# [Hadoop]Hive Lab04

## 학습 목표

가. Hue에 대해 알아보자.

나. 데이터를 올리고 간단한 Query를 실행시켜보자.

# 목차

## [Hadoop]Hive Lab04 Hue

- 1-1 Hue에 대한 알아보기
- 1-1 데이터 다운로드 후, HDFS에 올리기
- 1-2 csv 파일을 이용하여 테이블 생성하기
- 1-3 기본 데이터 조회(select ~ )
- 1-4 Simple aggregations
- 1-5 grouping sets 함수
- 1-6 between을 이용한 기간
- 1-7 cube와 rollup
- 1-8 cube vs rollup
- 1-9 grouping sets(2) 항목이 두개
- 1-10 SUBSTRING() 함수를 이용한 년도 추출
- 1-11 having 절을 이용한 그룹화 결과 값의 조건 으로 조회하기

### 1-1 Hue에 대한 알아보기

- Hue(Hadoop User Experience)는 Apache Hadoop 클러스터와 함께 사용되는 웹 기반 사용자 인터페이스
- Hue(Hadoop User Experience)는 Hadoop 에코 시스템과 함께 그룹화되어 Hive 작업 및 Pig 스크립트 등을 실행 가능.

### 1-1 데이터 불러오기

[cloudera@quickstart ~]\$ cd ~ [cloudera@quickstart ~]\$ cd datase [cloudera@quickstart dataset]\$ cd Cogsley Data [cloudera@quickstart Cogsley\_Data]\$ ls -ltr total 4336

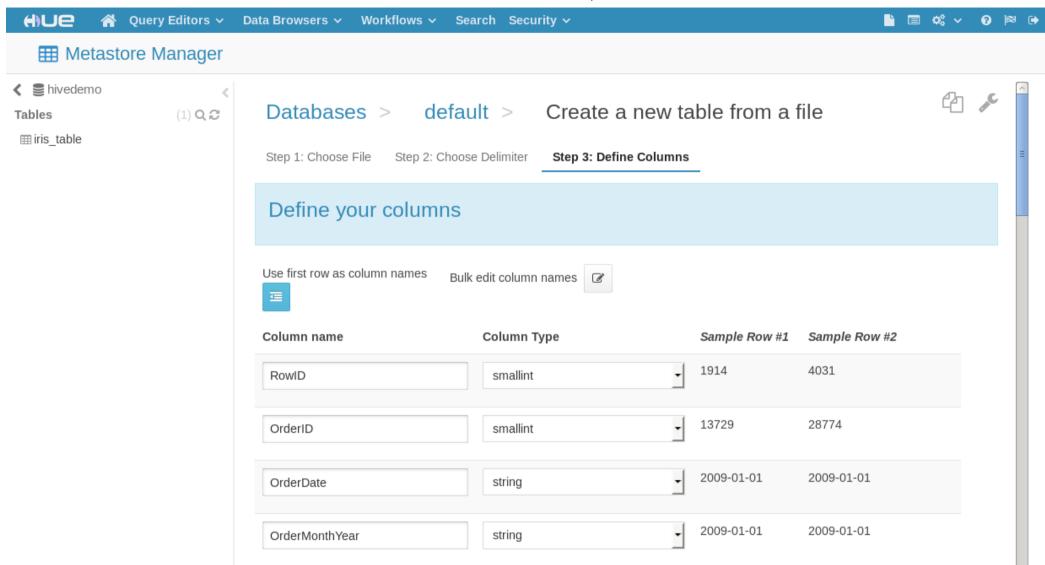
-rw-rw-r-- 1 cloudera cloudera 2162356 Jun 9 21:53 CogsleyServices SalesData US.csv

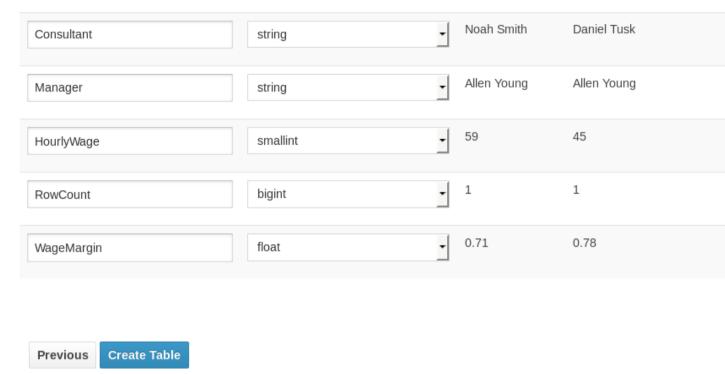
[cloudera@quickstart Cogsley\_Data]\$ hdfs dfs -put CogsleyServices\_SalesData\_US.csv

### HUE 환경에서 데이터 테이블 만들기

Hue 선택 - Data Browsers - Metastore tables - File로 부터 테이블 만들기 - 파일 선택 후, 진행.

2019. 2. 25. **Evernote Export** 





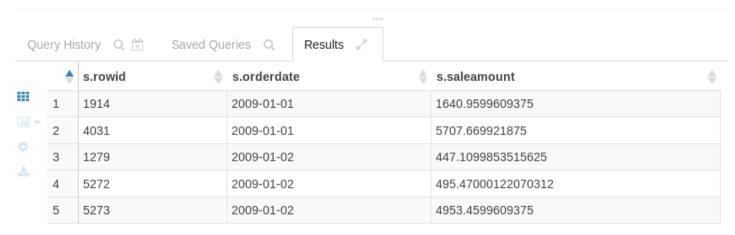
'Create Table' 선택하기.

# 1-3 기본 데이터 조회(select ~ )

(가) 100개의 행의 전체 컬럼을 확인하기 select \* from sales\_nocomma limit 100;

### (나) 테이블 이름을 약자로 지정하고 이름을 이용하여 컬럼 선택하기

```
select
   s.rowid,
   s.orderdate,
   s.saleamount
from
   sales_nocomma s
limit 100;
```



## (다) 컬럼 선택하고 해당 이름을 넣기

```
select
   s.rowid as RowNum,
  s.orderdate as OrderDate,
   s.saleamount as Sales
   sales_nocomma s
limit 100;
```

## [실행결과]

	rownum	orderdate		
1	1914	2009-01-01	1640.9599609375	
2	4031	2009-01-01	5707.669921875	
3	1279	2009-01-02	447.1099853515625	
4	5272	2009-01-02	495.47000122070312	
5	5273	2009-01-02	4953.4599609375	
6	5274	2009-01-02	6024.919921875	

# 1-4 Simple aggregations

SUM(), MIN(), MAX(), AVG(), COUNT() 함수를 이용하여 해당 결과에 대한 내용을 그룹별로 요약이 가능하다.

## (가) 컬럼 선택하고 해당 이름을 넣기

as 명령어를 이용하여 컬럼명을 지정할 수 있다.

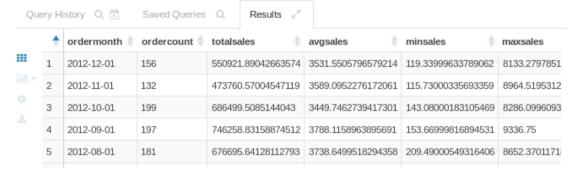
sales nocomma 이후에 약자 's'를 적어 테이블 명을 사용할 수 있다.

```
select
ordermonthyear as OrderMonth,
count(1) as OrderCount,
sum(s.saleamount) as TotalSales,
avg(s.saleamount) as AvgSales,
min(s.saleamount) as MinSales,
max(s.saleamount) as MaxSales

from
sales_nocomma s
where
lower(ordermonthyear) != 'ordermonthyear'
group by
ordermonthyear
order by
ordermonthyear desc;
```

### [실행결과]

2019. 2. 25.



## 1-5 grouping sets 함수

- GROUPING SETS은 ROLLUP이나 CUBE 처럼 GROUP BY 절에 명시해서 그룹 쿼리에 사용된다.
- GROUPING SETS절은 그룹 쿼리이나 UNION ALL 개념이 섞여 있다.
- GROUPING SETS (expr1, expr2, expr3)를 GROUP BY 절에 명시했을 때, 괄호 안에 있는 세 표현식별로 각각 집계가 이루어짐.
  - ((GROUP BY expr1) UNION ALL (GROUP BY expr2) UNION ALL (GROUP BY expr3)) 형태를 갖음.

```
-- Enhanced aggregations with grouping sentences
select
    ordermonthyear as OrderMonth,
    productcategory as Category,
    sum(saleamount) as TotalSales
from
```

```
sales nocomma
where
    lower(ordermonthyear) != 'ordermonthyear'
group by
    ordermonthyear,
   productcategory
-- enhancing
grouping sets
    (ordermonthyear, productcategory) -- same as union of two queries with group by of a and b separately
```

### [실행결과]

	*	ordermonth $\qquad \qquad \qquad$	category	totalsales $\protect\ =$
	1	NULL	Consulting	7511609.6118469238
.id =	2	NULL	Development	16437797.044464111
	3	NULL	Training	5315590.8316497803
	4	2009-01-01	NULL	734559.35684204102
	5	2009-02-01	NULL	539887.79889678955
	6	2009-03-01	NULL	559449.95849609375
	7	2009-04-01	NULL	614983.31002807617

그룹 함수의 종류는 ROLLUP, CUBE, GROUPING SETS 등의 함수가 있다.

## 1-6 between을 이용한 기간

# CUBE는 가능한 조합별로 집계를 수행한다.

```
select
    ordermonthyear as OrderMonth,
    productcategory as Category,
    sum(saleamount) as TotalSales
from
    sales_nocomma
where
    ordermonthyear between '2009-01-01' and '2009-02-01'
group by
    ordermonthyear,
    productcategory
with cube;
                                                                                                                                                             Colored by Color Scripter
```

		ordermonth \$	category	totalsales
	1	NULL	NULL	1274447.1557388306
.hl w	2	NULL	Consulting	292113.04821777344
	3	NULL	Development	701459.01047515869
	4	NULL	Training	280875.09704589844
	5	2009-01-01	NULL	734559.35684204102
	6	2009-01-01	Consulting	147329.35925292969
	7	2009-01-01	Development	435158.64923095703
	8	2009-01-01	Training	152071.3483581543
	9	2009-02-01	NULL	539887.79889678955
	10	2009-02-01	Consulting	144783.68896484375
	11	2009-02-01	Development	266300.36124420166
	12	2009-02-01	Training	128803.74868774414

rollup는 그룹화된 컬럼의 subtotal 값을 생성한다.

# 1-7 cube와 rollup

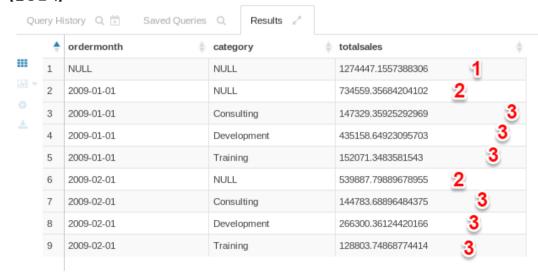
ROLLUP과 CUBE는 GROUP BY 절에서 사용되며 그룹별 소계를 추가로 보여주는 역할을 한다. ROLLUP은 레벨별로, CUBE는 가능한 조합별로 집계를 수행한다.

```
select
    ordermonthyear as OrderMonth,
    productcategory as Category,
    sum(saleamount) as TotalSales
from
    sales_nocomma
where
    lower(ordermonthyear) != 'ordermonthyear'
group by
    ordermonthyear,
    productcategory
with cube;
```

	*	ordermonth #	category	totalsales
	1	NULL	NULL	29264997.487960815
*	2	NULL	Consulting	7511609.6118469238
	3	NULL	Development	16437797.044464111
	4	NULL	Training	5315590.8316497803
	5	2009-01-01	NULL	734559.35684204102
	6	2009-01-01	Consulting	147329.35925292969
	7	2009-01-01	Development	435158.64923095703
	8	2009-01-01	Training	152071.3483581543

ordermonthyear 가 2009년 1월 1일부터 2월 1일까지 데이터를 그룹화

```
select
    ordermonthyear as OrderMonth,
    productcategory as Category,
    sum(saleamount) as TotalSales
from
    sales_nocomma
where
    ordermonthyear between '2009-01-01' and '2009-02-01'
group by
    ordermonthyear,
    productcategory
with rollup;
```



2019. 2. 25. **Evernote Export** 

## 1-8 cube vs rollup



## 1-9 grouping sets

2009/01/01 ~ 2009/02/01 인 것을 찾아, 그중에 ordermonthyear, productcategory을 그룹화 시킨다.

ordermonthyear, productcategory로 그룹화 된 이들의 ordermonthyear, productcategory 의 각각의 기준의 합계

```
select
    ordermonthyear as OrderMonth,
    productcategory as Category,
    sum(saleamount) as TotalSales
from
    sales_nocomma
where
    ordermonthyear between '2009-01-01' and '2009-02-01'
group by
    ordermonthyear,
    productcategory
GROUPING sets (ordermonthyear, productcategory);
```

### [실행결과]

•	ordermonth	category \$\\$	totalsales
1	NULL	Consulting	292113.04821777344
2	NULL	Development	701459.01047515869
3	NULL	Training	280875.09704589844
4	2009-01-01	NULL	734559.35684204102
5	2009-02-01	NULL	539887.79889678955

# 1-9 grouping sets(2) - 항목이 두개

```
select
    ordermonthyear as OrderMonth,
    productcategory as Category,
    sum(saleamount) as TotalSales
from
    sales_nocomma
where
    ordermonthyear between '2009-01-01' and '2009-02-01'
group by
    ordermonthyear,
    productcategory
-- enhancing
grouping sets
    ((ordermonthyear, productcategory), ordermonthyear, productcategory);
```

## [실행결과]

•	ordermonth \$	category $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$	totalsales
1	NULL	Consulting	292113.04821777344
2	NULL	Development	701459.01047515869
3	NULL	Training	280875.09704589844
4	2009-01-01	NULL	734559.35684204102
5	2009-01-01	Consulting	147329.35925292969
6	2009-01-01	Development	435158.64923095703
7	2009-01-01	Training	152071.3483581543
8	2009-02-01	NULL	539887.79889678955
9	2009-02-01	Consulting	144783.68896484375
10	2009-02-01	Development	266300.36124420166
11	2009-02-01	Training	128803.74868774414

# 1-10 SUBSTRING() 함수를 이용한 년도 추출

```
select *
from
    sales_nocomma
where
    SUBSTRING(orderdate,0,4)='2009'
limit 1000;
```

### [실행결과]

	*	sales_nocomma.rowid #	sales_nocomma.orderid +	sales_nocomma.orderdate +	sales_nocomma.ordern
	1	1914	13729	2009-01-01	2009-01-01
.lıl 🕶	2	4031	28774	2009-01-01	2009-01-01
	3	1279	9285	2009-01-02	2009-01-01
	4	5272	NULL	2009-01-02	2009-01-01

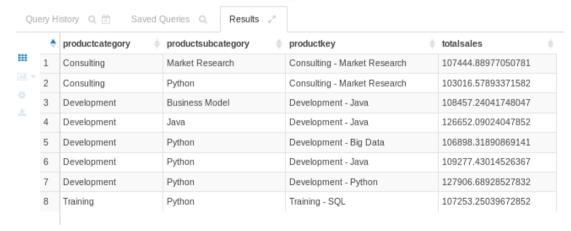
••••

2009년도의 데이터가 확인이 된다.

# 1-11 having 절을 이용한 그룹화 결과 값의 조건 으로 조회하기

2009년도의 주문 날짜 조회 후, productcategory, productsubcategory, productkey를 그룹화 시킨다. 이후, 총 매출 합계가 100000인 것을 뽑는다.

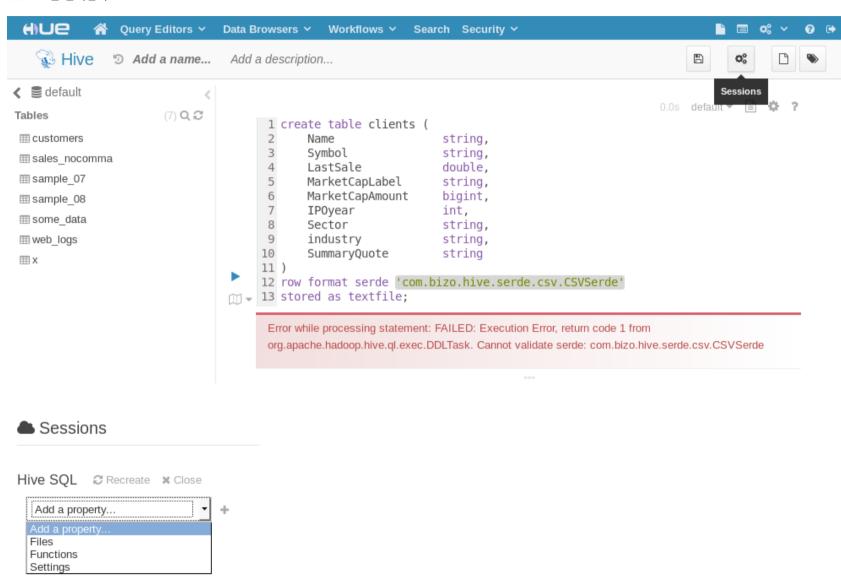
```
select
    productcategory,
    productkey,
    sum(saleamount) as TotalSales
from
    sales_nocomma
where
    orderdate between '2009-01-01' and '2009-12-31'
group by
    productcategory,
    productcategory,
    productsubcategory,
    productkey
having
    sum(saleamount) > 100000
limit 1000;
```



2019. 2. 25.

#### 1-13 client.csv 데이터 셋 준비

Sessions을 선택한다.



Files를 선택 후, clients.csv 파일을 업로드 한다.

# Choose a file

```
Home / user / cloudera

...
2015_11_18
2015_11_19
2015_11_20
2015_11_21
CogsleyServices_SalesData-US_withComma.csv
clients.csv

Upload a file

Select this folder Create folder
```

# 1-14 client.csv 데이터 셋 준비

```
create table clients (
  Name
                 string,
                 string,
  Symbol
  LastSale
                double,
  MarketCapLabel string,
  MarketCapAmount bigint,
  IPOyear
                 int,
  Sector
                string,
                string,
  industry
  SummaryQuote
                     string
row format serde 'com.bizo.hive.serde.csv.CSVSerde'
stored as textfile;
```

