Hadoop for Data Science

参考 ： https://www.lynda.com/Hadoop-tutorials/Hadoop-Data-Science-Tips-Tricks-Techniques/585009-2.html

1. Working with Files

Organize files in HDFS

eg:
mkdir hadoop_for_ds
ls
cp -rf /media/sf_Exercise_Files/* hadoop_for_ds/
cd hadoop_for_ds/
ls
cat /data/clients.csv
head -15 data/clients.csv
head -1 data/clients.csv
tail -15 data/clients.csv
eg:
hadoop fs -mkdir /data
hadoop fs -mkdir /data/sales
hadoop fs -mkdir /data/clients
hadoop fs -ls /data

Upload files in HDFS
hadoop fs -put /data/clients.csv /data/clients
hadoop fs -ls /data/clients
hadoop fs -put ~/hadoop_for_ds/data/sales-yearly/CogsleyServices-SalesData-2009.csv /data/sales
hadoop fs -ls /data/sales
wget "https://s3.amazonaws.com/data.openaddress.io/runs/152829/us/ca/san_diego.zip"
ls
unzip san_diego.zip
head -50 /us/ca/san_diego.csv
hadoop fs -mkdir /data/addresses
hadoop fs -put ~/hadoop_for_ds/us/ca/san_diego.csv /data/addresses
hadoop fs -ls /data/addresses

Move files in HDFS
hadoop fs -mkdir /raw
hadoop fs -mkdir /raw/sales /raw/clients : make two directories on the same line
hadoop fs -mkdir /tmp
hadoop fs -ls /raw
hadoop fs -cp /data/clients/clients.csv /raw/clients/
hadoop fs -ls /raw/clients
hadoop fs -cp /data/sales/CogsleyServices-SalesData-2009.csv /raw/sales
hadoop fs -ls /raw/sales

Remove files in HDFS

eg:
hadoop fs -rm /data/clients/clients.csv
hadoop fs -ls /data/clients
hadoop fs -rmdir /data/sales : this only removes empty directory
hadoop fs -rm -R /data/sales : this can remove directories that contain files.
hadoop fs -ls /data
hadoop fs -mkdir /tmp_folder

```
hadoop fs -rm /tmp_folder : this will not work, since /tmp_folder is a directory
hadoop fs -rmdir /tmp_folder : this will work
```

2. Connecting to Hadoop

Explore Hive through Beeline

```
beeline
!connect jdbc:hive2://
username
password
show schemas;
use default;
show tables;
describe customers
select * from customers limit 10;
!quit
```

Access Hive in Python

```
sudo yum install python-pip
sudo pip install --upgrade pip
sudo yum install gcc gcc-c++ make openss-devel
sudo yum install python-devel.x86_64
sudo yum install cyrus-sasl-devel.x86_64
sudo pip install pyhs2
python
import pyhs2
hive_connection = pyhs2.connect(host='localhost', port=10000, authMechanism='PLAIN', user='cloudera', password='cloudera',
database='default')
hive_cursor = hive_connection.cursor()
hive_cursor.execute('select * from default.customers limit 10')
vals = hivecursor.fetchall()
for row in vals :
    print row
```

Create aggregates in Hive

```
hadoop fs -put data/sales-yearly/* /raw/sales
hadoop fs -ls /raw/sales
hadoop fs -put setup/csv-serde-1.1.2-0.11.0-all.jar /user/cloudera
-- switch to hue and run this in hive to install csv serde
add jar hdfs:///user/cloudera/csv-serde-1.1.2-0.11.0-all.jar;
--setup back_office database
create database backup_office;
use back_office;
create external table stage_sales(
...)
row format serde 'com.bizo.hive.serde.csv.CSVSerde'
with SERDEPROPERITIES (
    "seperatorChar" = ",",
    "quoteChar" = '"',
    "escapeChar" = "\\"
)
location '/raw/sales/';
select * from back_office.stage_sales;
create table back_office.monthly_sales_totals
```

```
stored as parquet
tblproperities('parquet.compression'='SNAPPY') as
select companyname, ordermonthyear, sum(saleamount) as sale_total
from stage_sales
group by companyname, ordermonthyear;
select * from back_office.monthly_sales_totals;

Select partitions in Hive

create table back_office.sales_all_years_partitioned (
...
)
partitioned by (companyname_partition string)
stored as textfile;
insert into table back_office.sales_all_years_partitioned partition (companyname_partition) select *, Companyname from
back_office.stage_sales;
-- now look at the folder structure in hdfs
hadoop fs -ls /user/hive/warehouse/back_office.db/sales_all_years_partitioned
hadoop fs -cat /user/hive/warehouse/back_office.db/sales_all_years_partitioned/companyname_partition=DIRECTV/*
select * from back_office.sales_all_years_partitioned where companyname_partition='DIRECTV';
select * from back_office.sales_all_years_partitioned where companyname='DIRECTV';
explain select * from back_office.sales_all_years_partitioned where companyname_partition='DIRECTV';
explain select * from back_office.sales_all_years_partitioned where companyname='DIRECTV';



3. Complex Data Structures in Hive

Map data in Hive

select addresses from customers;
select addresses['billing'] from customers;
select addresses['billing'].city from customers;
select addresses['billing'].zip_code from customers;



Arrays in Hive
select id, name, orders from customers limit 10;
-- get the first order of each customer
select id, name, orders[0] from customers limit 10;
-- get the third order of each customer
select id, name, orders[2] from customers limit 10;
select id, name, orders[0].items as first_order_items from customers limit 10;
select id, name, orders[0].items[0] as first_item_from_first_order from customers limit 10;
select id, name, orders[0].items[0].name as first_item_name_from_first_order from customers limit 10;

Structs in Hive
select name as customer_name, orders from orders limit 10;
select name as customer_name, exploded_orders from orders lateral view explode(orders) o as exploded_orders order by
customer_name;
with pivoted_orders as (
    select name as customer_name, exploded_orders
    from orders
    lateral view explode(orders) o as exploded_orders
)
select customer_name,
exploded_orders.order_id as order_id,
exploded_orders.order_date as order_date,
exploded_items.product_id as item_product_id,
```

```
exploded_items.sku as item_sku,
exploded_items.name as item_name,
exploded_items.price as item_price,
exploded_items.qty as item_quantity
from pivoted_orders
lateral view explode(exploded_orders.items) i as exploded_items
order by customer_name, order_id, order_date;

Create flat tables for Impala
create table default.customer_orders
stored as parquet
tblproperties('parquet.compression'='SNAPPY') as
with pivoted_orders as (
    select name as customer_name, exploded_orders
    from orders
    lateral view explode(orders) o as exploded_orders
)
select customer_name,
exploded_orders.order_id as order_id,
from_unixtime(UNIX_TIMESTAMP(concat(substr(exploded_orders.order_date,1,10),' ', substr(exploded_orders.order_date,12,20)) as
order_date,
exploded_items.product_id as item_product_id,
exploded_items.sku as item_sku,
exploded_items.name as item_name,
exploded_items.price as item_price,
exploded_items.qty as item_quantity
from pivoted_orders
lateral view explode(exploded_orders.items) i as exploded_items
order by customer_name, order_id, order_date;
select * from customer_orders limit 10;
select orders from customers limit 10; -- this will give an error, since Impala doesn't have the abilit to query complex types.


Deconstruct Impala queries
select * from customer_orders limit 100;
compute stats default.customer_orders;
explain select * from customer_orders limit 100;
explain select * from customer_orders where customer_name = 'Melvin Garcia';
explain select c.id, c.name, month(order_date) as order_month, year(order_date) as order_year, sum(item_price) as
total_sales_amount, sum(item_price)/sum(item_quantity) as average_price_per_item
from customer_orders o
left join default.customers c on o.customer_name=c.name
group by c.id, c.name, order_month, order_year;
```