioned in the problem statement? Thank you.

Reply

Thanuja Sri December 9, 2017 at 7:16 AM

Really useful information about hadoop, i have to know information about hadoop online training institutes.

Reply

SuryaK January 1, 2018 at 7:13 AM

sqoop-import --connect jdbc:mysql://localhost/retail_db --username root --password cloudera --as-textfile -m 2 --query 'select * from products_replica where (product_id <=1111) AND $CONDITIONS' --split-by product_id --target-dir '/user/cloudera/problem5/products-text-part1' --null-string 'NA' --null-non-string '-1000' --fields-terminated-by '*' --lines-terminated-by '\n'

Arun cant we do problem 2 like above,Can you let me know if this is also a way to solce,and why is output-dir used?

Reply

Home

Subscribe to: Posts (Atom)

If you have landed on this page then you are most likely aspiring to learn Hadoop ecosystem of technologies and tools. Why not make you...

(no title)

If you have landed on this page then you are most likely aspiring to learn Hadoop ecosystem of technologies and tools. Why not make you...