1) Yelp dataset 다운로드(6 json files)

<https://www.yelp.com/dataset>

**yelp\_dataset.tar.gz**

2) AWS 계정 생성(기생성) 및 Acces Key 생성

AWS 로그인 -> 오른쪽상단 계정명 -> 내 보안 자격증명( [My Security Credentials](https://console.aws.amazon.com/iam/home?region=ap-northeast-2#security_credential)) ->

액세스 키(액세스 키 ID 및 보안 액세스 키)

Access keys (access key ID and secret access key) -> 새 액세스키 만들기 -> rootkey.csv파일 다운

3) AWS Configure

- AWS CLI download

- AWS CLI

|  |
| --- |
| $ aws configure  AWS Access Key ID [None] : *Access key ID 입력*  AWS Secret Access Key [None] : Secret *Access key 입력*  Default region name [None] :  Default output format [None] : |

4) S3 bucket, 폴더 생성

- Amazon S3 -> Create bucket -> sk002yelp 버킷 생성

- Json 파일명에 맞는 폴더 생성

5) File Upload

- Json 파일 업로드

|  |
| --- |
| $ aws s3 ls  $ aws s3 cp photos.json s3://sk002yelp/photos  …  $ aws s3 cp review.json s3://sk002yelp/review |

6) Key-pairing

- AWS EC3 dashboard -> Network & Security -> Key Pairs

- Create Key Pair -> 키명 입력후 키 생성 및 다운로드

- ~/.ssh/폴더에 저장

- Ssh keygen

- 서비스 -> EC2 -> 네트워크 및 보안 -> 보안그룹 -> 작업(인바운드규칙편집) -> 규칙추가 -> ssh

7) EMR Cluster 생성

- Amazon EMR -> Create Cluster

- S/W : Hadoop, jupyterhub, hive, hue, spark, zeppelin, tez, sqoop, hcatalog, pig

- h/w ; master 1 instance, core 4 instances, task 0 instance

8) SSH connection 확인

- ssh -i ~/*mykey*.pem hadoop@ec2-13-125-115-186.ap-northeast-2.compute.amazonaws.com

9) Web UI 활성화

- SSH 터널 설정

ssh -i *mykey*.pem -ND 8157 hadoop@ec2-13-125-115-186.ap-northeast-2.compute.amazonaws.com

- Proxy 설정

Google chrome extension으로 foxtyproxy 설치

Foxyproxy-settings.xml 저장

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <foxyproxy>  <proxies>  <proxy name="emr-socks-proxy" id="2322596116" notes="" fromSubscription="false" enabled="true" mode="manual" selectedTabIndex="2" lastresort="false" animatedIcons="true" includeInCycle="true" color="#0055E5" proxyDNS="true" noInternalIPs="false" autoconfMode="pac" clearCacheBeforeUse="false" disableCache="false" clearCookiesBeforeUse="false" rejectCookies="false">  <matches>  <match enabled="true" name="\*ec2\*.amazonaws.com\*" pattern="\*ec2\*.amazonaws.com\*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />  <match enabled="true" name="\*ec2\*.compute\*" pattern="\*ec2\*.compute\*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />  <match enabled="true" name="10.\*" pattern="http://10.\*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />  <match enabled="true" name="\*10\*.amazonaws.com\*" pattern="\*10\*.amazonaws.com\*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />  <match enabled="true" name="\*10\*.compute\*" pattern="\*10\*.compute\*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false" />  <match enabled="true" name="\*.compute.internal\*" pattern="\*.compute.internal\*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false"/>  <match enabled="true" name="\*.ec2.internal\* " pattern="\*.ec2.internal\*" isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false" fromSubscription="false"/>  </matches>  <manualconf host="localhost" port="8157" socksversion="5" isSocks="true" username="" password="" domain="" />  </proxy>  </proxies>  </foxyproxy> |

Foxyproxy -> import -> choose file -> foxyproxy-settings.xml -> open -> add

Proxy mode -> use proxies based on their predefined patterns and priorities

포트 : Hue: 8888, Zeppelin: 8890

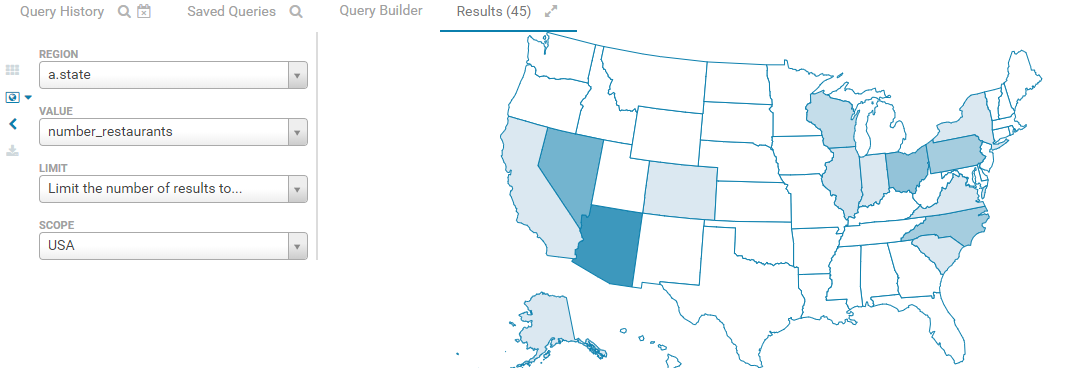
10) Hive 테이블 생성

|  |
| --- |
| create external table users (  user\_id string,  name string,  review\_count int,  yelping\_since string,  friends ARRAY<STRING>,  useful int,  funny int,  cool int,  fans int,  elite ARRAY<INT>,  average\_stars float,  compliment\_hot int,  compliment\_more int,  compliment\_profile int,  compliment\_cute int,  compliment\_list int,  compliment\_note int,  compliment\_plain int,  compliment\_cool int,  compliment\_funny int,  compliment\_writer int,  compliment\_photos int  )  ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'  LOCATION 's3://sk002yelp/user/'    create external table photos (  business\_id string,  photo\_id string,  label string,  caption string  )  ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  --WITH SERDEPROPERTIES ("mapping.postal\_code"="postal code")  LOCATION 's3://sk002yelp/photos/';  create external table checkin (  business\_id string,  time struct<Monday:map<string,int>, Tuesday:map<string,int>, Wednesday:map<string,int>, Thursday:map<string,int>, Friday:map<string,int>, Saturday:map<string,int>, Sunday:map<string,int>>  )  ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  --WITH SERDEPROPERTIES ("mapping.postal\_code"="postal code")  LOCATION 's3://sk002yelp/checkin/';  create external table business (  business\_id string,  name string,  neighborhood string,  address string,  city string,  state string,  postal\_code string,  latitude float,  longitude float,  stars float,  review\_count int,  is\_open int,  attributes map<string,string>,  categories array<string>,  hours struct<Monday:string, Tuesday:string, Wednesday:string, Thursday:string, Friday:string, Saturday:string, Sunday:string>  )  ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  --WITH SERDEPROPERTIES ("mapping.postal\_code"="postal code")  LOCATION 's3://sk002yelp/business/';  create external table tip (  business\_id string,  user\_id string,  tip\_date string,  likes int,  text string  )  ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  WITH SERDEPROPERTIES ("mapping.tip\_date"="date")  LOCATION 's3://sk002yelp/tip/';  CREATE EXTERNAL TABLE review (  review\_id string,  user\_id string,  business\_id string,  stars int,  review\_date string,  text string,  useful int,  funny int,  cool int  )  --ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'  ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'  WITH SERDEPROPERTIES ("mapping.review\_date"="date")  LOCATION 's3://sk002yelp/review/';  -- HDFS에 테이블 생성  CREATE TABLE IF NOT EXISTS review2  STORED AS PARQUET  AS SELECT \* FROM review;  CREATE TABLE IF NOT EXISTS business2  STORED AS PARQUET  AS SELECT \* FROM business;  CREATE TABLE IF NOT EXISTS checkin2  STORED AS PARQUET  AS SELECT \* FROM checkin;  CREATE TABLE IF NOT EXISTS photos2  STORED AS PARQUET  AS SELECT \* FROM photos;  CREATE TABLE IF NOT EXISTS tip2  STORED AS PARQUET  AS SELECT \* FROM tip;  CREATE TABLE IF NOT EXISTS users2  STORED AS PARQUET  AS SELECT \* FROM users; |

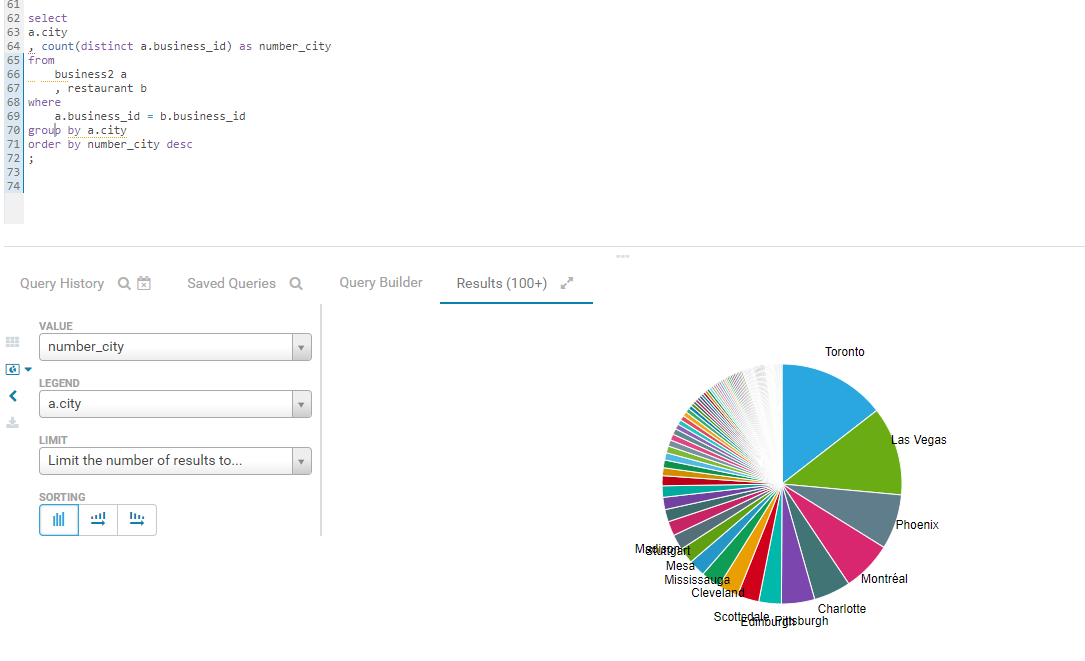
2일차

|  |
| --- |
| create table if not exists category as  select  business\_id  , category  from  business2 lateral view explode(categories) catTable as category  ;  create table restaurant as  select  a.name  , a.review\_count  , a.stars  , b.category  , a.attributes  from  business2 a  , category b  where  a.business\_id = b.business\_id  and b.category = 'Restaurants'  ; |

|  |
| --- |
| select a.state , count(distinct a.business\_id) as number\_restaurants  from business2 a , restaurant b  where a.business\_id = b.business\_id  group by a.state; |



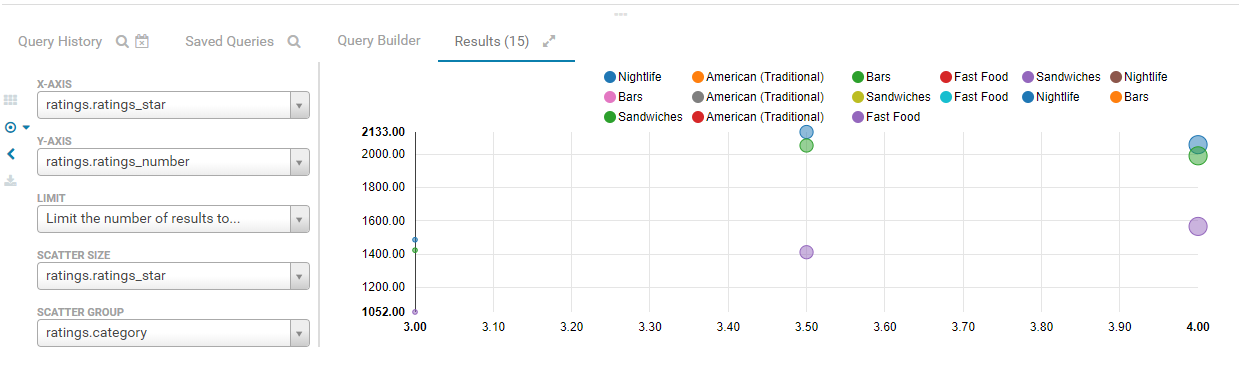
|  |
| --- |
| select  a.city  , count(distinct a.business\_id) as number\_city  from  business2 a  , restaurant b  where  a.business\_id = b.business\_id  group by a.city  order by number\_city desc  ; |



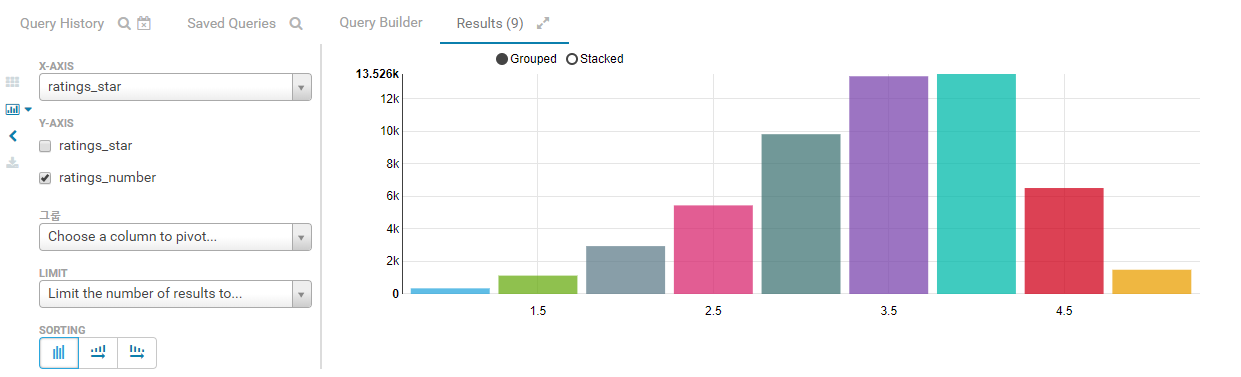
|  |
| --- |
| select  a.category  , count(distinct a.business\_id) as number\_cat  from  category a  , restaurant b  where  a.business\_id = b.business\_id  and a.category not in ('Restaurants','Food')  group by a.category  order by number\_cat desc  limit 15; |



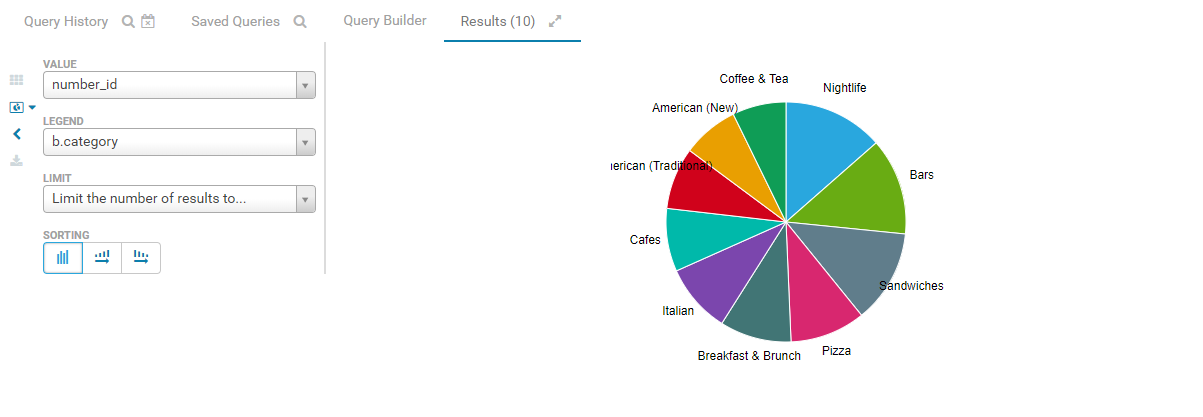
|  |
| --- |
| create table ratings  stored as parquet  as  select  b.category  , a.stars as ratings\_star  , count(distinct a.business\_id) as ratings\_number  from  restaurant a  , category b  where  a.business\_id = b.business\_id  and b.category in ( "Nightlife","Bars", "Sandwiches", "Fast Food","American (Traditional)")  group by b.category  , a.stars  order by a.stars asc, ratings\_number desc  ;  select \* from ratings where ratings\_star between 3 and 4  ; |



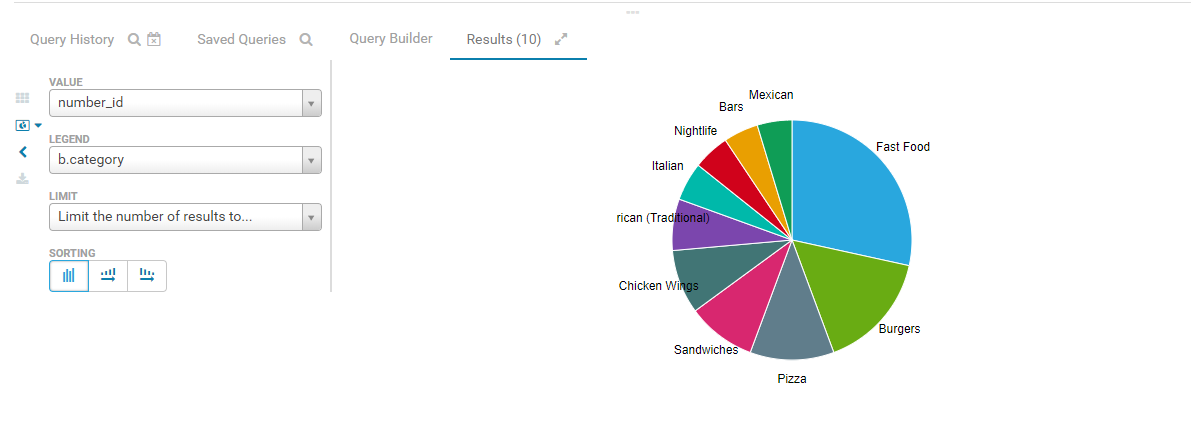
|  |
| --- |
| select  a.stars as ratings\_star  , count(distinct a.business\_id) as ratings\_number  from  restaurant a  , category b  where  a.business\_id = b.business\_id  group by a.stars  ; |



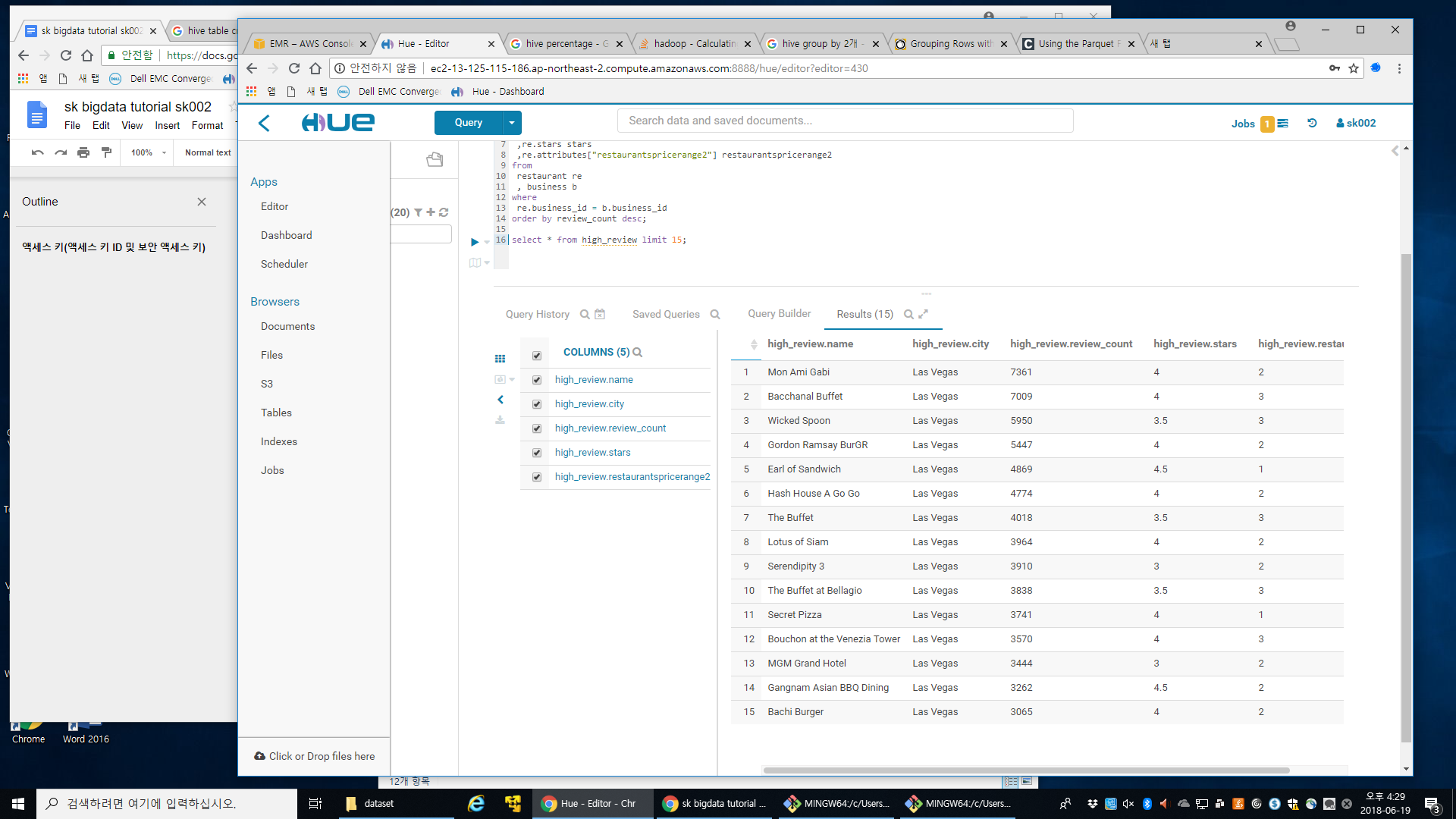
|  |
| --- |
| select  b.category  , count(distinct a.business\_id) number\_id  from  restaurant a  , category b  where  a.business\_id = b.business\_id  and a.stars >= 4  and b.category not in ('Restaurants','Food')  group by b.category  order by number\_id desc  limit 10 |



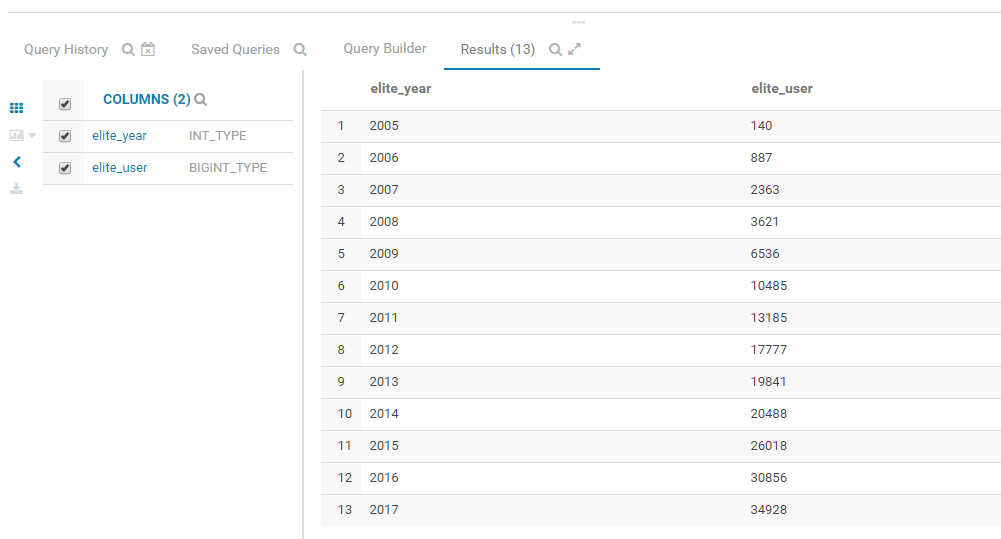
|  |
| --- |
| select  b.category  , count(distinct a.business\_id) number\_id  from  restaurant a  , category b  where  a.business\_id = b.business\_id  and a.stars <= 2  and b.category not in ('Restaurants','Food')  group by b.category  order by number\_id desc  limit 10  ; |



|  |
| --- |
| create table high\_review  stored as parquet  as select  re.name name  ,b.city city  ,re.review\_count review\_count  ,re.stars stars  ,re.attributes["restaurantspricerange2"] restaurantspricerange2  from  restaurant re  , business b  where  re.business\_id = b.business\_id  order by review\_count desc;  select \* from high\_review limit 15; |



|  |
| --- |
| select elite\_year, count(distinct user\_id) as elite\_user from elite\_user  group by elite\_year  order by elite\_year  ; |



|  |
| --- |
| select elite\_year, round(avg(average\_stars),2) as avg\_rating from elite\_user  group by elite\_year  order by elite\_year  ; |



3일차

최종 발표 파일 작성