1. checkin, restaurant 데이터를 활용하여 인기 요일/시간대 분석

\* 소스코드

create table checkin\_detail

stored as parquet as

select

business\_id

, 'monday' as dayofweek

, t1.k as tm

, t1.v as cnt\_checkin

from

checkin2 lateral view explode(time.monday) t1 as k, v

;

insert into checkin\_detail select business\_id, 'tuesday' as dayofweek, t1.k as tm, t1.v as cnt\_checkin from checkin2 lateral view explode(time.tuesday) t1 as k, v;

insert into checkin\_detail select business\_id, 'wednesday' as dayofweek, t1.k as tm, t1.v as cnt\_checkin from checkin2 lateral view explode(time.wednesday) t1 as k, v;

insert into checkin\_detail select business\_id, 'thursday' as dayofweek, t1.k as tm, t1.v as cnt\_checkin from checkin2 lateral view explode(time.thursday) t1 as k, v;

insert into checkin\_detail select business\_id, 'friday' as dayofweek, t1.k as tm, t1.v as cnt\_checkin from checkin2 lateral view explode(time.friday) t1 as k, v;

insert into checkin\_detail select business\_id, 'saturday' as dayofweek, t1.k as tm, t1.v as cnt\_checkin from checkin2 lateral view explode(time.saturday) t1 as k, v;

insert into checkin\_detail select business\_id, 'sunday' as dayofweek, t1.k as tm, t1.v as cnt\_checkin from checkin2 lateral view explode(time.sunday) t1 as k, v;

;

select dayofweek, count(1) from checkin\_detail group by dayofweek

;

-- 요일별

create table dm\_checkin\_dayofweek

stored as parquet as

select

a.name

, b.dayofweek

, sum(b.cnt\_checkin) as number\_checkin

from

restaurant a

, checkin\_detail b

where

a.business\_id = b.business\_id

group by a.name

, b.dayofweek

;

-- 시간대별

create table dm\_checkin\_time

stored as parquet as

select

a.name

, b.tm

, sum(b.cnt\_checkin) as number\_checkin

from

restaurant a

, checkin\_detail b

where

a.business\_id = b.business\_id

group by a.name

, b.tm

;

--결과 테이블

create table result1\_output1

stored as parquet as

select

a.\*

from

(

select

dayofweek

, name

, number\_checkin

, row\_number() over(partition by dayofweek order by number\_checkin desc) as rn\_dayofweek

from

dm\_checkin\_dayofweek

) a

where

a.rn\_dayofweek <= 10

;

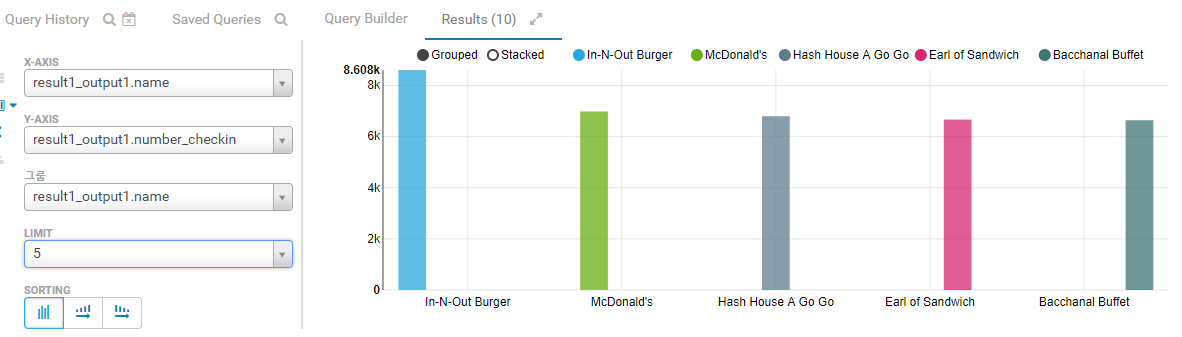
--요일별 checkin 5위까지 출력

select \* from result1\_output1 where rn\_dayofweek <= 5 order by dayofweek, rn\_dayofweek;



-- 일요일에 checkin인기 업체

select \* from result1\_output1 where dayofweek = 'sunday';



-- 일요일 checkin 1위 업체의 시간대별 checkin 현황

select

a.name

, a.tm

, a.number\_checkin

from

dm\_checkin\_time a

, result1\_output1 b

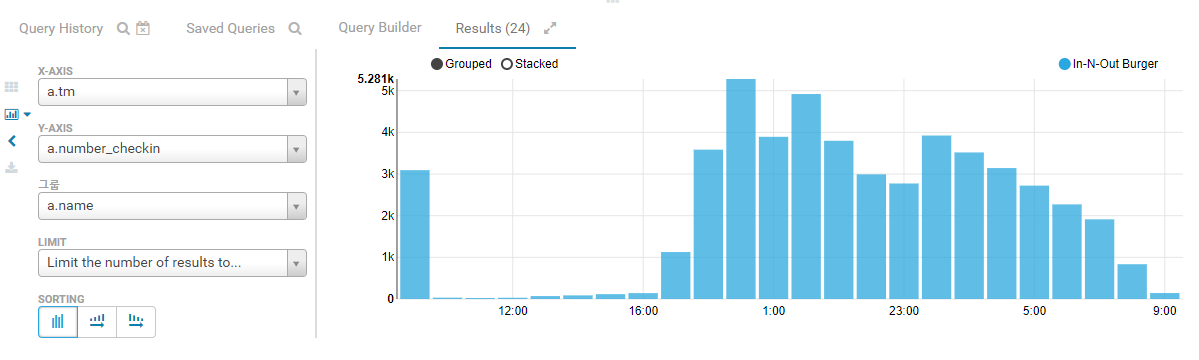
where

a.name = b.name

and b.dayofweek = 'sunday'

and b.rn\_dayofweek = 1

;



2. users 테이블에서 friend가 가장 많은 상위 100명 user의 업체 분석

--user별 friend별 temp 테이블 생성

create table users\_t1

stored as parquet as

select

user\_id

, t1.friend

from

users2 lateral view explode(friends) t1 as friend

;

drop table power\_users purge;

--friend가 많은 상위 100명(power user) 테이블 생성

create table power\_users

stored as parquet as

select

a.user\_id

, b.name

, b.review\_count

, b.useful

, b.cool

, b.funny

, b.fans

, number\_friend

, row\_number() over(order by number\_friend desc) as rn

from

(

select

user\_id

, count(friend) as number\_friend

from

users\_t1

group by user\_id

) a

, users2 b

where

a.user\_id = b.user\_id

limit 100

;

drop table result2\_output purge;

-- power user가 리뷰를 남긴 레스토랑 분석

create table result2\_output

stored as parquet as

select

c.name

, min(b.stars) as min\_stars

, max(b.stars) as max\_stars

, max(c.review\_count) as review\_count

, count(a.user\_id) as number\_user

from

power\_users a

, review2 b

, restaurant c

where

a.user\_id = b.user\_id

and b.business\_id = c.business\_id

group by c.name

;

-- 파워유저가 최저,최고 별점이 4점 이상인 레스토랑 상위 10위

select

name

, max\_stars as stars

, review\_count

, number\_user

from

result2\_output

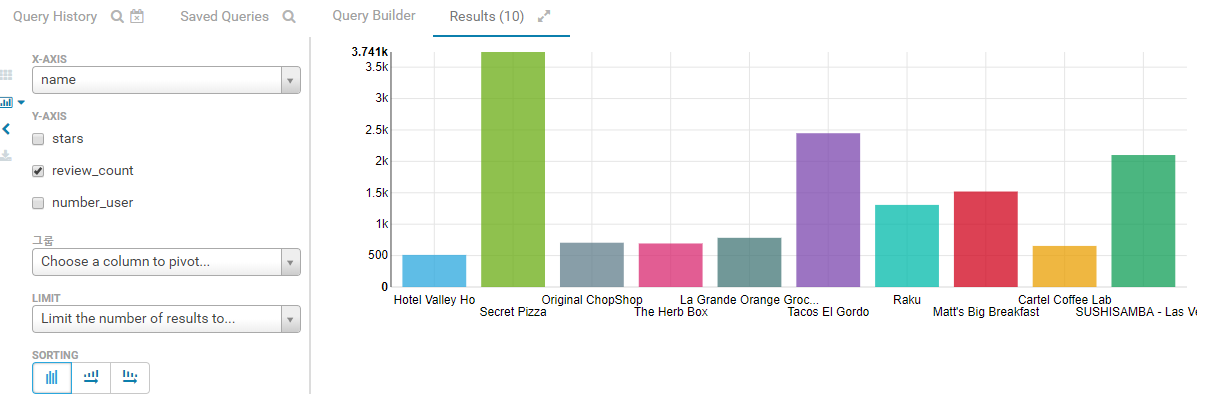
where

max\_stars >= 4

and min\_stars >= 4

order by number\_user desc

limit 10



3. 리뷰의 useful을 사용해서 best 10, worst 10 레스토랑을 보여줌

%pyspark

df\_review = spark.sql("select business\_id, stars, sum(useful) as number\_useful from review2 where stars in (1,5) group by business\_id, stars having sum(useful) >= 1000")

df\_review.registerTempTable("v\_review")

%pyspark

df\_restraunt = spark.sql("select business\_id, name, stars, review\_count from restaurant")

df\_restraunt.registerTempTable("v\_restraunt")

%sql

select

case when t.stars = 1 then '2.Worst'

when t.stars = 5 then '1.Best'

else 'None' end as top

, t.rn

, t.name

, t.number\_useful

, t.restraunt\_stars as avg\_stars

from

(

select

a.stars

, b.stars as restraunt\_stars

, b.name

, a.number\_useful

, row\_number() over(partition by a.stars order by a.number\_useful desc) as rn

from

v\_review a

, v\_restraunt b

where

a.business\_id = b.business\_id

) t

where

t.rn <= 10

order by 1

