ETL Tool for ADDB

연세대학교 인공지능학과 PIAO, SHENGMIN

2023년 12월

과제명: IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발

과제번호: 2017-0-00477

Context

- What is ETL?
- How ETL works
- The benefits and challenges of ETL
- ETL for ADDB

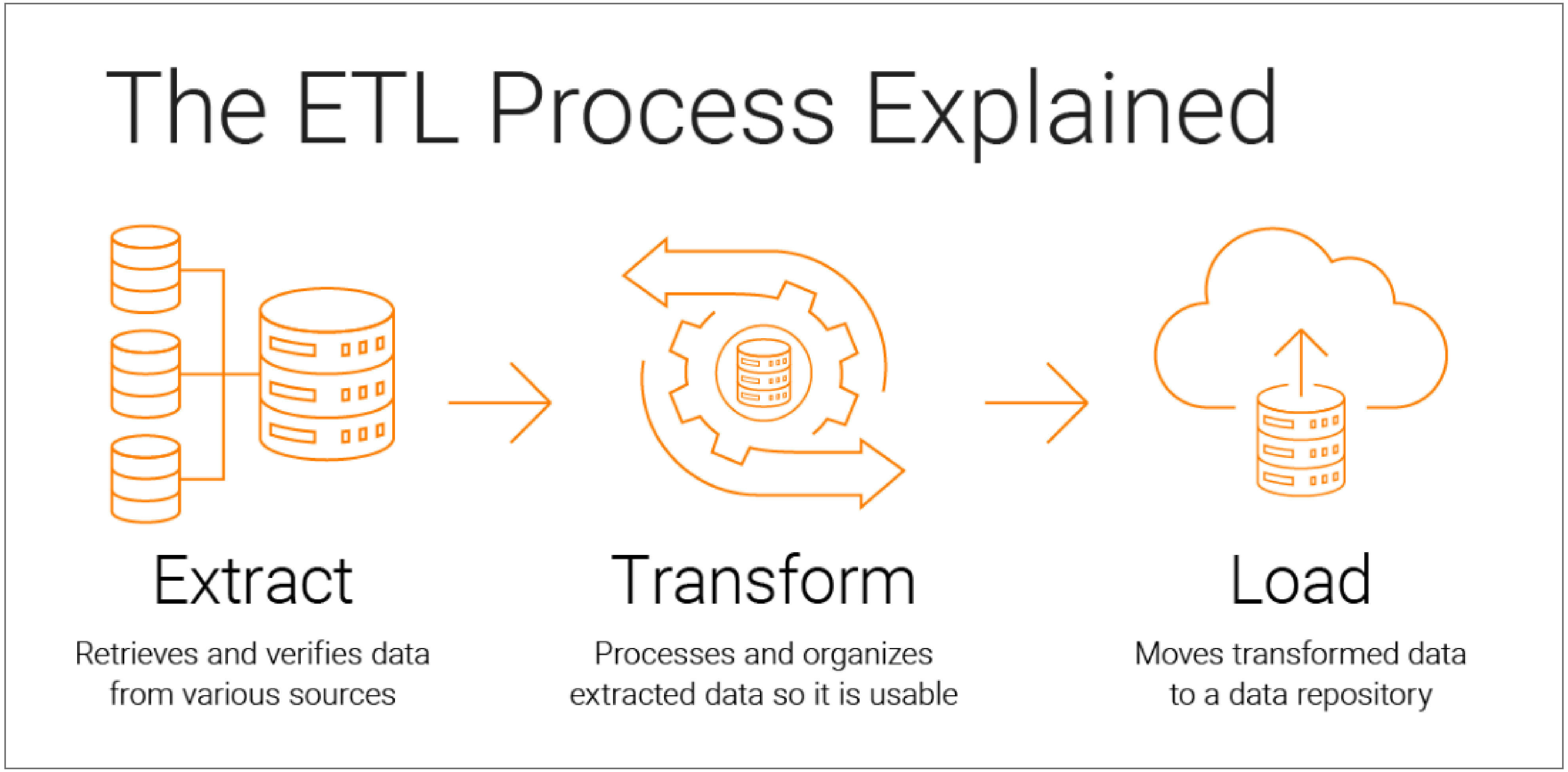
What is ETL

- ETL → **E**xtract, **T**ransform, **L**oad

- A data integration process that combines data from multiple data sources into a single, consistent data store that is loaded into a data warehouse or other target system

https://www.ibm.com/topics/etl

How ETL works


The ETL Process Explained
Extract
Retrieves and verifies data from various sources
Transform
Processes and organizes extracted data so it is usable
Load
Moves transformed data to a data repository

https://www.informatica.com/resources/articles/what-is-etl.html

How ETL works – Extract

- Copy or export raw data from source locations to a staging area

- Extract data from a variety of data sources (structured/unstructured)
    - SQL or NoSQL servers
    - CRM and ERP systems
    - Flat files
    - Email
    - Web pages

https://www.ibm.com/topics/etl

How ETL works – Transform

- Transformed and consolidated the raw data in the staging area for its intended analytical use case

- This phase can involve the following tasks:
  - Filtering, cleansing, de-duplicating, validating, and authenticating the data.
  - Performing calculations, translations, or summarizations based on the raw data. This can include changing row and column headers for consistency, converting units of measurement, editing text strings, and more.
  - Conducting audits to ensure data quality and compliance

https://www.ibm.com/topics/etl

How ETL works – Load

- Moved the transformed data from the staging area into a target data warehouse.

- Typically, this involves an initial loading of all data, followed by periodic loading of incremental data changes and, less often, full refreshes to erase and replace data in the warehouse.

https://www.ibm.com/topics/etl

The benefits and challenges of ETL

- **Benefits:**
  - Improve quality by performing data cleansing prior to loading the data

- **Challenges:**
  - ETL is a time-consuming batch operation, which is more recommended for creating smaller target data repositories that require less frequent updating

https://www.ibm.com/topics/etl

ETL for ADDB – Framework

1. **Prepare CSV file and CREATE query file:**
   - Prepare and upload csv files that contains table data for later use
   - Prepare and upload CREATE query files corresponding to each csv file

2. **Run ETL bash:**
   - Upload csv files to Hadoop Distributed File System
   - Create a database in ADDB
   - Create tables by running the corresponding CREATE query file
   - Insert the table data from csv files into the corresponding tables

3. **Run ADDB:**
   - Run ADDB and verify if the ETL bash are functioning correctly

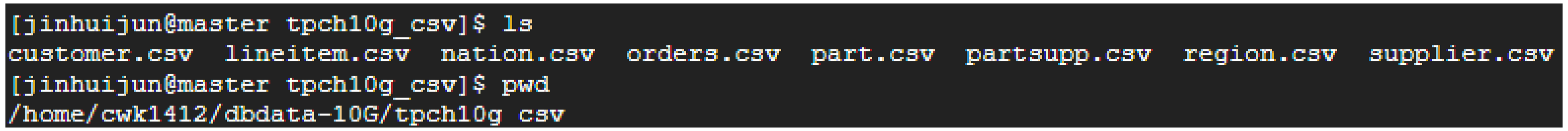
ETL for ADDB – Example

0. **Run the basic process mentioned in README file**
    - jps process and mounting
    - HDFS process
    - Redis process
    - ADDB-Spark process
        - cd addb-spark
        - ./addb_spark –start

ETL for ADDB – Example

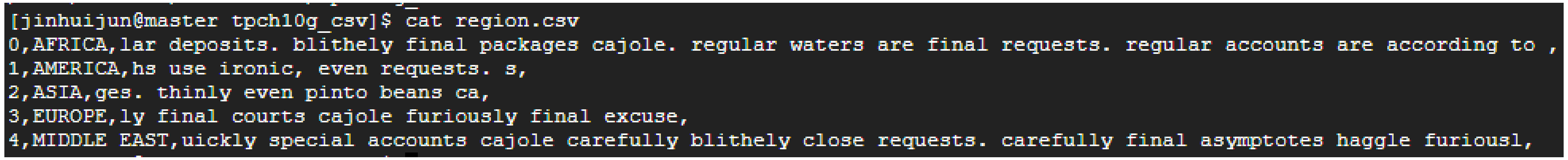
1. **Prepare CSV file and CREATE query file:**
   - Prepare and upload csv files that contains table data for later use
   - Prepare and upload CREATE query files corresponding to each csv file

Suppose we now have these csv files and their corresponding folder locations



```
[jinhuijun@master tpch10g_csv]$ ls
customer.csv  lineitem.csv  nation.csv  orders.csv  part.csv  partsupp.csv  region.csv  supplier.csv
[jinhuijun@master tpch10g_csv]$ pwd
/home/cwk1412/dbdata-10G/tpch10g_csv
```

The data format in the file is as follows

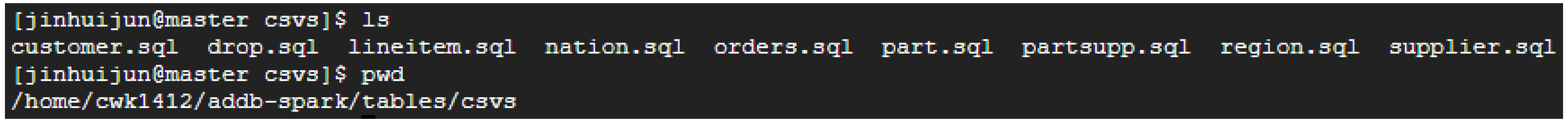


```
[jinhuijun@master tpch10g_csv]$ cat region.csv
0,AFRICA,lar deposits. blithely final packages cajole. regular waters are final requests. regular accounts are according to ,
1,AMERICA,hs use ironic, even requests. s,
2,ASIA,ges. thinly even pinto beans ca,
3,EUROPE,ly final courts cajole furiously final excuse,
4,MIDDLE EAST,uickly special accounts cajole carefully blithely close requests. carefully final asymptotes haggle furiousl,
```

ETL for ADDB – Example

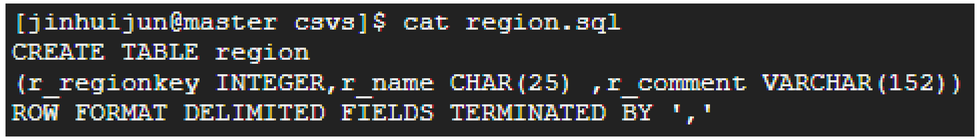
1. **Prepare CSV file and CREATE query file:**
   - Prepare and upload csv files that contains table data for later use
   - Prepare and upload CREATE query files corresponding to each csv file

Suppose we now have the corresponding CREATE query files and their corresponding folder locations



```
[jinhuijun@master csvs]$ ls
customer.sql  drop.sql  lineitem.sql  nation.sql  orders.sql  part.sql  partsupp.sql  region.sql  supplier.sql
[jinhuijun@master csvs]$ pwd
/home/cwk1412/addb-spark/tables/csvs
```

The data format in the file is as follows

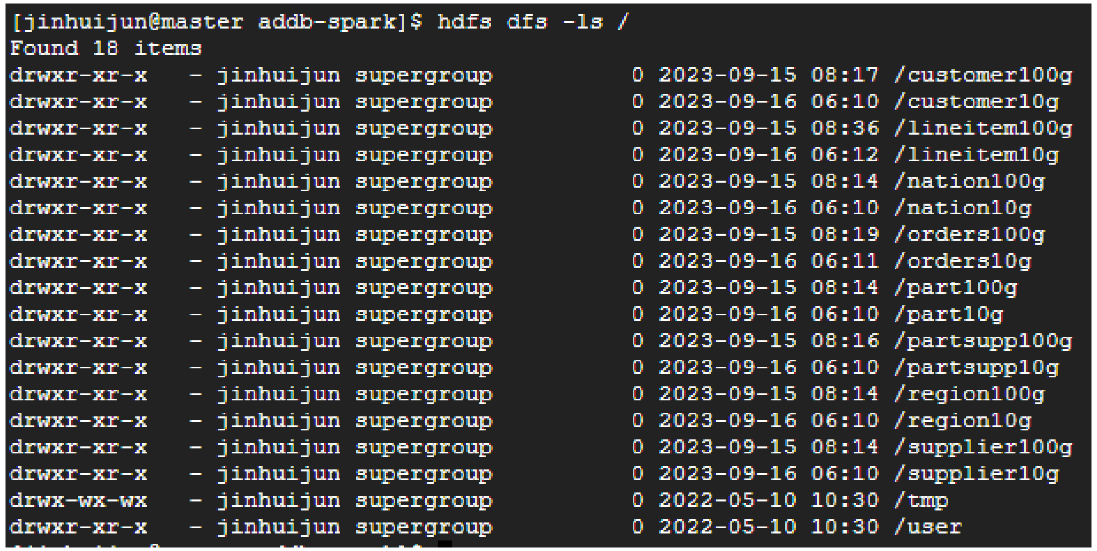


```
[jinhuijun@master csvs]$ cat region.sql
CREATE TABLE region
(r_regionkey INTEGER,r_name CHAR(25) ,r_comment VARCHAR(152))
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
```

ETL for ADDB – Framework

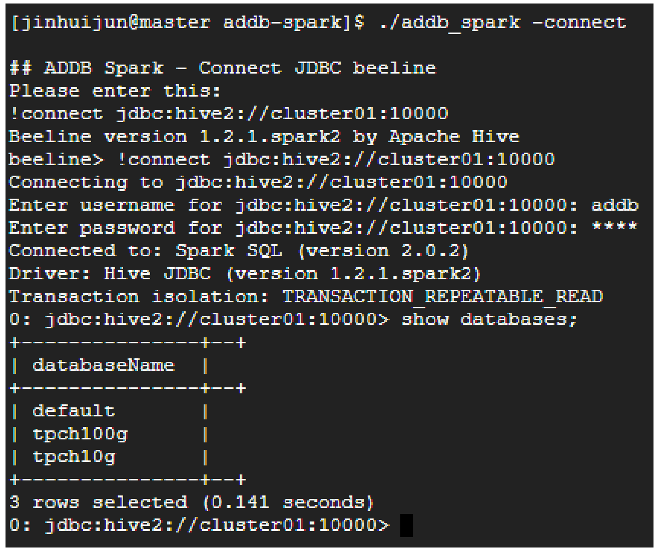
2. Run ETL bash:

- Before we run the ETL bash let us check the HDFS and ADDB
- Since this is a tutorial, we will start with no data on our current HDFS and ADDB
- Of course, there is no problem that these relevant data has already been uploaded to the system



```
[jinhuijun@master addb-spark]$ hdfs dfs -ls /
Found 18 items
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:17 /customer100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /customer10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:36 /lineitem100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:12 /lineitem10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:14 /nation100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /nation10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:19 /orders100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:11 /orders10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:14 /part100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /part10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:16 /partsupp100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /partsupp10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:14 /region100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /region10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:14 /supplier100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /supplier10g
drwx-wx-wx   - jinhuijun supergroup          0 2022-05-10 10:30 /tmp
drwxr-xr-x   - jinhuijun supergroup          0 2022-05-10 10:30 /user
```

HDFS



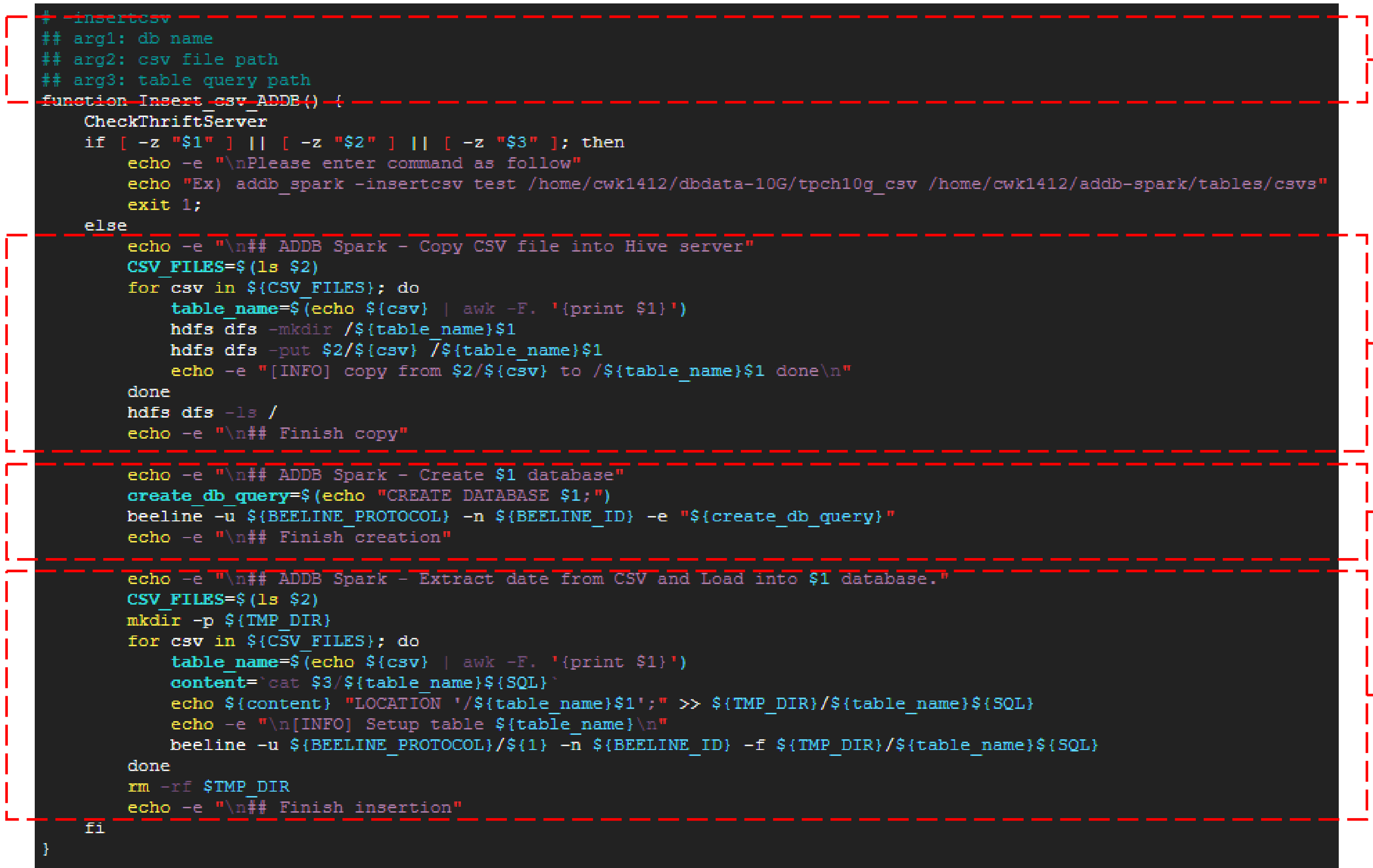
```
[jinhuijun@master addb-spark]$ ./addb_spark -connect

## ADDB Spark - Connect JDBC beeline
Please enter this:
!connect jdbc:hive2://cluster01:10000
Beeline version 1.2.1.spark2 by Apache Hive
beeline> !connect jdbc:hive2://cluster01:10000
Connecting to jdbc:hive2://cluster01:10000
Enter username for jdbc:hive2://cluster01:10000: addb
Enter password for jdbc:hive2://cluster01:10000: ****
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000> show databases;
+---------------+--+
| databaseName  |
+---------------+--+
| default       |
| tpch100g      |
| tpch10g       |
+---------------+--+
3 rows selected (0.141 seconds)
0: jdbc:hive2://cluster01:10000>
```

ADDB

ETL for ADDB – Framework

2. Run ETL bash:

- The corresponding bash located in “/home/cwk1412/addb-spark/addb_spark”



```bash
# -insertcsv
## arg1: db name
## arg2: csv file path
## arg3: table query path
function Insert_csv_ADDB() {
    CheckThriftServer
    if [ -z "$1" ] || [ -z "$2" ] || [ -z "$3" ]; then
        echo -e "\nPlease enter command as follow"
        echo "Ex) addb_spark -insertcsv test /home/cwk1412/dbdata-10G/tpch10g_csv /home/cwk1412/addb-spark/tables/csvs"
        exit 1;
    else
        echo -e "\n## ADDB Spark - Copy CSV file into Hive server"
        CSV_FILES=$(ls $2)
        for csv in ${CSV_FILES}; do
            table_name=$(echo ${csv} | awk -F. '{print $1}')
            hdfs dfs -mkdir /${table_name}$1
            hdfs dfs -put $2/${csv} /${table_name}$1
            echo -e "[INFO] copy from $2/${csv} to /${table_name}$1 done\n"
        done
        hdfs dfs -ls /
        echo -e "\n## Finish copy"
        echo -e "\n## ADDB Spark - Create $1 database"
        create_db_query=$(echo "CREATE DATABASE $1;")
        beeline -u ${BEELINE_PROTOCOL} -n ${BEELINE_ID} -e "${create_db_query}"
        echo -e "\n## Finish creation"
        echo -e "\n## ADDB Spark - Extract date from CSV and Load into $1 database."
        CSV_FILES=$(ls $2)
        mkdir -p ${TMP_DIR}
        for csv in ${CSV_FILES}; do
            table_name=$(echo ${csv} | awk -F. '{print $1}')
            content=`cat $3/${table_name}${SQL}`
            echo ${content} "LOCATION '/${table_name}$1';" >> ${TMP_DIR}/${table_name}${SQL}
            echo -e "\n[INFO] Setup table ${table_name}\n"
            beeline -u ${BEELINE_PROTOCOL}/${1} -n ${BEELINE_ID} -f ${TMP_DIR}/${table_name}${SQL}
        done
        rm -rf $TMP_DIR
        echo -e "\n## Finish insertion"
    fi
}
```

- *This command requires three arguments*

- *Upload the csv file to Hadoop Distributed File System*

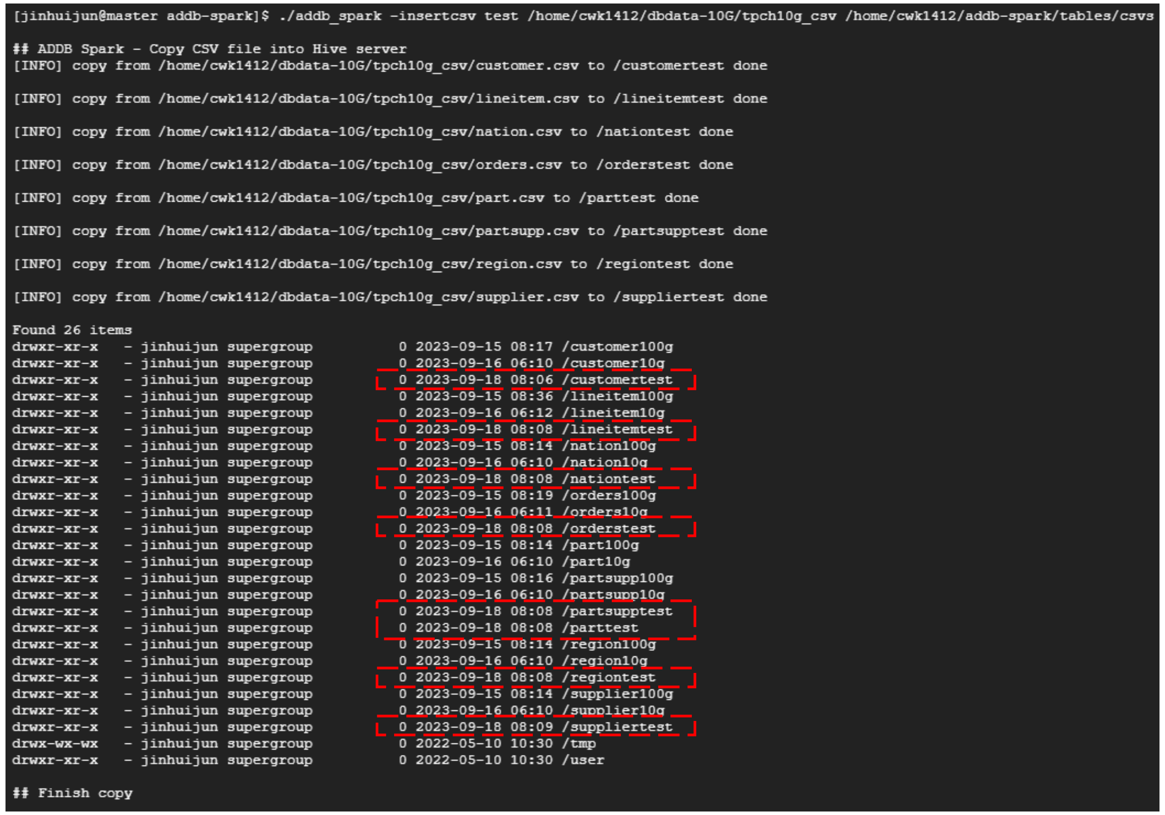
- *Create a database in ADDB*

- *Create the table by running corresponding CREATE query file*
- *Insert the table data from csv file to corresponding table*

ETL for ADDB – Framework

2. Run ETL bash:

- Now let's use the command "./addb_spark -insertcsv {*DBname*} {*csv_files_path*} {*create_query_files_path*}" to run ETL



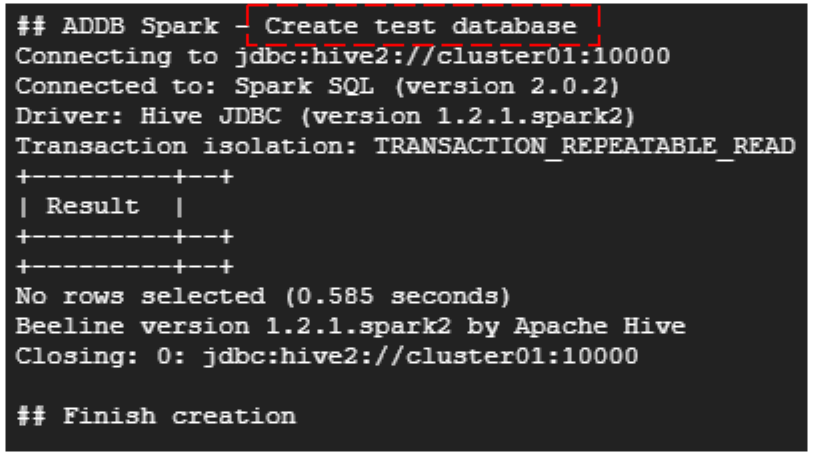
```
[jinhuijun@master addb-spark]$ ./addb_spark -insertcsv test /home/cwk1412/dbdata-10G/tpch10g_csv /home/cwk1412/addb-spark/tables/csvs

## ADDB Spark - Copy CSV file into Hive server
[INFO] copy from /home/cwk1412/dbdata-10G/tpch10g_csv/customer.csv to /customertest done

[INFO] copy from /home/cwk1412/dbdata-10G/tpch10g_csv/lineitem.csv to /lineitemtest done

[INFO] copy from /home/cwk1412/dbdata-10G/tpch10g_csv/nation.csv to /nationtest done

[INFO] copy from /home/cwk1412/dbdata-10G/tpch10g_csv/orders.csv to /orderstest done

[INFO] copy from /home/cwk1412/dbdata-10G/tpch10g_csv/part.csv to /parttest done

[INFO] copy from /home/cwk1412/dbdata-10G/tpch10g_csv/partsupp.csv to /partsupptest done

[INFO] copy from /home/cwk1412/dbdata-10G/tpch10g_csv/region.csv to /regiontest done

[INFO] copy from /home/cwk1412/dbdata-10G/tpch10g_csv/supplier.csv to /suppliertest done

Found 26 items
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:17 /customer100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /customer10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-18 08:06 /customertest
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:36 /lineitem100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:12 /lineitem10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-18 08:08 /lineitemtest
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:14 /nation100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /nation10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-18 08:08 /nationtest
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:19 /orders100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:11 /orders10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-18 08:08 /orderstest
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:14 /part100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /part10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:16 /partsupp100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /partsupp10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-18 08:08 /partsupptest
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-18 08:08 /parttest
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:14 /region100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /region10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-18 08:08 /regiontest
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-15 08:14 /supplier100g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-16 06:10 /supplier10g
drwxr-xr-x   - jinhuijun supergroup          0 2023-09-18 08:09 /suppliertest
drwx-wx-wx   - jinhuijun supergroup          0 2022-05-10 10:30 /tmp
drwxr-xr-x   - jinhuijun supergroup          0 2022-05-10 10:30 /user

## Finish copy
```

- *Upload the csv file to Hadoop Distributed File System*

ETL for ADDB – Framework

2. Run ETL bash:

- Now let’s use the command “./addb_spark -insertcsv {*DBname*} {*csv_files_path*} {*create_query_files_path*}” to run ETL



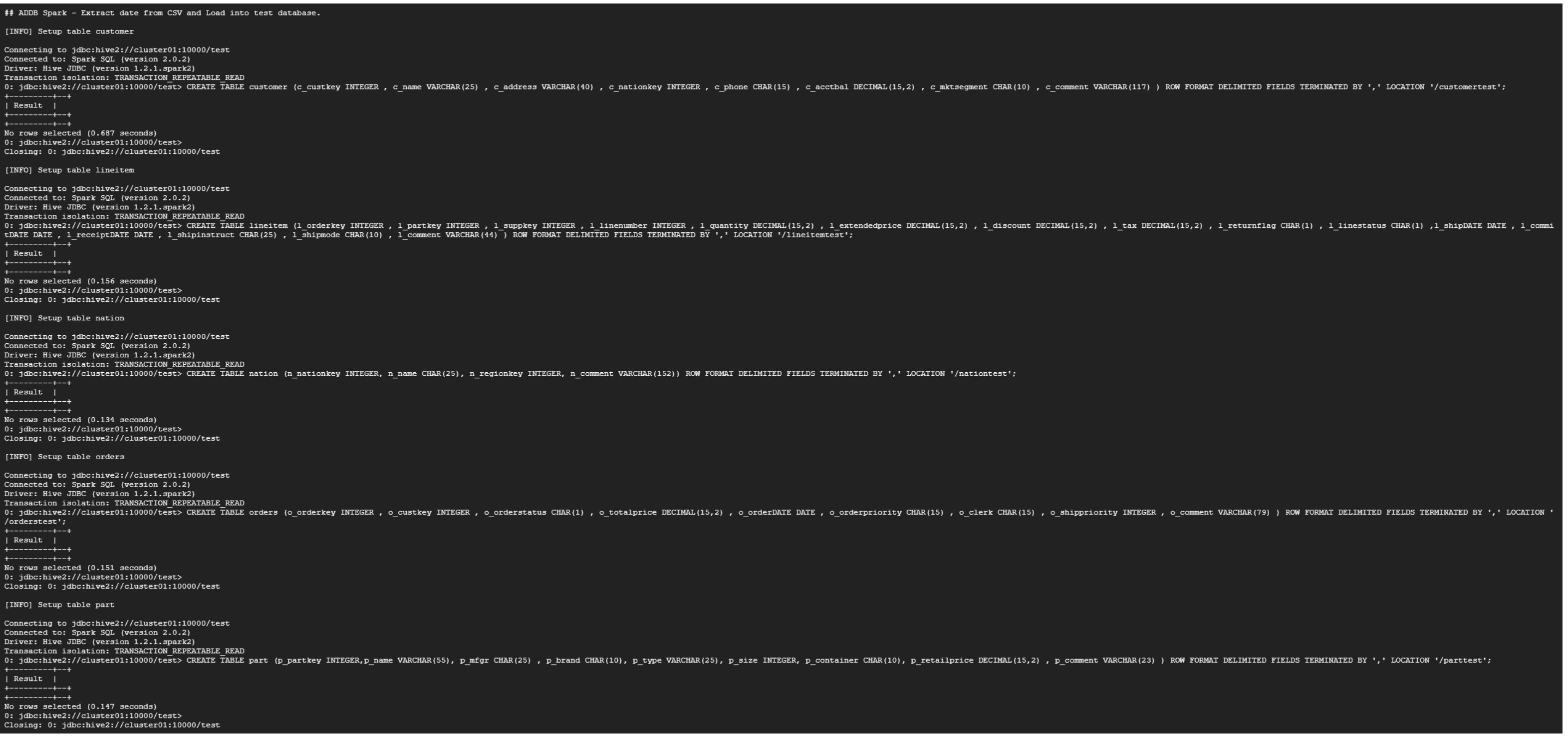
```
## ADDB Spark - Create test database
Connecting to jdbc:hive2://cluster01:10000
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.585 seconds)
Beeline version 1.2.1.spark2 by Apache Hive
Closing: 0: jdbc:hive2://cluster01:10000

## Finish creation
```

- *Create a database in ADDB*

ETL for ADDB – Framework

2. Run ETL bash:

- Now let’s use the command “./addb_spark -insertcsv {*DBname*} {*csv_files_path*} {*create_query_files_path*}” to run ETL



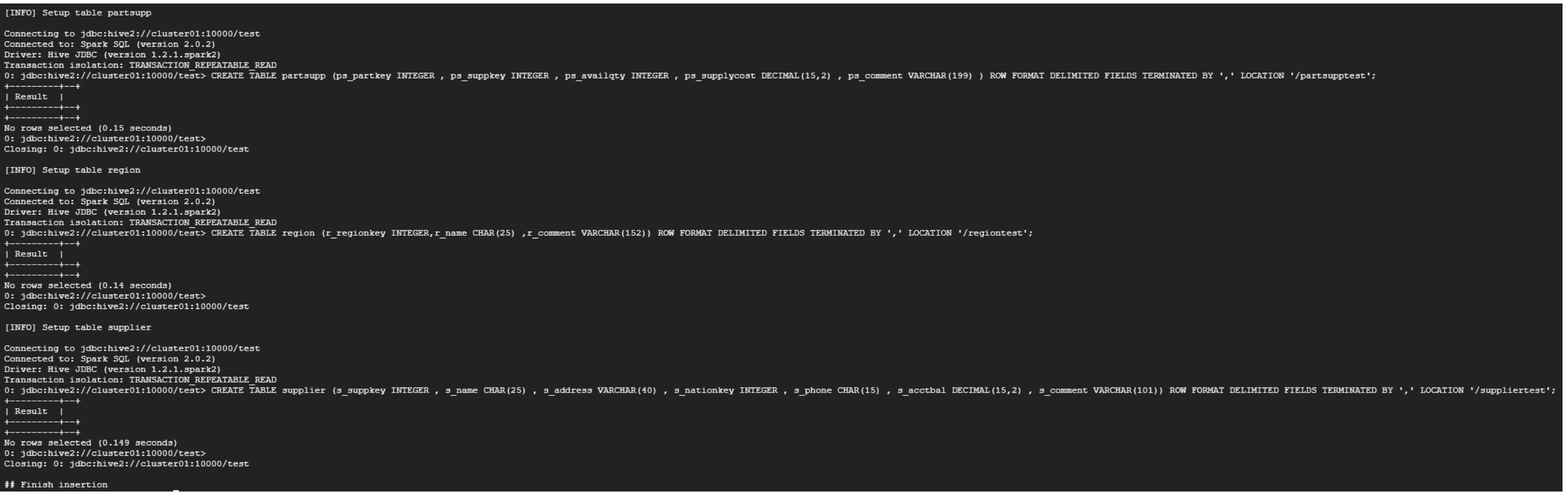
```
## ADDB Spark - Extract date from CSV and Load into test database.

[INFO] Setup table customer

Connecting to jdbc:hive2://cluster01:10000/test
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000/test> CREATE TABLE customer (c_custkey INTEGER , c_name VARCHAR(25) , c_address VARCHAR(40) , c_nationkey INTEGER , c_phone CHAR(15) , c_acctbal DECIMAL(15,2) , c_mktsegment CHAR(10) , c_comment VARCHAR(117) ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/customertest';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.687 seconds)
0: jdbc:hive2://cluster01:10000/test>
Closing: 0: jdbc:hive2://cluster01:10000/test

[INFO] Setup table lineitem

Connecting to jdbc:hive2://cluster01:10000/test
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000/test> CREATE TABLE lineitem (l_orderkey INTEGER , l_partkey INTEGER , l_suppkey INTEGER , l_linenumber INTEGER , l_quantity DECIMAL(15,2) , l_extendedprice DECIMAL(15,2) , l_discount DECIMAL(15,2) , l_tax DECIMAL(15,2) , l_returnflag CHAR(1) , l_linestatus CHAR(1) ,l_shipDATE DATE , l_commi
tDATE DATE , l_receiptDATE DATE , l_shipinstruct CHAR(25) , l_shipmode CHAR(10) , l_comment VARCHAR(44) ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/lineitemtest';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.156 seconds)
0: jdbc:hive2://cluster01:10000/test>
Closing: 0: jdbc:hive2://cluster01:10000/test

[INFO] Setup table nation

Connecting to jdbc:hive2://cluster01:10000/test
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000/test> CREATE TABLE nation (n_nationkey INTEGER, n_name CHAR(25), n_regionkey INTEGER, n_comment VARCHAR(152)) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/nationtest';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.134 seconds)
0: jdbc:hive2://cluster01:10000/test>
Closing: 0: jdbc:hive2://cluster01:10000/test

[INFO] Setup table orders

Connecting to jdbc:hive2://cluster01:10000/test
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000/test> CREATE TABLE orders (o_orderkey INTEGER , o_custkey INTEGER , o_orderstatus CHAR(1) , o_totalprice DECIMAL(15,2) , o_orderDATE DATE , o_orderpriority CHAR(15) , o_clerk CHAR(15) , o_shippriority INTEGER , o_comment VARCHAR(79) ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '
/orderstest';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.151 seconds)
0: jdbc:hive2://cluster01:10000/test>
Closing: 0: jdbc:hive2://cluster01:10000/test

[INFO] Setup table part

Connecting to jdbc:hive2://cluster01:10000/test
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000/test> CREATE TABLE part (p_partkey INTEGER,p_name VARCHAR(55), p_mfgr CHAR(25) , p_brand CHAR(10), p_type VARCHAR(25), p_size INTEGER, p_container CHAR(10), p_retailprice DECIMAL(15,2) , p_comment VARCHAR(23) ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/parttest';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.147 seconds)
0: jdbc:hive2://cluster01:10000/test>
Closing: 0: jdbc:hive2://cluster01:10000/test
```

- *Create the table by running corresponding CREATE query file*
- *Insert the table data from csv file to corresponding table*

ETL for ADDB – Framework

2. Run ETL bash:

- Now let’s use the command “./addb_spark -insertcsv {*DBname*} {*csv_files_path*} {*create_query_files_path*}” to run ETL



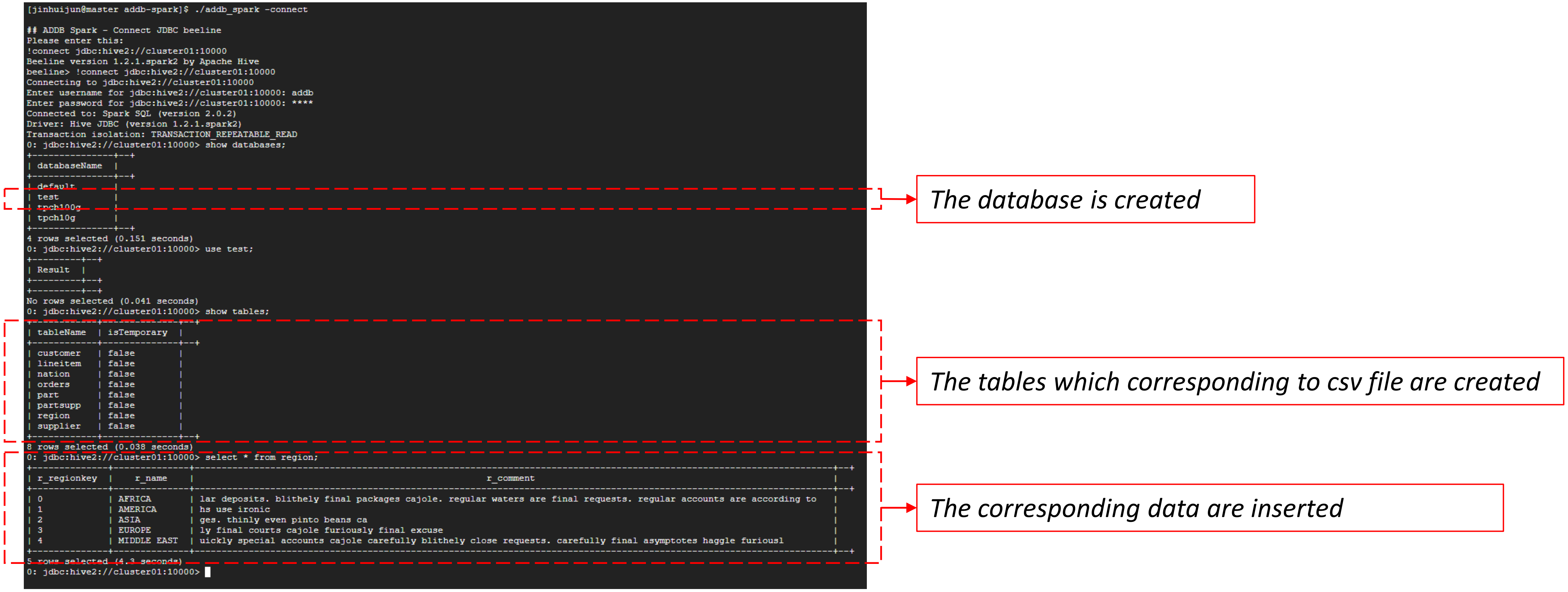
```
[INFO] Setup table partsupp

Connecting to jdbc:hive2://cluster01:10000/test
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000/test> CREATE TABLE partsupp (ps_partkey INTEGER , ps_suppkey INTEGER , ps_availqty INTEGER , ps_supplycost DECIMAL(15,2) , ps_comment VARCHAR(199) ) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/partsupptest';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.15 seconds)
0: jdbc:hive2://cluster01:10000/test>
Closing: 0: jdbc:hive2://cluster01:10000/test

[INFO] Setup table region

Connecting to jdbc:hive2://cluster01:10000/test
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000/test> CREATE TABLE region (r_regionkey INTEGER,r_name CHAR(25) ,r_comment VARCHAR(152)) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/regiontest';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.14 seconds)
0: jdbc:hive2://cluster01:10000/test>
Closing: 0: jdbc:hive2://cluster01:10000/test

[INFO] Setup table supplier

Connecting to jdbc:hive2://cluster01:10000/test
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000/test> CREATE TABLE supplier (s_suppkey INTEGER , s_name CHAR(25) , s_address VARCHAR(40) , s_nationkey INTEGER , s_phone CHAR(15) , s_acctbal DECIMAL(15,2) , s_comment VARCHAR(101)) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/suppliertest';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.149 seconds)
0: jdbc:hive2://cluster01:10000/test>
Closing: 0: jdbc:hive2://cluster01:10000/test

## Finish insertion
```

- *Create the table by running corresponding CREATE query file*
- *Insert the table data from csv file to corresponding table*

ETL for ADDB – Framework

3. Run ADDB:

- Run ADDB and verify if the ETL bash are functioning correctly



```
[jinhuijun@master addb-spark]$ ./addb_spark -connect

## ADDB Spark - Connect JDBC beeline
Please enter this:
!connect jdbc:hive2://cluster01:10000
Beeline version 1.2.1.spark2 by Apache Hive
beeline> !connect jdbc:hive2://cluster01:10000
Connecting to jdbc:hive2://cluster01:10000
Enter username for jdbc:hive2://cluster01:10000: addb
Enter password for jdbc:hive2://cluster01:10000: ****
Connected to: Spark SQL (version 2.0.2)
Driver: Hive JDBC (version 1.2.1.spark2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
0: jdbc:hive2://cluster01:10000> show databases;
+---------------+--+
| databaseName  |
+---------------+--+
| default       |
| test          |
| tpch100g      |
| tpch10g       |
+---------------+--+
4 rows selected (0.151 seconds)
0: jdbc:hive2://cluster01:10000> use test;
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.041 seconds)
0: jdbc:hive2://cluster01:10000> show tables;
+------------+--------------+--+
| tableName  | isTemporary  |
+------------+--------------+--+
| customer   | false        |
| lineitem   | false        |
| nation     | false        |
| orders     | false        |
| part       | false        |
| partsupp   | false        |
| region     | false        |
| supplier   | false        |
+------------+--------------+--+
8 rows selected (0.038 seconds)
0: jdbc:hive2://cluster01:10000> select * from region;
+--------------+--------------+--------------------------------------------------------------------------------------------------------------------------+--+
| r_regionkey  |    r_name    |                                                        r_comment                                                         |
+--------------+--------------+--------------------------------------------------------------------------------------------------------------------------+--+
| 0            | AFRICA       | lar deposits. blithely final packages cajole. regular waters are final requests. regular accounts are according to       |
| 1            | AMERICA      | hs use ironic                                                                                                            |
| 2            | ASIA         | ges. thinly even pinto beans ca                                                                                          |
| 3            | EUROPE       | ly final courts cajole furiously final excuse                                                                            |
| 4            | MIDDLE EAST  | uickly special accounts cajole carefully blithely close requests. carefully final asymptotes haggle furiousl             |
+--------------+--------------+--------------------------------------------------------------------------------------------------------------------------+--+
5 rows selected (4.3 seconds)
0: jdbc:hive2://cluster01:10000>
```

The database is created
The tables which corresponding to csv file are created
The corresponding data are inserted

Demonstration video..

ADDB ETL 툴 기능 테스트 (100GB)

- https://www.youtube.com/watch?v=eTdK3H7al0Y

ADDB ETL 툴 기능 테스트 (10GB)

- https://www.youtube.com/watch?v=TExrANTyLlI&t=2s

Reference Sites

- https://www.ibm.com/topics/etl
- https://itholic.github.io/etl/
- https://www.integrate.io/ko/blog/the-top-7-etl-tools-ko/