

5 장

상품 리뷰 데이터를 이용한 리포트 작성

1 데이터 생성

3장의 1. import CSV - 2) dataset2를 참고해 데이터를 불러온다. 데이터베이스명은 mydata로 생성하고, 테이블은 dataset2로 설정하자.

2 Division별 평점 분포 계산

“박 대리님, 의류 회사에서 상품별 리뷰 데이터를 분석해 달라는 요청을 받았습니다. 상품 평점을 통해 상품의 문제점과 개선 방향을 찾는 것이 이 분석의 주요 목적입니다. 박 대리님이 자유롭게 분석해서 공유해 주세요.”

1) Division별 평균 평점 계산

의류 회사의 상품 리뷰 데이터를 분석해 달라는 요청을 받았다. 먼저 Division별로 평점을 계산해 보고, 어떤 Division의 상품이 좋은 평가를 받는지 또는 좋지 않은 평가를 받는지 살펴보자.

먼저 해당 데이터 세트의 칼럼 구조를 살펴보면 다음과 같다.

- **Clothing ID:** 상품 번호(Unique Value)
- **Age:** 리뷰 작성자의 연령
- **Title:** 리뷰 제목
- **Review Text:** 리뷰 내용
- **Rating:** 리뷰 작성자가 제출한 평점
- **Recommended IND:** 리뷰어에 의한 상품 추천 여부
- **Positive Feedback Count:** 긍정적 피드백 수
- **Division Name:** 상품이 속한 Division
- **Department Name:** 상품이 속한 Department
- **Class Name:** 상품의 타입

Division별 평균 Rating을 계산하려면 먼저 Division Name으로 그룹핑한 뒤, 점수를 평균하면 된다.

a) DIVISION NAME별 평균 평점

```
SELECT 'DIVISION NAME',
AVG(RATING) AVG_RATE
FROM MYDATA.DATASET2
GROUP
BY 1
ORDER
BY 2 DESC
;
```

	DIVISION NAME	AVG RATE
1		5.0000
2	Initmates	4.2950
3	General Petite	4.2089
4	General	4.1726

[그림 5-1 쿼리 실행 결과]

3개 DIVISION 모두 유사한 평점을 갖는 것으로 파악된다. DIVISION NAME에 공란이 있는데 이는 입력값이 없는 것으로 보고 무시하겠다.

다음으로 Department별 평균 평점을 계산해 보자.

b) DEPARTMENT별 평균 평점

```
SELECT 'DEPARTMENT NAME',
AVG(RATING) AVG_RATE
FROM MYDATA.DATASET2
GROUP
BY 1
ORDER
BY 2 DESC
;
```

	DEPARTMENT NAME	AVG RATE
1		5.0000
2	Bottoms	4.2833
3	Intimate	4.2831
4	Jackets	4.2787
5	Tops	4.1691
6	Dresses	4.1468
7	Trend	3.8500

[그림 5-2 쿼리 실행 결과]

Bottoms부터 Dresses까지의 상위 5개 Departments는 유사한 평점을 가진 것으로 보인다. Trend는 3.85점으로 상위 5개 Departments 대비 평점이 매우 낮게 나타나는 것으로 확인된다. Trend에 어떤 문제가 있는지 간단하게 살펴보자.

먼저 Trend의 좋지 않은 리뷰(평점 3점 이하)의 연령별 분포를 살펴본다.

Trend의 3점 이하 리뷰의 연령별 분포를 계산하려면 어떻게 쿼리를 작성해야 할까? 먼저 Department Name을 Trend로 한정해야 하고 RATING은 3점 이하라는 조건을 추가해야 한다.

c) Trend의 평점 3점 이하 리뷰

```
SELECT *
FROM MYDATA.DATASET2
WHERE 'DEPARTMENT NAME' = 'Trend'
AND RATING <= 3
;
```

	rating	recommended	IND	Positive Feedback Count	Division Name	Department name
1	3	1		0	General	Trend
2	3	0		3	General Petite	Trend
3	3	1		10	General	Trend
4	3	0		4	General Petite	Trend
5	3	0		2	General Petite	Trend
6	1	0		0	General	Trend
7	2	0		2	General	Trend
8	3	1		0	General	Trend
9	1	0		0	General	Trend
10	3	1		4	General	Trend

[그림 5-3 쿼리 실행 결과]

쿼리를 실행한 결과를 보면, Rating은 3점 이하이면서 Department Name은 Trend인 데이터가 모두 조회된다. 우리가 구하고 위의 조건(Trend의 3점 이하 평점)을 만족하는 리뷰의 연령별 분포다.

따라서 위의 데이터 세트에서 연령으로 데이터를 그룹핑하고 카운트로 집계하면, 우리가 원하는 결과를 얻을 수 있다. 여기서 연령은 어떻게 계산할 수 있을까?

구간을 그룹으로 나누어 집계하는 방법에는 크게 2가지 방법이 있다. 연령을 예로 들어 보면 다음과 같다.

2) case when

case when 구문을 사용해 우리가 원하는 그룹을 생성할 수 있다. 예를 들어 연령을 10세 단위로 그룹핑한다면 다음과 같이 할 수 있다.

```
SELECT CASE WHEN AGE BETWEEN 0 AND 9 THEN '0009'  
WHEN AGE BETWEEN 10 AND 19 THEN '1019'  
WHEN AGE BETWEEN 20 AND 29 THEN '2029'  
WHEN AGE BETWEEN 30 AND 39 THEN '3039'  
WHEN AGE BETWEEN 40 AND 49 THEN '4049'  
WHEN AGE BETWEEN 50 AND 59 THEN '5059'  
WHEN AGE BETWEEN 60 AND 69 THEN '6069'  
WHEN AGE BETWEEN 70 AND 79 THEN '7079'  
WHEN AGE BETWEEN 80 AND 89 THEN '8089'  
WHEN AGE BETWEEN 90 AND 99 THEN '9099' END AGEBAND,  
AGE  
FROM MYDATA.DATASET2  
WHERE 'DEPARTMENT NAME' = 'Trend'  
AND RATING <= 3  
;
```

	AGEBAND	AGE
1	60	66
2	30	36
3	60	66
4	30	34
5	40	41
6	50	53
7	30	36
8	30	39
9	50	51
10	50	56

[그림 5-4 쿼리 실행 결과]

AGE와 AGEBAND를 살펴보면, 우리가 의도한 대로 연령대가 생성된 것을 확인할 수 있다. 위의 방법은 하나씩 연령 구간을 설정해야 한다는 점에서 번거로움이 있다. 다음과 같은 방법을 사용하면, 10세 단위로 연령을 쉽게 나눌 수 있다.

3) FLOOR

연령을 10으로 나눈 값을 버림 하면 어떤 값을 얻을 수 있을까? 예를 들어 23을 10으로 나누면 2.3이라는 값을 얻게 된다. 이 값의 소수점을 버림 하면 2가 나오고, 여기에 10을 곱하면 20이라는 값이 계산된다. 23세의 연령대인 20을 구할 수 있다.

다른 값도 마찬가지다. 57을 10으로 나눈 뒤 소수점을 버림 하면 5라는 값이 되고, 여기에 10을 곱하면 50이라는 값을 얻을 수 있다.

```
SELECT FLOOR(AGE/10)*10 AGEband,
      AGE
     FROM MYDATA.DATASET2
    WHERE 'DEPARTMENT NAME' = 'Trend'
      AND RATING <= 3
;
```

	AGEBAND	AGE
1	60	66
2	30	36
3	60	66
4	30	34
5	40	41
6	50	53
7	30	36
8	30	39
9	50	51
10	50	56

[그림 5-5 쿼리 실행 결과]

CASE WHEN을 사용하는 것보다 훨씬 간단한 방법으로 10세 단위로 연령을 나눌 수 있게 되었다. 그렇다면 이 값으로 데이터를 그룹핑하고 COUNT로 집계한다면, 연령별 리뷰 수를 구할 수 있지 않을까?

a) Trend의 평점 3점 이하 리뷰의 연령 분포

```
SELECT FLOOR(AGE/10)*10 AGEBAND,
COUNT(*) CNT
FROM MYDATA.DATASET2
WHERE 'DEPARTMENT NAME' = 'Trend'
AND RATING <= 3
GROUP
BY 1
ORDER
BY 2 DESC;
```

	AGEBAND	CNT
1	50	10
2	40	9
3	30	6
4	60	4
5	20	4
6	70	1

[그림 5-6 쿼리 실행 결과]

50대에서 3점 이하의 평점 수가 가장 많은 것으로 확인된다. 그렇다고 50대가 Trend라는 Department에 가장 많은 불만이 있다고 할 수 있을까? 만약 Trend의

리뷰 중 50대 리뷰가 가장 많다면, 위 결과는 당연한 말이 되지 않을까?

그렇기 때문에 Trend의 전체 연령별 리뷰 수를 구해 보아야 한다.

b) Department별 연령별 리뷰 수

```
SELECT FLOOR(AGE/10)*10 AGEBAND,  
COUNT(*) CNT  
FROM MYDATA.DATASET2  
WHERE 'Department name' = 'Trend'  
GROUP  
BY 1  
ORDER  
BY 2 DESC  
;
```

AGEBAND	CNT
1	30
2	40
3	50
4	20
5	60
6	80
7	70

[그림 5-7 쿼리 실행 결과]

Trend의 전체 리뷰 수를 보면 30, 40, 50대 순으로 리뷰 수가 많은 것으로 확인된다. 이를 종합해 보면, 50대의 Trend에 대한 평점이 다소 좋지 않은 것으로 생각할 수 있다. 연령별 3점 이하의 리뷰 수를 비중으로 구한다면, 더 명확하게 결과를 확인할 수 있을 것이다. 50대의 3점 이하 Trend 리뷰를 몇 가지 살펴보자.

c) 50대 3점 이하 Trend 리뷰

```
SELECT *
FROM MYDATA.DATASET2
WHERE 'Department name' = 'Trend'
AND RATING <= 3
AND AGE BETWEEN 50 AND 59 LIMIT 10
;
```

Clothing ID	Age	Title	Review Text
1135	53		
1145	51	Beyond boxy	I am not sure who would look good in this dress. it is extremely
570	56	Fun fabric	The print is unique and fun. however, the longer length and heavy
552	53	You don't know what you get...	Like a previous reviewer mentioned the jeans do not necessarily
569	55	Gorgeous lace top, design fail	I recommend this topper but with reservations. it is beautiful look
569	58	Fly away sides	I waited a long time to get this top. unfortunately i will be returnir
569	53	Pretty bad in real life	When i unwrapped this a chemical "stink"" hit me that amost kno
1146	52	Disappointed	I have purchased a lot from retailer but lately, some of the items
1142	54	Neck	The dress is pretty, i ordered a 4 and 6 not knowing how it ran, i
567	57	Frompy	I so wanted to like this dress i ordered it for summer graduation

[그림 5-8 쿼리 실행 결과]

I am not sure who would look good in this dress. it is extremely oversized with tons of fabric that will not flatter anyone!

Like a previous reviewer mentioned the jeans do not necessarily look like the one in the photo. I first ordered them in my regular size and also had to notice that they are cut like men's jeans with legs so skinny that I could only force them over my calves. the embroidery is beautiful, but it also stops the material from stretching, so the super skinny legs are a real problem. also, they are not as intensely stone washed as shown in the picture. I decided to order them again one size bigger.

위 리뷰를 살펴보면, 사이즈에 관한 complain이 존재함을 알 수 있다. 추후 사이즈와 관련된 리뷰를 좀 더 깊게 살펴볼 필요가 있어 보인다.

3 평점이 낮은 상품의 주요 Complain

먼저 Department별로 평점이 낮은 주요 10개 상품을 조회한 후, 해당 상품들의 리뷰를 살펴보자. Department별 평점이 낮은 10개 상품을 임시 테이블로 생성한다.

1) Department Name, Clothing Name별 평균 평점 계산

```
SELECT 'DEPARTMENT NAME',  
       'CLOTHING ID',  
       AVG(RATING) AVG_RATE  
  FROM MYDATA.DATASET2  
 GROUP  
 BY 1,2  
 ;
```

	DEPARTMENT NAME	CLOTHING ID	AVG RATE
1	Intimate	767	4.5000
2	Dresses	1080	4.2683
3	Dresses	1077	4.0706
4	Bottoms	1049	4.4000
5	Tops	847	4.0000
6	Tops	858	3.9500
7	Dresses	1095	4.0330
8	Bottoms	1065	3.7500
9	Tops	853	3.6667
10	Jackets	1120	4.5000

[그림 5-9 쿼리 실행 결과]

2) Department별 순위 생성

Department, Clothing id의 평균 평점을 계산하고, Department 내에서 평균 평점을 기준으로 순위를 매긴다.

```

SELECT *,  

ROW_NUMBER() OVER( PARTITION BY 'DEPARTMENT NAME'  

ORDER BY AVG_RATE) RNK  

FROM  

(SELECT 'DEPARTMENT NAME',  

'CLOTHING ID',  

AVG(RATING) AVG_RATE  

FROM MYDATA.DATASET2  

GROUP  

BY 1,2) A  

;

```

	DEPARTMENT NAME	CLOTHING ID	AVG RATE	RNK
1	Bottoms	18	1.00	1
2	Bottoms	588	2.00	2
3	Bottoms	1039	2.40	3
4	Bottoms	495	2.50	4
5	Bottoms	1058	2.50	5
6	Bottoms	450	2.50	6
7	Bottoms	1006	2.60	7
8	Bottoms	453	2.75	8
9	Bottoms	603	3.00	9
10	Bottoms	63	3.00	10

[그림 5-10 쿼리 실행 결과]

	DEPARTMENT NAME	CLOTHING ID	AVG RATE	RNK
1	Dresses	1195	2.0000	1
2	Dresses	1152	2.0000	2
3	Dresses	1200	3.0000	3
4	Dresses	1073	3.5000	4
5	Dresses	1084	3.5156	5
6	Dresses	1088	3.5556	6
7	Dresses	1079	3.5962	7
8	Dresses	1102	3.6000	8
9	Dresses	1076	3.6744	9
10	Dresses	1075	3.7069	10

[그림 5-11 쿼리 실행 결과]

계산 결과를 보면, 각 Department별로 평점(AVG_RATE)이 가장 낮은 순서대로 순위가 매겨짐을 볼 수 있다. 이제 해당 데이터 세트에서 RNK 값이 1~10위 사이인 데이터를 가지고 오면, 각 Department별로 평점이 좋지 않은 10개의 상품을 조회할 수 있다.

3) 1~10위 데이터 조회

```
SELECT *
FROM
(SELECT *,  
ROW_NUMBER() OVER( PARTITION BY 'DEPARTMENT NAME' ORDER BY AVG_  
RATE) RNK  
FROM  
(SELECT 'DEPARTMENT NAME',  
'CLOTHING ID',  
AVG(RATING) AVG_RATE  
FROM DATA.DATASET2  
GROUP  
BY 1,2) A) A  
WHERE RNK <= 10  
;
```

	DEPARTMENT NAME	CLOTHING ID	AVG RATE	RNK
1		72	5.00	1
2		492	5.00	2
3		152	5.00	3
4		184	5.00	4
5		772	5.00	5
6		136	5.00	6
7	Bottoms	18	1.00	1
8	Bottoms	588	2.00	2
9	Bottoms	1039	2.40	3
10	Bottoms	495	2.50	4
11	Bottoms	1058	2.50	5
12	Bottoms	450	2.50	6
13	Bottoms	1006	2.60	7
14	Bottoms	453	2.75	8
15	Bottoms	603	3.00	9
16	Bottoms	63	3.00	10
17	Dresses	1195	2.00	1
18	Dresses	1152	2.00	2
19	Dresses	1200	3.00	3
20	Dresses	1073	3.50	4

[그림 5-12 쿼리 실행 결과]

이제 해당 데이터를 테이블로 생성한다.

a) Department별 평균 평점이 낮은 10개 상품

```
CREATE TEMPORARY TABLE DATA.STAT AS
SELECT *
FROM
(SELECT *,  
ROW_NUMBER() OVER( PARTITION BY 'DEPARTMENT NAME' ORDER BY AVG_  
RATE) RNK
FROM
(SELECT 'DEPARTMENT NAME',  
'CLOTHING ID',  
AVG(RATING) AVG_RATE  
FROM MYDATA.DATASET2
GROUP
BY 1,2) A) A
WHERE RNK <= 10;
```

이제 생성한 테이블을 이용하면, 각 Department별로 평점이 낮은 상품들의 Complain 내용을 확인할 수 있다. 예를 들어 Bottoms의 평점이 낮은 10개 상품의 리뷰를 조회해 보겠다. 다음과 같은 쿼리를 실행하면, Bottoms의 평점이 낮은 10개 상품의 Clothing ID를 조회할 수 있다.

```
SELECT 'CLOTHING ID'
FROM MYDATA.STAT
WHERE 'DEPARTMENT NAME' = 'Bottoms';
```

CLOTHING ID	
1	18
2	588
3	1039
4	495
5	1058
6	450
7	1006
8	453
9	603
10	63

[그림 5-13 쿼리 실행 결과]

위의 CLOTHING ID에 해당하는 리뷰 내용을 조회하는 방법은 다음과 같다.

```
SELECT *
FROM MYDATA.DATASET2
WHERE 'CLOTHING ID' IN
(SELECT 'CLOTHING ID'
FROM MYDATA.STAT
WHERE 'DEPARTMENT NAME' = 'Bottoms')
ORDER
BY 'CLOTHING ID'
```

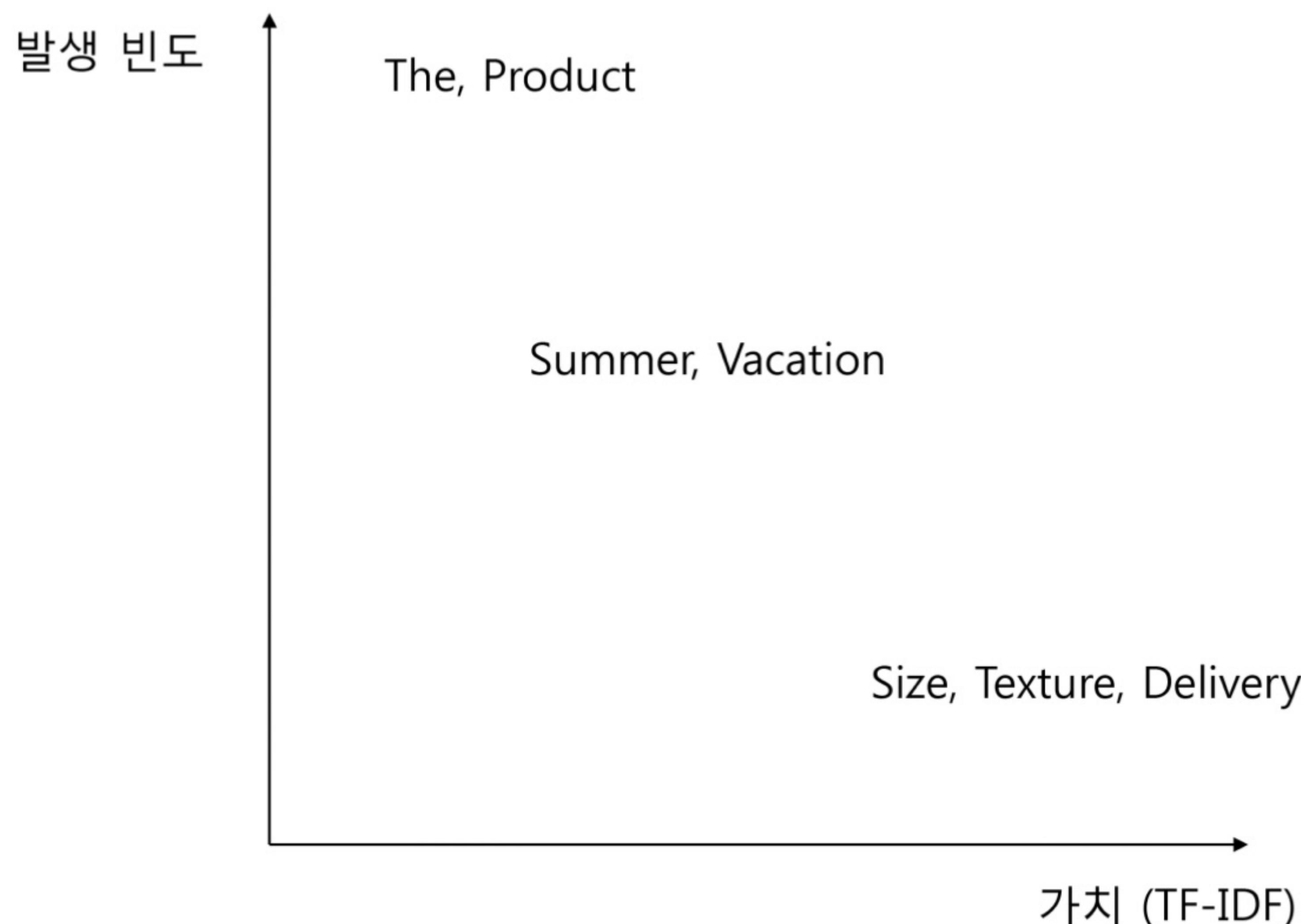
Bottoms의 평점이 낮은 상품의 리뷰 내용을 몇 가지 살펴보자.

These shorts are just beautiful. with that said, they were expensive, but made reasonable when I snagged them in the online sale! they do also run large and they seem to be made for short-waisted bodies. I will be bringing them back to my local store to (hopefully) exchange for a smaller size. fingers crossed that they still have them in stock!

I was so excited about these pants but they were a total disappointment in person. the fabric is a rough gauzy texture. the bottom has a big hem that weighs down the pants and makes them hang funny like you are walking on stilts or something, they do not hang or move well on the body. also, they are very long but small in the waist and have a high crotch so the fit is pretty off. sadly, these are going back.

역시 사이즈 및 소재에 대한 내용이 다수 확인된다.

b) TF-IDF



[그림 5-14 TF-IDF]

리뷰 데이터는 수많은 단어로 구성되어 있다. 문단에는 the, product와 같이 자주 사용되지만 가치가 없는 단어들도 있고, Size, Textured와 같이 평가 내용을 파악하는 데 도움이 되는 가치 있는 단어들도 있다.

이를 판단하기 위해 NLP(Natural Language Processing)에서 TF-IDF라는 Score를 이용해 단어별로 가치 수준을 매긴다. TF-IDF는 R, Python을 통해 계산할 수 있다.

4 연령별 Worst Department

리뷰 데이터를 기반으로 프로모션을 진행한다고 가정해 보자. 먼저 연령대별로 가장 낮은 점수를 준 Department를 구하고, 해당 Department의 할인 쿠폰을 발송하

기로 한다.

연령별로 가장 낮은 점수를 준 Department가 구해지면, 연령별로 가장 낮은 점수를 준 Department에 혜택을 준다.

이를 구하려면 어떻게 해야 할까? 과정은 다음과 같다.

- 연령, Department별 가장 낮은 점수 계산
- 생성한 점수를 기반으로 Rank 생성
- Rank 값이 1인 데이터를 조회

우선 연령, Department별 가장 낮은 점수를 계산한다.

Department Name과 연령으로 그룹핑한 뒤, 평점을 평균하면 된다. 이때 연령은 앞에서 구한 것처럼 10세 단위로 나눈다.

```
SELECT 'DEPARTMENT NAME',
FLOOR(AGE/10)*10 AGEBOARD,
AVG(RATING) AVG_RATING
FROM MYDATA.DATASET2
GROUP
BY 1,2
```

	DEPARTMENT NAME	AGEBOARD	AVG_RATING
1	Intimate	30	4.2589
2	Dresses	30	4.1208
3	Dresses	60	4.2386
4	Bottoms	50	4.3565
5	Tops	40	4.1014
6	Dresses	40	4.1193
7	Tops	30	4.1491
8	Dresses	20	4.1725
9	Dresses	50	4.1818
10	Intimate	40	4.2830

[그림 5-15 쿼리 실행 결과]

다음으로 연령별로 생성한 점수를 기준으로 Rank를 계산한다. Rank를 생성할 때는 가장 낮은 점수가 1위가 되도록 오름차순으로 정렬한다. 우리는 여기서 ROW_NUMBER를 이용하여 RANK를 생성하겠다.

```
SELECT *,  
ROW_NUMBER() OVER(PARTITION BY AGEBOARD ORDER BY AVG_RATING) RNK  
FROM  
(SELECT 'DEPARTMENT NAME',  
FLOOR(AGE/10)*10 AGEBOARD,  
AVG(RATING) AVG_RATING  
FROM MYDATA.DATASET2  
GROUP  
BY 1,2) A
```

	DEPARTMENT NAME	AGEBOARD	AVG_RATING	RNK
1	Intimate	10	3.3333	1
2	Tops	10	4.1667	2
3	Dresses	10	4.3846	3
4	Bottoms	10	4.5000	4
5	Jackets	10	5.0000	5
6	Trend	20	3.8333	1
7	Jackets	20	4.1504	2
8	Tops	20	4.1637	3
9	Dresses	20	4.1725	4
10	Bottoms	20	4.2986	5

[그림 5-16 쿼리 실행 결과]

연령별로 평균 평점 점수가 가장 낮은 Department가 Rank 값 1을 갖게 된다.

Department Name	ageband	avg_rating	rnk
tops	10	3	2
bottoms	10	1	1
dresses	10	5	5
intimate	10	4	3
jackets	10	4	4
tops	20	5	5
bottoms	20	2	1
dresses	20	4	2
intimate	20	4	3
jackets	20	5	4

[표 5-1 연령별 평균 평점 순위]

예를 들어, 쿼리 결과가 위와 같이 나온다면, 10대의 가장 낮은 평균 평점을 받은 Department는 Bottms, 20대의 가장 낮은 평균 평점을 받은 Department는 Bottoms 가 된다.

이제 Rank 값이 1인 값을 조회하면, 연령별로 가장 낮은 평점을 준 Department를 찾을 수 있다.

```
SELECT *
FROM
(SELECT *,  
ROW_NUMBER() OVER(PARTITION BY AGEBOARD ORDER BY AVG_RATING) RNK  
FROM  
(SELECT 'DEPARTMENT NAME',  
FLOOR(AGE/10)*10 AGEBOARD,  
AVG(RATING) AVG_RATING  
FROM MYDATA.DATASET2  
GROUP  
BY 1,2) A) A  
WHERE RNK = 1;
```

	DEPARTMENT NAME	AGEBOARD	AVG_RATING	RNK
1	Intimate	10	3.3333	1
2	Trend	20	3.8333	1
3	Dresses	30	4.1208	1
4	Trend	40	3.8750	1
5	Trend	50	3.3043	1
6	Trend	60	3.8000	1
7	Trend	70	2.0000	1
8	Tops	80	4.4222	1
9	Dresses	90	3.2857	1

[그림 5-17 쿼리 실행 결과]

5 Size Complain

앞에서 살펴본 내용에 따르면, Complain 내용의 다수가 Size와 관련된 문제였다. 먼저 전체 리뷰 내용 중 Size와 관련된 리뷰가 얼마나 되는지 확인해 보자.

먼저 전체 리뷰의 수와 Size가 언급된 리뷰의 수가 몇 개나 되는지 확인해 보겠다. 즉 다음과 같이 Review Text의 내용 중 size라는 단어가 언급된 Review가 몇 개인지 계산해야 한다.

Review Text 샘플

'I had such high hopes for this dress and really wanted it to work for me. I initially ordered the petite small (my usual size) but I found this to be outrageously small, so small in fact that I could not zip it up! I reordered it in petite medium, which was just ok. overall, the top half was comfortable and fit nicely, but the bottom half had a very tight under layer and several somewhat cheap (net) over layers. imo, a major design flaw was the net over layer sewn directly into the zipper - it c.'

'I added this in my basket at the last minute to see what it would look like in person. (store pick up). I went with the darkle color only because I am so pale :-) the color is really gorgeous, and turns out it matched everything I was trying on with it perfectly. it is a little baggy on me and the xs is the mallet size (bummer, no petite). I decided to jeep it though, because as I said, it matvehd everything my jeans, pants, and the 3 skirts I was trying on (of which I kept all) oops. '

'I ordered this in carbon for store pick up, and had a ton of stuff (as always) to try on and used this top to pair (skirts and pants). everything went with it. the color is really nice charcoal with shimmer and went well with pencil skirts, flare

pants, etc. my only compaint is it is a bit big, sleeves are long and it doesn't go in petite. also a bit loose for me but no xxS... so I kept it and will decide later since the light color is already sold out in the smallest size... '

Size가 포함된 리뷰의 수를 구하는 방법에는 여러 가지가 있을 수 있다. 그중에서 우리는 집계 함수와 CASE WHEN 구문을 이용해 구해 보겠다.

```
SELECT 'REVIEW TEXT',
CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END SIZE_YN
FROM MYDATA.DATASET2
;
```

위 쿼리를 실행하면 다음과 같은 결과가 출력된다.

review_text	size_yn
... absolutely wonderful ...	0
... I happened to find it in a store ...	0
... the petite small (my usual size) but I found this to be	1
... I love, love, love this jumpsuite ...	0

[표 5-2 쿼리 실행 결과]

REVIEW_TEXT, SIZE_YN 칼럼 2가지가 출력된다. REVIEW_TEXT는 리뷰의 내용이고, SIZE_YN은 리뷰 내용 중 size가 포함되어 있으면 1, 그렇지 않으면 0인 값이 출력되는 칼럼이다.

여기서 SIZE_YN을 더한 값은 무엇이 될까? 바로 Review text 중 size가 포함된 리뷰의 수가 될 것이다. 전체의 리뷰 수는 COUNT(*)로 구할 수 있다. 다음과 같이 구할 수 있다.

```
SELECT SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END ) N_SIZE,  
COUNT(*) N_TOTAL  
FROM MYDATA.DATASET2  
;
```

	N_SIZE	N_TOTAL
1	6133	20603

[그림 5-18 쿼리 실행 결과]

전체 리뷰 내용 중 약 30%가량이 Size와 관련된 리뷰였다. 다음으로 사이즈를 Large, Loose, Small, Tight로 상세히 나누어 살펴보자.

```
SELECT SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END ) N_SIZE,  
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LARGE%' THEN 1 ELSE 0 END ) N_LARGE,  
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LOOSE%' THEN 1 ELSE 0 END ) N_LOOSE,  
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SMALL%' THEN 1 ELSE 0 END ) N_SMALL,  
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%TIGHT%' THEN 1 ELSE 0 END ) N_TIGHT,  
SUM(1) N_TOTAL  
FROM MYDATA.DATASET2  
;
```

	N_SIZE	N_LARGE	N_LOOSE	N_SMALL	N_TIGHT	N_TOTAL
1	6133	2508	1105	3432	1445	20603

[그림 5-19 쿼리 실행 결과]

Large, Loose와 비교해 Small, Tight와 관련된 리뷰가 더 많은 것으로 확인된다. 이제 카테고리별로 해당 수치들을 확인해 보자.

```

SELECT 'Department Name',
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END ) N_SIZE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LARGE%' THEN 1 ELSE 0 END ) N_LARGE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LOOSE%' THEN 1 ELSE 0 END ) N_LOOSE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SMALL%' THEN 1 ELSE 0 END ) N_SMALL,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%TIGHT%' THEN 1 ELSE 0 END ) N_TIGHT,
SUM(1) N_TOTAL
FROM MYDATA.DATASET2
GROUP
BY 1
;

```

	Department Name	N_SIZE	N_LARGE	N_LOOSE	N_SMALL	N_TIGHT	N_TOTAL
1	Intimate	412	171	58	296	120	1558
2	Dresses	1669	697	294	900	503	5353
3	Bottoms	1145	296	143	484	265	3230
4	Tops	2576	1200	570	1527	494	9427
5	Jackets	288	126	35	199	52	922
6	Trend	40	17	5	25	9	100
7		3	1	0	1	2	13

[그림 5-20 쿼리 실행 결과]

Dresses, Bottoms, Tops에서 사이즈와 관련된 리뷰가 많은 것으로 확인되고 Dresses, Bottoms는 Tops와 비교해 Small, Tight와 관련된 리뷰가 많았다. 다음으로 이를 연령별로 나누어 구해 본다.

```

SELECT FLOOR(AGE/10)*10 AGEBAND,
'Department Name',
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END ) N_SIZE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LARGE%' THEN 1 ELSE 0 END ) N_LARGE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LOOSE%' THEN 1 ELSE 0 END ) N_LOOSE,

```

```

SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SMALL%' THEN 1 ELSE 0 END ) N_
SMALL,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%TIGHT%' THEN 1 ELSE 0 END ) N_
TIGHT,
SUM(1) N_TOTAL
FROM MYDATA.DATASET2
GROUP
BY 1,2
ORDER
BY 1,2
;

```

AGEBAND	Department Name	N_SIZE	N_LARGE	N_LOOSE	N_SMALL	N_TIGHT	N_TOTAL
1	10	Bottoms	3	2	1	1	8
2	10	Dresses	2	0	0	4	2
3	10	Intimate	2	0	0	1	1
4	10	Jackets	0	0	0	0	1
5	10	Tops	5	1	1	3	1
6	20		1	0	0	1	1
7	20	Bottoms	126	38	17	58	33
8	20	Dresses	232	101	40	136	66
9	20	Intimate	60	26	12	45	23
10	20	Jackets	33	14	3	27	7
							113

[그림 5-21 쿼리 실행 결과]

단순히 리뷰의 수를 계산하게 되면 Department에서 Size와 관련된 주된 Complain 내용이 무엇인지 파악하기가 힘들다. 절대 수가 아닌 비중을 구한다면 더 직관적으로 Complain의 분포를 파악할 수 있을 것이다.

AGEBAND	Department Name	N_SIZE	N_LARGE	N_LOOSE	N_SMALL	N_TIGHT	N_TOTAL
1	10	Bottoms	3	2	1	1	8
2	10	Dresses	2	0	0	4	2
3	10	Intimate	2	0	0	1	1
4	10	Jackets	0	0	0	0	1
5	10	Tops	5	1	1	3	1
6	20		1	0	0	1	1
7	20	Bottoms	126	38	17	58	33
8	20	Dresses	232	101	40	136	66
9	20	Intimate	60	26	12	45	23
10	20	Jackets	33	14	3	27	7
							113

[그림 5-22 쿼리 실행 결과]

가장 우측의 N_TOTAL은 연령별, Department별 총 리뷰 수이다. 총 리뷰 수로 각 칼럼을 나누면, 각 그룹에서 size 세부 그룹의 비중을 구할 수 있다.

```

SELECT FLOOR(AGE/10)*10 AGEBAND,
'Department Name',
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END )/
SUM(1) N_SIZE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LARGE%' THEN 1 ELSE 0 END )/
SUM(1) N_LARGE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LOOSE%' THEN 1 ELSE 0 END )/
SUM(1) N_LOOSE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SMALL%' THEN 1 ELSE 0 END )/
SUM(1) N_SMALL,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%TIGHT%' THEN 1 ELSE 0 END )/
SUM(1) N_TIGHT
FROM MYDATA.DATASET2
GROUP
BY 1,2
ORDER
BY 1,2
;

```

AGEBAND	Department Name	N_SIZE	N_LARGE	N_LOOSE	N_SMALL	N_TIGHT
1 10	Bottoms	0.3750	0.2500	0.1250	0.1250	0.1250
2 10	Dresses	0.1538	0.0000	0.0000	0.3077	0.1538
3 10	Intimate	0.3333	0.0000	0.0000	0.1667	0.1667
4 10	Jackets	0.0000	0.0000	0.0000	0.0000	0.0000
5 10	Tops	0.4167	0.0833	0.0833	0.2500	0.0833
6 20		0.5000	0.0000	0.0000	0.5000	0.5000
7 20	Bottoms	0.3452	0.1041	0.0466	0.1589	0.0904
8 20	Dresses	0.3102	0.1350	0.0535	0.1818	0.0882
9 20	Intimate	0.2390	0.1036	0.0478	0.1793	0.0916
10 20	Jackets	0.2920	0.1239	0.0265	0.2389	0.0619

[그림 5-23 쿼리 실행 결과]

총 리뷰 수를 SUM(1)과 같은 방법으로 계산했는데, 이는 COUNT(*)과 동일한 결과를 출력한다.

COUNT(*)은 데이터 ROW의 개수를 구하는 집계 함수였다. SUM(1)은 ROW마다 1이라는 수를 생성하고, 이를 모두 합한 결과를 의미해, 결론적으로 COUNT(*)과 SUM(1)은 동일한 결과를 출력하게 된다. 1을 SELECT에 입력하고 실행한 결과는 다음과 같다. 다음의 1이라는 칼럼을 모두 합한 결과는 COUNT(*)과 같게 된다.

	AGEBAND	Department Name	1
1	30	Intimate	1
2	30	Dresses	1
3	60	Dresses	1
4	50	Bottoms	1
5	40	Tops	1
6	40	Dresses	1
7	30	Tops	1
8	30	Tops	1
9	20	Dresses	1
10	50	Dresses	1

[그림 5-24 쿼리 실행 결과]

6 Clothing ID별 Size Review

연령, Department별로 사이즈와 관련된 리뷰를 살펴보았다. 이제 좀 더 세부적으로 내용을 살펴보고자 한다. 어떤 상품이 Size와 관련된 리뷰 내용이 많은지 살펴보자.

먼저 상품 ID(Clothing ID)별로 사이즈와 관련된 리뷰 수를 계산하고, 사이즈 타입 (small, loose, tight etc)별로 리뷰 수를 다시 집계해 보자.

```

SELECT      'CLOTHING ID',
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END) N_SIZE
FROM MYDATA.DATASET2
GROUP
BY 1
;

```

CLOTHING ID	N_SIZE
1	767
2	1080
3	1077
4	1049
5	847
6	858
7	1095
8	1065
9	853
10	1120

[그림 5-25 쿼리 실행 결과]

다음으로 사이즈 태입을 추가해 집계해 보자.

```

SELECT 'CLOTHING ID',
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END) N_SIZE_T,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END )/
SUM(1) N_SIZE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LARGE%' THEN 1 ELSE 0 END )/
SUM(1) N_LARGE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LOOSE%' THEN 1 ELSE 0 END )/
SUM(1) N_LOOSE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SMALL%' THEN 1 ELSE 0 END )/
SUM(1) N_SMALL,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%TIGHT%' THEN 1 ELSE 0 END )/
SUM(1) N_TIGHT
FROM MYDATA.DATASET2
GROUP
BY 1
;

```

CLOTHING ID	N_SIZE_T	N_SIZE	N_LARGE	N_LOOSE	N_SMALL	N_TIGHT
1	767	0	0	0	0	0
2	1080	58	0.2358	0.0854	0.0285	0.1667
3	1077	83	0.3255	0.1569	0.0549	0.1686
4	1049	11	0.44	0	0	0.16
5	847	1	0.25	0.25	0	0
6	858	9	0.45	0.1	0.1	0.35
7	1095	107	0.3919	0.1429	0.044	0.2564
8	1065	4	0.3333	0.0833	0	0.25
9	853	0	0.1667	0	0	0
10	1120	1	0.5	0.5	0	0.5

[그림 5-26 쿼리 실행 결과]

이제 어떤 옷이 사이즈와 관련된 Complain이 많고, 어떤 타입의 Complain이 많은지 알게 되었다. 상품 개발팀이나 디자인팀에서 이 정보를 알고 있다면, 다음 상품 개발에 큰 도움이 될 것이다. 타 부서와 관련 내용을 공유하도록 해당 테이블을 mydata에 SIZE_STAT이라는 이름으로 생성해 보자.

```
CREATE TABLE MYDATA.SIZE_STAT_AS
SELECT 'CLOTHING ID',
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END) N_SIZE_T,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SIZE%' THEN 1 ELSE 0 END )/
SUM(1) N_SIZE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LARGE%' THEN 1 ELSE 0 END )/
SUM(1) N_LARGE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%LOOSE%' THEN 1 ELSE 0 END )/
SUM(1) N_LOOSE,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%SMALL%' THEN 1 ELSE 0 END )/
SUM(1) N_SMALL,
SUM(CASE WHEN 'REVIEW TEXT' LIKE '%TIGHT%' THEN 1 ELSE 0 END )/
SUM(1) N_TIGHT
FROM MYDATA.DATASET2
GROUP
BY 1
;
```

[그림 5-27 쿼리 실행 결과]

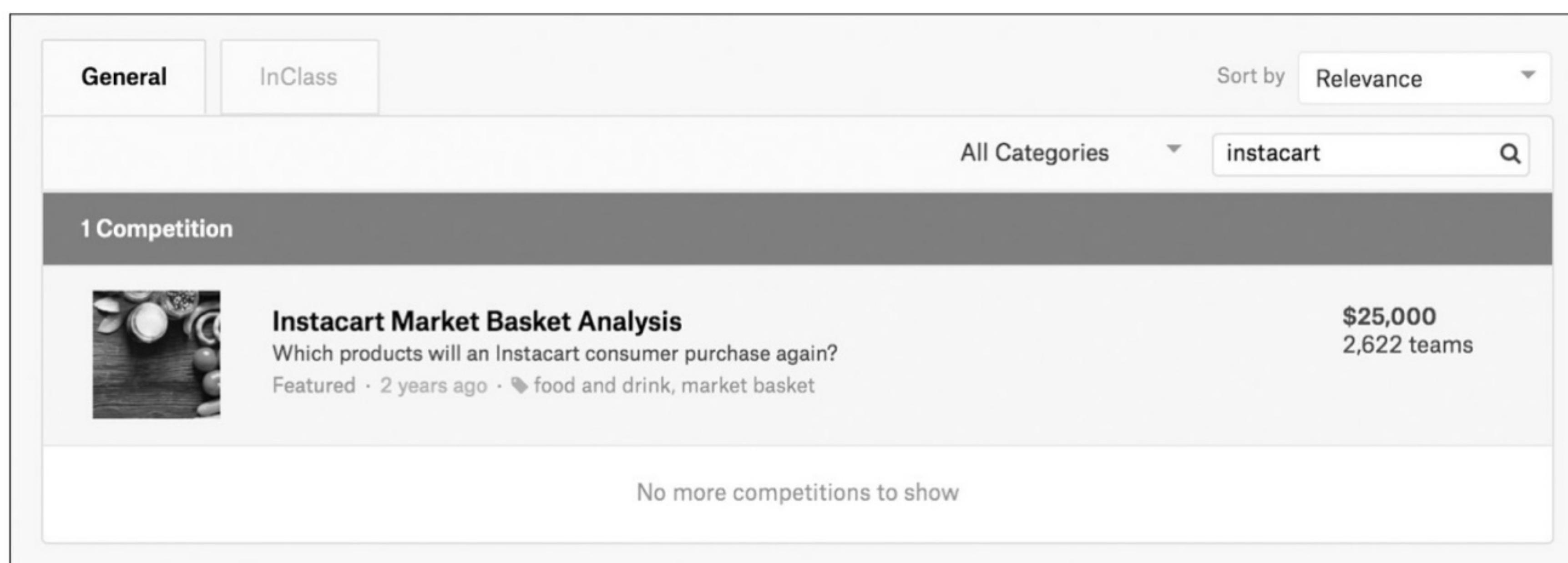
우리는 이 장에서 상품 리뷰라는 텍스트 데이터를 다루어 보았다. 실제로 대부분의 상품명, 상품 카테고리, 회원 주소와 같은 정보는 텍스트로 되어 있다. 같이 살펴본 내용을 잘 학습한다면, 텍스트 데이터도 SQL에서 어렵지 않게 다룰 수 있으리라 생각한다.

6 장

식품 배송 데이터 분석

1 데이터 생성

이번에는 Instacart라는 e-commerce 회사의 데이터를 분석해 본다. 먼저 해당 데이터 세트는 kaggle(Machine Learning Competition)에 존재하는 데이터 세트이다.



[그림 6-1 Kaggle Instacart]

해당 데이터의 크기가 매우 크기 때문에 샘플링한 데이터 세트를 다음 링크에서 다운로드해 사용한다. 아래 url을 통해 샘플링한 데이터를 다운로드받을 수 있다.

https://github.com/billyrohh/instacart_dataset

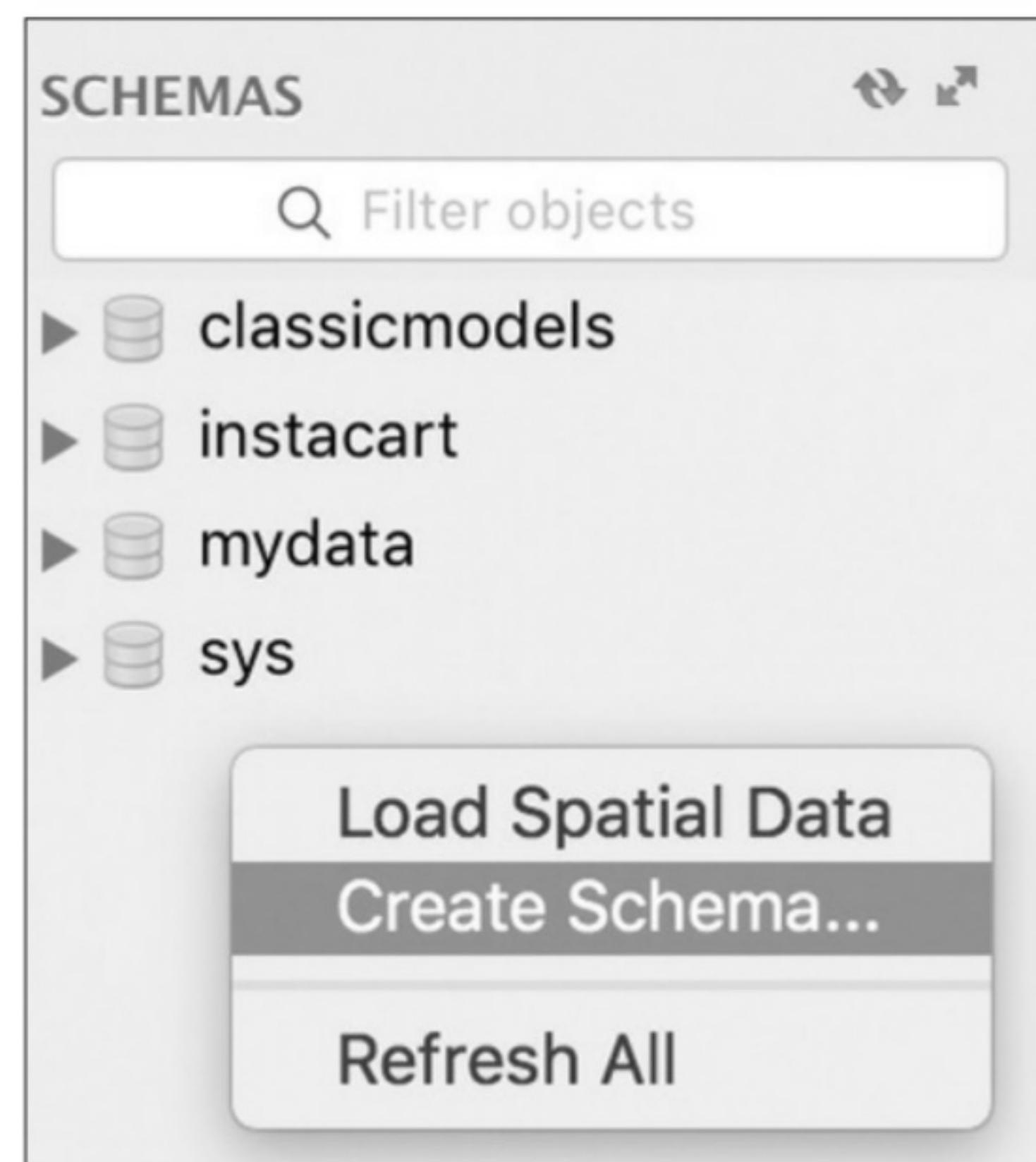
The screenshot shows a GitHub repository page. At the top, there are four buttons: 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The 'Clone or download' button is highlighted with a black border. Below the buttons, it says 'Latest commit dff2b47 on 13 Sep'. A list of five commits follows, each with the message 'Add Dataset' and the timestamp '3 months ago'. The commits are: 'Add Dataset' (3 months ago), and 'Add Dataset' (3 months ago).

[그림 6-2 Github 1]

Clone or download를 클릭한 뒤 Download Zip을 선택하면, Zip 파일 형태로 데이터를 다운로드받을 수 있다.

This screenshot shows the same GitHub repository page as Figure 6-2, but with the 'Clone or download' button expanded. It displays two options: 'Clone with HTTPS' and 'Use SSH'. Below these options is a text input field containing the URL 'https://github.com/billyrohh/instacart_dataset'. At the bottom of the expanded menu, there are two buttons: 'Open in Desktop' and 'Download ZIP'. The 'Download ZIP' button is highlighted with a black border. The background of the page shows the same list of commits as Figure 6-2.

[그림 6-3 Github 2]



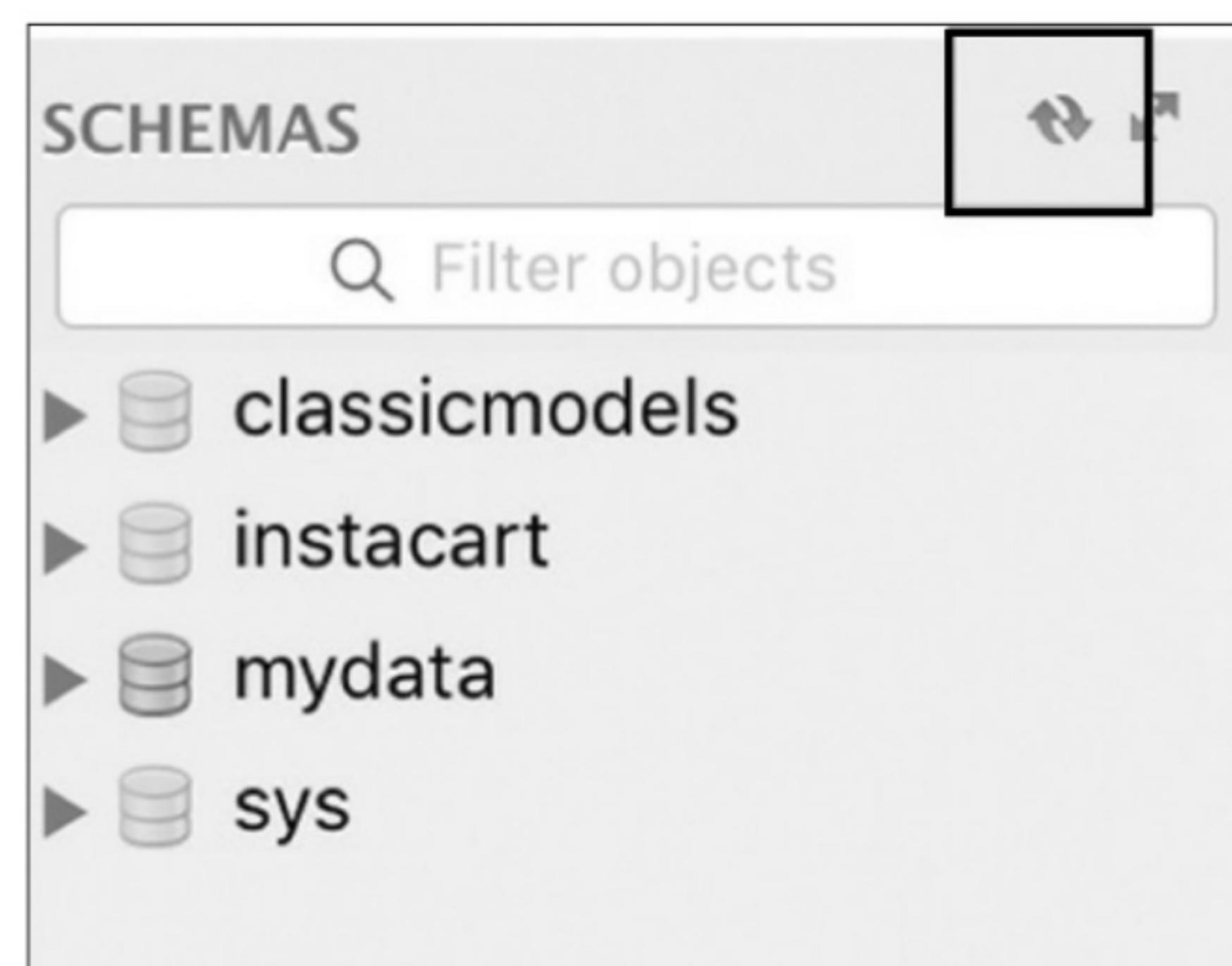
[그림 6-4 MySQL Workbench Create Schema 1]

데이터를 모두 다운받은 뒤, Workbench의 Schemas 부분의 공란 우측을 클릭해 Create Schema를 선택한다.



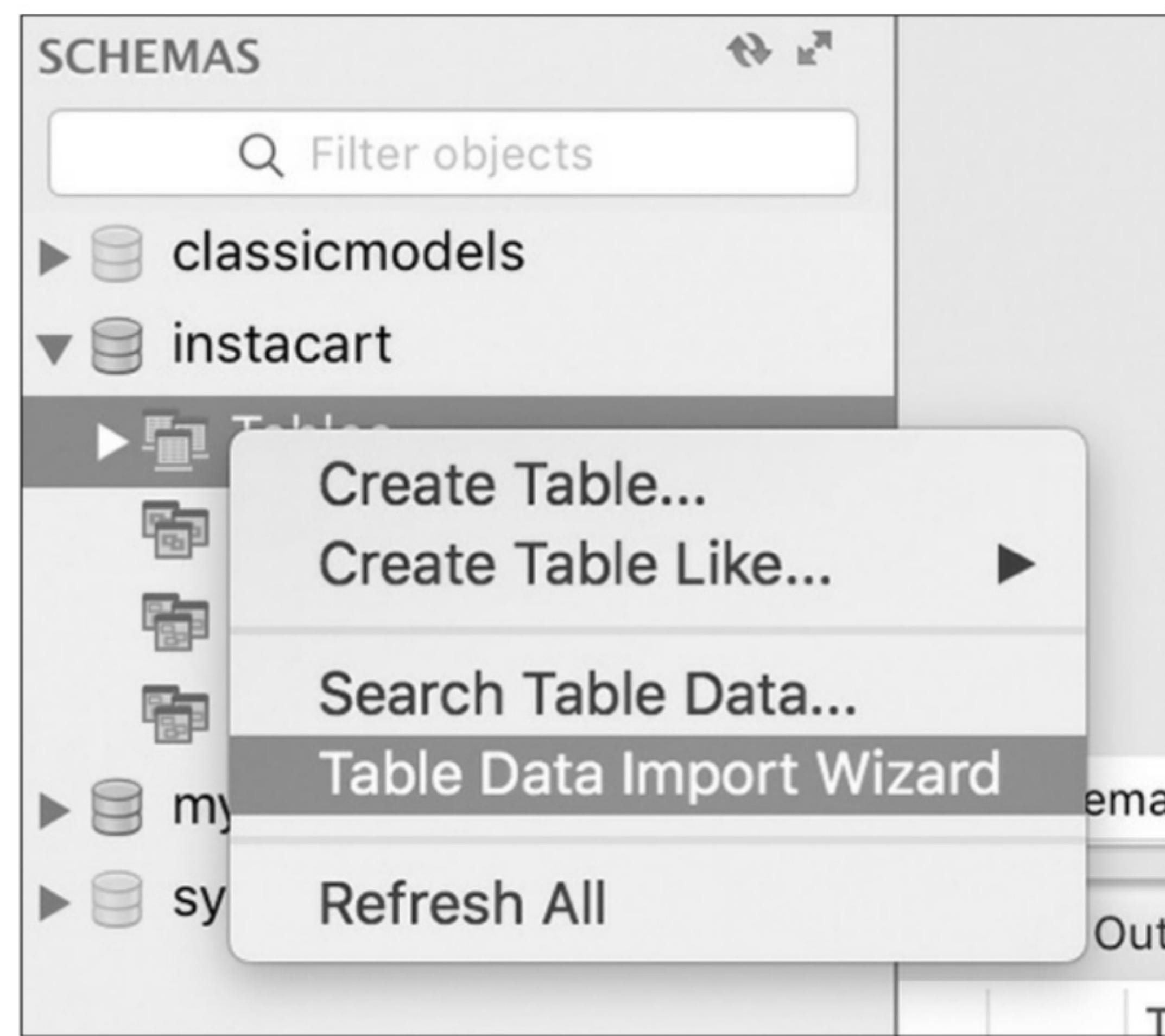
[그림 6-5 MySQL Workbench Create Schema 2]

database 명은 instacart로 설정한다. (3장 참고)



[그림 6-6 MySQL Workbench Create Schema 3]

데이터베이스를 생성한 후 새로 고침을 클릭하면 instacart가 생성된다.

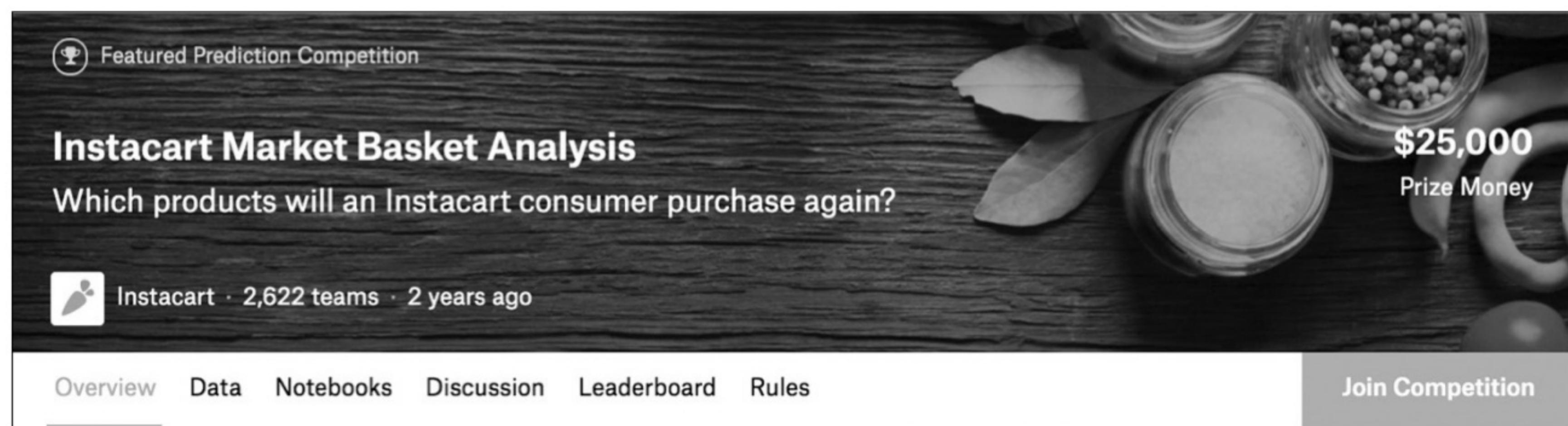


[그림 6-7 데이터 불러오기]

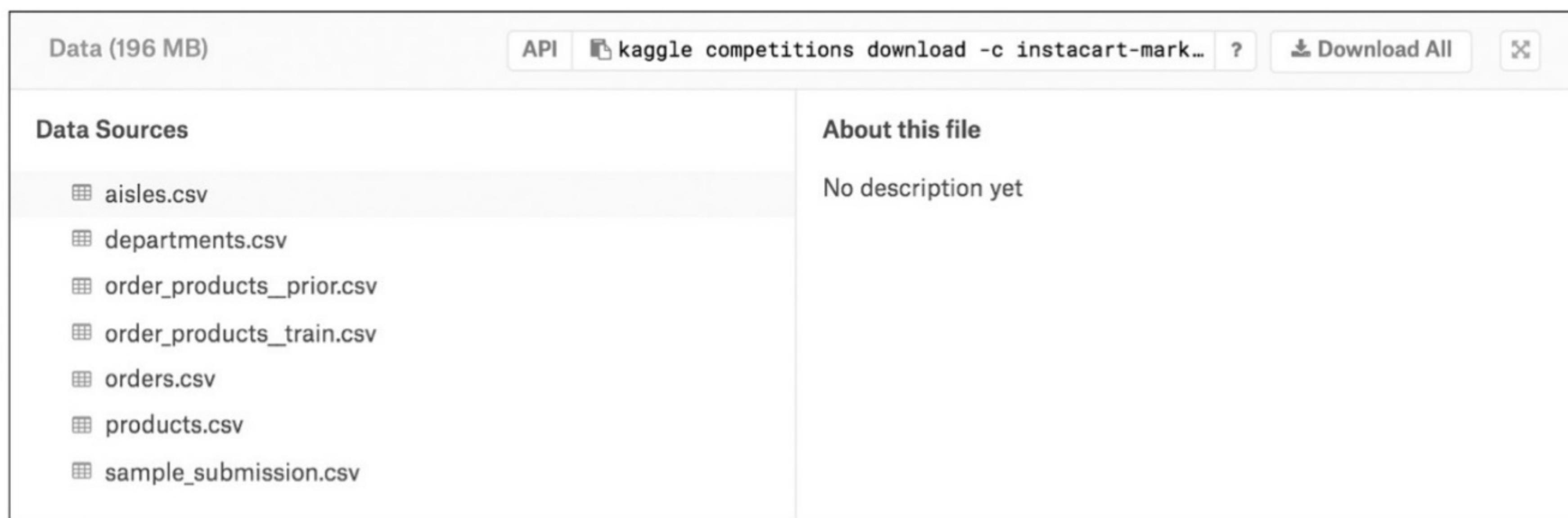
Instacart를 우클릭한 뒤 Table Data Import Wizard를 클릭하고, 다운받은 파일을 import해 준다. (3장 참고)

2 Instacart Dataset

“박 대리님, 식자재 배송 업체에서 데이터 분석해 달라는 요청이 들어왔습니다. 분석의 목적은 Business 현황과 재구매에 대한 insight입니다. 박 대리님이 자유롭게 분석해 주시고, 인사이트 알려 주세요.”



[그림 6-8 Kaggle Instacart]



[그림 6-9 Kaggle Instacart]

처음 접하는 데이터가 있다면 항상 위와 같이 각 테이블이 무엇을 의미하는지, 어떤 칼럼이 있는지 확인하고 테이블 간의 관계를 파악해야 한다.

데이터 세트에는 aisles, departments, order_products_prior (order_products_train은 사용하지 않는다.), orders, products 테이블이 존재한다. aisles, departments는 상품의 카테고리를 의미하고 order_products_prior는 각 주문 번호의 상세 구매 내역, orders는 주문 대표 정보, products는 상품 정보를 포함한다.

각 테이블에서 정보를 보여주는 형식은 다음과 같다.

aisles

	aisle_id	aisle
1	1	prepared soups salads
2	2	specialty cheeses
3	3	energy granola bars
4	4	instant foods
5	5	marinades meat preparation

[그림 6-10 Aisles]

departments

	department_id	department
1	1	frozen
2	2	other
3	3	bakery
4	4	produce
5	5	alcohol

[그림 6-11 departments]

order_products_prior

	order_id	product_id	add_to_cart_order	reordered
1	2581	40174	1	0
2	2581	17461	2	0
3	2581	19731	3	1
4	5880	24838	1	1
5	5880	24852	2	1

[그림 6-12 order_products_prior]

orders

	order_id	user_id	eval_set	order_number	order_dow	order_hour_of_day	days_since_prior_order
1	2352815	185282	prior	5	2	09	6
2	1414622	41949	prior	9	0	16	6
3	2534064	24631	prior	66	1	13	4
4	338572	95289	prior	4	2	14	18
5	2578618	170417	prior	5	1	13	14

[그림 6-13 orders]

Products

product_id		product_name	aisle_id	department_id
1	1	Chocolate Sandwich Cookies	61	19
2	10	Sparkling Orange Juice & Prickly Pear Beverage	115	7
3	11	Peach Mango Juice	31	7
4	23	Organic Turkey Burgers	49	12
5	25	Salted Caramel Lean Protein & Fiber Bar	3	19

[그림 6-14 products]

3 지표 추출

재구매와 관련된 요인들을 찾기 전 해당 Business의 전반적인 현황을 파악하자.

- 1) 전체 주문 건수
- 2) 구매자 수
- 3) 상품별 주문 건수
- 4) 카트에 가장 먼저 넣는 상품 10개
- 5) 시간별 주문 건수
- 6) 첫 구매 후 다음 구매까지 걸린 평균 일수
- 7) 주문 건당 평균 구매 상품 수(UPT, Unit Per Transaction)
- 8) 인당 평균 주문 건수
- 9) 재구매율이 가장 높은 상품 10개
- 10) Department별 재구매율이 가장 높은 상품 10개

1) 전체 주문 건수

4장 자동차 매출 데이터를 이용한 리포트 작성에서도 주문 건수를 여러 번 계산했다. 주문 건수를 계산하기 위해서 사전에 필요한 작업이 있었다. 바로 데이터의 중

복 여부를 확인하는 것이다.

만약 ORDERS라는 테이블에 주문 번호(ORDER_ID)가 중복되어 존재하는데 단순히 COUNT하게 되면, 실제 주문 건수보다 더 큰 값이 계산되어 나올 것이다.

order_id
A
A
B
C

[표 6-1 Order_id]

ORDER_ID에 위와 같이 A라는 값이 두 번 중복으로 입력되어 있을 때, 우리는 A를 두 번 카운트할 것이 아니라 중복을 제거하고 한 번만 COUNT해야 한다. 중복을 제거하고 개수를 세는 방법에는 COUNT(DISTINCT)가 있었다.

쿼리를 작성하기 전에 데이터 구조를 정확히 파악하는 것은 굉장히 중요한 일이다. 만, 급하게 쿼리를 작성하다 보면 이런 과정을 종종 빠뜨리곤 한다.

데이터의 정합성을 보장하는 것은 매우 중요한 일이다. 쿼리를 작성하기 전, 항상 데이터의 구조를 정확히 파악하고 수치를 구하자.

그럼 주문 건수와 구매자 수는 다음과 같은 방법으로 간단히 구할 수 있다.

```
SELECT COUNT(DISTINCT ORDER_ID) F  
FROM INSTACART.ORDERS;
```

F
1 3220

[그림 6-15 쿼리 실행 결과]

2) 구매자 수

```
SELECT COUNT(DISTINCT USER_ID) BU  
FROM INSTACART.ORDERS;
```

BU
1 3159

[그림 6-16 쿼리 실행 결과]

주문 건수와 구매자 수의 경우 ORDERS 테이블에서 중복으로 존재 가능하므로 distinct를 이용해 중복을 제외하고 카운트한다.

3) 상품별 주문 건수

상품별 주문 건수를 계산하려면, 먼저 상품명으로 데이터를 그룹핑하고 ORDER_ID를 카운트해 집계하면 된다.

문제는 주문 번호(ORDER_ID)는 ORDER_PRODUCTS_PRIOR라는 테이블에 존재하고, 상품명(PRODUCT_NAME)은 PRODUCTS라는 테이블에 존재한다는 점이다.

분산된 정보를 함께 사용하려면 2개의 테이블을 결합(JOIN)해야 한다.

	order_id	product_id	add_to_cart_order	reordered
1	2581	40174	1	0
2	2581	17461	2	0
3	2581	19731	3	1
4	5880	24838	1	1
5	5880	24852	2	1
6	5880	16797	3	1
7	5880	5456	4	0
8	5880	25199	5	0
9	6030	280	1	0
10	7374	25146	1	0

[그림 6-17 쿼리 실행 결과]

	product_id	product_name	aisle_id	department_id
1	1	Chocolate Sandwich Cookies	61	19
2	10	Sparkling Orange Juice & Prickly Pear Beverage	115	7
3	11	Peach Mango Juice	31	7
4	23	Organic Turkey Burgers	49	12
5	25	Salted Caramel Lean Protein & Fiber Bar	3	19
6	29	Fresh Cut Golden Sweet No Salt Added Whole Kernel Corn	81	15
7	32	Nacho Cheese White Bean Chips	107	19
8	34	Peanut Butter Cereal	121	14
9	35	Italian Herb Porcini Mushrooms Chicken Sausage	106	12
10	37	Noodle Soup Mix With Chicken Broth	69	15

[그림 6-18 쿼리 실행 결과]

2개 테이블에 공통으로 존재하는 product_id로 2개의 테이블을 JOIN한다.

```
SELECT *
FROM INSTACART.ORDER_PRODUCTS__PRIOR A
LEFT
JOIN INSTACART.PRODUCTS B
ON A.PRODUCT_ID = B.PRODUCT_ID
;
```

	order_id	product_id	add_to_cart_order	reordered	product_name	aisle_id	department_id
1	206076	1	8	1	Chocolate Sandwich Cookies	61	19
2	1211209	1	4	0	Chocolate Sandwich Cookies	61	19
3	773393	10	3	0	Sparkling Orange Juice & Prickly Pear Beverage	115	7
4	11151662	23	5	1	Organic Turkey Burgers	49	12
5	397329	29	9	1	Fresh Cut Golden Sweet No Salt Added Whole Kernel Corn	81	15
6	303387	34	5	0	Peanut Butter Cereal	121	14
7	316724	34	15	0	Peanut Butter Cereal	121	14
8	767193	34	18	0	Peanut Butter Cereal	121	14
9	808779	34	19	0	Peanut Butter Cereal	121	14
10	953456	34	2	1	Peanut Butter Cereal	121	14

[그림 6-19 쿼리 실행 결과]

2개의 테이블을 조합한 결과를 보면, order_products_prior의 product_id에 해당하는 product_name, aisle_id, department_id가 결합한 것을 볼 수 있다.

이제 결합한 결과에서 PRODUCT_NAME으로 데이터를 그룹핑하고, ORDER_ID를 카운트하면 된다. 이때 주문 번호(ORDER_ID)는 동일한 값이 중복으로 존재하므로 중복을 제거하고 집계해야 한다. 다음과 같은 방법으로 계산 가능하다.

```

SELECT B.PRODUCT_NAME,
       COUNT(DISTINCT A.ORDER_ID) F
  FROM INSTACART.ORDER_PRODUCTS__PRIOR A
  LEFT
 JOIN INSTACART.PRODUCTS B
    ON A.PRODUCT_ID = B.PRODUCT_ID
 GROUP
    BY 1;

```

	PRODUCT_NAME	F
1	"Constant Comment"" Decaffeinated Black Tea Blend"	1
2	#2 Coffee Filters	1
3	0% Fat Black Cherry Greek Yogurt	1
4	0% Fat Blueberry Greek Yogurt	1
5	0% Fat Free Organic Milk	4
6	0% Fat Organic Greek Vanilla Yogurt	1
7	0% Fat Superfruits Greek Yogurt	1
8	0% Greek Strained Yogurt	6
9	0% Greek, Blueberry on the Bottom Yogurt	2
10	1 % Lowfat Milk	1

[그림 6-20 쿼리 실행 결과]

4) 장바구니에 가장 먼저 넣는 상품 10개

우리는 쇼핑몰에서 물건을 구입할 때, 장바구니 담기라는 기능을 자주 사용한다. 가장 필요한 상품을 먼저 검색하고 장바구니에 담는 경향이 있다. 그렇다면 장바구니에 먼저 담기는 상품들에는 무엇이 있을까?

먼저 ORDER_PRODUCTS__PRIOR라는 테이블을 보면 ADD_TO_CART_ORDER라는 칼럼이 존재한다. ADD_TO_CART_ORDER는 상품이 몇 번째로 장바구니에 담겼는지를 의미한다.

	order_id	product_id	add_to_cart_order	reordered
1	2581	40174	1	0
2	2581	17461	2	0
3	2581	19731	3	1
4	5880	24838	1	1
5	5880	24852	2	1
6	5880	16797	3	1
7	5880	5456	4	0
8	5880	25199	5	0
9	6030	280	1	0
10	7374	25146	1	0

[그림 6-21 쿼리 실행 결과]

예를 들어, 주문 번호(ORDER_ID) 2581에서 40174라는 상품은 첫 번째로 장바구니에 담겼고, 19731이라는 상품은 세 번째로 장바구니에 담겼다.

그러면 해당 테이블을 이용해 어떻게 장바구니에 먼저 담긴 상품들을 구할 수 있을까?

카트에 가장 먼저 담기는 상품을 조회하기 위해서 상품별로 가장 먼저 카트에 담긴 경우를 카운트한다. 즉 mineral water라는 상품이 A라는 주문 건에서 두 번째로 담기고, B라는 주문 건에서 첫 번째로 담기는 경우에 mineral water가 첫 번째로 카트에 담긴 건수는 1이 된다.

먼저 ORDER_PRODUCTS__PRIOR의 PRODUCT_ID별로 가장 먼저 담긴 경우 (ADD_TO_CART_ORDER = 1)에는 1을 출력하는 칼럼을 생성해 보자.

```
SELECT PRODUCT_ID,
CASE WHEN ADD_TO_CART_ORDER = 1 THEN 1 ELSE 0 END F_1ST
FROM INSTACART.ORDER_PRODUCTS__PRIOR
;
```

	PRODUCT_ID	F_1ST
1	40174	1
2	17461	0
3	19731	0
4	24838	1
5	24852	0
6	16797	0
7	5456	0
8	25199	0
9	280	1
10	25146	1

[그림 6-22 쿼리 실행 결과]

다음으로 상품 번호(PRODUCT_ID)로 데이터를 그룹핑하고 위의 F_1ST 칼럼을 합하면, 상품별로 장바구니에 가장 먼저 담긴 건수를 계산할 수 있다.

```
SELECT PRODUCT_ID,  
SUM(CASE WHEN ADD_TO_CART_ORDER = 1 THEN 1 ELSE 0 END) F_1ST  
FROM INSTACART.ORDER_PRODUCTS__PRIOR  
GROUP  
BY 1  
;
```

	PRODUCT_ID	F_1ST
1	40174	2
2	17461	4
3	19731	0
4	24838	5
5	24852	117
6	16797	17
7	5456	0
8	25199	0
9	280	1
10	25146	1

[그림 6-23 쿼리 실행 결과]

○ 이제 F_1ST(장바구니에 가장 먼저 담긴 건수)로 데이터에 순서를 매긴다.

```
SELECT *,  
ROW_NUMBER() OVER(ORDER BY F_1ST DESC) RNK  
FROM  
(SELECT PRODUCT_ID,  
SUM(CASE WHEN ADD_TO_CART_ORDER = 1 THEN 1 ELSE 0 END) F_1ST  
FROM INSTACART.ORDER_PRODUCTS__PRIOR  
GROUP  
BY 1) A  
;
```

	PRODUCT_ID	F_1ST	RNK
1	24852	117	1
2	13176	62	2
3	27845	37	3
4	21137	31	4
5	21903	27	5
6	47209	24	6
7	19660	18	7
8	16797	17	8
9	5785	16	9
10	12341	16	10

[그림 6-24 쿼리 실행 결과]

우리는 1~10위의 상품 번호만 궁금하므로 WHERE 절에 조건을 추가해 RNK가 1~10 사이인 데이터만 출력한다. 이때 RNK는 SELECT 문에서 새롭게 생성한 칼럼 이므로 WHERE 절에서 바로 사용할 수가 없다. 따라서 위의 쿼리를 SUBQUERY로 사용해 조건을 생성해야 한다.

```
SELECT *
FROM
(SELECT *,  
ROW_NUMBER() OVER(ORDER BY F_1ST DESC) RNK
FROM
(SELECT PRODUCT_ID,  
SUM(CASE WHEN ADD_TO_CART_ORDER = 1 THEN 1 ELSE 0 END) F_1ST
FROM INSTACART.ORDER_PRODUCTS__PRIOR
GROUP
BY 1) A) BASE
WHERE RNK BETWEEN 1 AND 10;
```

	PRODUCT_ID	F_1ST	RNK
1	24852	117	1
2	13176	62	2
3	27845	37	3
4	21137	31	4
5	21903	27	5
6	47209	24	6
7	19660	18	7
8	16797	17	8
9	5785	16	9
10	12341	16	10

[그림 6-25 쿼리 실행 결과]

우리는 랭크(RNK)를 생성해 10개의 데이터를 출력했지만, ORDER BY를 이용해 간단히 상위 10개의 데이터를 호출할 수도 있다.

```
SELECT PRODUCT_ID,  
SUM(CASE WHEN ADD_TO_CART_ORDER = 1 THEN 1 ELSE 0 END) F_1ST  
FROM INSTACART.ORDER_PRODUCTS__PRIOR  
GROUP  
BY 1  
ORDER  
BY 2 DESC LIMIT 10  
;
```

	PRODUCT_ID	F_1ST
1	24852	117
2	13176	62
3	27845	37
4	21137	31
5	21903	27
6	47209	24
7	19660	18
8	16797	17
9	5785	16
10	12341	16

[그림 6-26 쿼리 실행 결과]

이처럼 간단하게 앞에서 구한 내용을 출력할 수 있다. 쿼리를 작성하는 방법은 그 만큼 다양하고, 정해진 방법이 없다. 더 효율적인 방법을 고민하고 작성하도록 연습할 필요가 있다. 우리가 앞에서 랭크를 생성하고 이용한 방법은 많이 사용되는 방법이라 자주 살펴보고 연습할 필요가 있다.

5) 시간별 주문 건수

ORDERS의 ORDER_HOUR_OF_DAY로 그룹핑한 뒤, ORDER_ID를 카운트한다. 이때 ORDER_ID에 중복이 있을 수 있어 DISTINCT를 추가해 중복을 제거해 준다.

```
SELECT ORDER_HOUR_OF_DAY,
       COUNT(DISTINCT ORDER_ID) F
  FROM INSTACART.ORDERS
 GROUP
 BY 1
 ORDER
 BY 1;
```

	ORDER_HOUR_OF_DAY	F
1	00	21
2	01	12
3	02	3
4	03	7
5	04	2
6	05	4
7	06	33
8	07	95
9	08	163
10	09	239

[그림 6-27 쿼리 실행 결과]

6) 첫 구매 후 다음 구매까지 걸린 평균 일수

ORDERS 테이블에서 DAYS_SINCE_PRIOR_ORDER는 이전 주문이 이루어진 지 며칠 뒤에 구매가 이루어졌는지를 나타내는 값이다. 즉 주문 번호의 ORDER_NUMBER가 2인(유저의 2번째 주문 건) DAYS_SINCE_PRIOR_ORDER는 첫 구매 후 다음 구매까지 걸린 기간이 된다. 이 기간을 평균하면 첫 구매 후 다음 구매까지 걸린 평균 일수를 계산할 수 있다.

```
SELECT AVG(DAYS_SINCE_PRIOR_ORDER) AVG_RECENCY
FROM INSTACART.ORDERS
WHERE ORDER_NUMBER = 2;
```

	AVG_RECENCY
1	13.6518

[그림 6-28 쿼리 실행 결과]

원하는 데이터를 정확하게 추출하려면 데이터의 구조를 정확하게 파악해야 한다. 테이블의 구조를 정확히 파악하지 않았다면 풀기 힘든 문제가 될 수 있다.

7) 주문 건당 평균 구매 상품 수(UPT, Unit Per Transaction)

PRODUCT_ID를 카운트해 상품 개수를 계산하고, 이를 주문 건수로 나누어 주문 1 건에 평균적으로 몇 개의 상품을 구매하는지 파악할 수 있다.

```
SELECT COUNT(PRODUCT_ID)/COUNT(DISTINCT ORDER_ID) UPT  
FROM INSTACART.ORDER_PRODUCTS__PRIOR;
```

UPT
1 10.0467

[그림 6-29 쿼리 실행 결과]

8) 인당 평균 주문 건수

거래당 구매 상품 수를 구했듯이, 전체 주문 건수를 구매자 수로 나누어 인당 평균 주문 건수를 계산할 수 있다.

```
SELECT COUNT(DISTINCT ORDER_ID)/COUNT(DISTINCT USER_ID) AVG_F  
FROM INSTACART.ORDERS;
```

AVG_F
1 1.0193

[그림 6-30 쿼리 실행 결과]

9) 재구매율이 가장 높은 상품 10개

재구매율이 가장 높은 상품을 구하려면 어떻게 해야 할까? 상품별로 재구매율을 계산한 뒤, 재구매율을 기준으로 랭크를 계산한다. 이후, 원하는 랭크 값(1~10)으로 조건을 생성한다.

- 상품별 재구매율 계산

상품 번호로 데이터를 그룹핑하고, 재구매 수(SUM(CASE WHEN REORDERED =1 THEN 1 ELSE 0 END))를 전체 구매 수(COUNT(*))로 나누어 재구매율을 계산한다.

```
SELECT PRODUCT_ID,  
SUM(CASE WHEN REORDERED =1 THEN 1 ELSE 0 END)/COUNT(*) RET_RATIO  
FROM INSTACART.ORDER_PRODUCTS__PRIOR  
GROUP  
BY 1  
;
```

	PRODUCT_ID	RET_RATIO
1	40174	0.5882
2	17461	0.7059
3	19731	0.6
4	24838	0.6429
5	24852	0.8189
6	16797	0.7143
7	5456	0.5294
8	25199	0.4286
9	280	0
10	25146	0.55

[그림 6-31 쿼리 실행 결과]

- 재구매율로 랭크(순위) 열 생성하기

앞에서 생성한 상품별 재구매율로 순위를 생성한다. 재구매율(RET_RATIO)로 순위를 매기려면 앞의 결과를 SUBQUERY로 생성하고, SELECT 문에서 ROW_NUMBER / RANK / DENSE_RANK 중 하나를 선택해 순위 열을 생성한다.

```

SELECT *,  

ROW_NUMBER() OVER(ORDER BY RET_RATIO DESC) RNK  

FROM  

(SELECT PRODUCT_ID,  

SUM(CASE WHEN REORDERED =1 THEN 1 ELSE 0 END)/COUNT(*) RET_RATIO  

FROM INSTACART.ORDER_PRODUCTS__PRIOR  

GROUP  

BY 1) A  

;

```

	PRODUCT_ID	RET_RATIO	RNK
1	12220	1	1
2	48763	1	2
3	20039	1	3
4	16839	1	4
5	30735	1	5
6	35788	1	6
7	33381	1	7
8	31371	1	8
9	34450	1	9
10	23631	1	10

[그림 6-32 쿼리 실행 결과]

- Top 10(재구매율) 상품 추출

상품별로 재구매율을 계산하고, 재구매율 순위 열을 생성했다. 이제 생성한 순위 열의 값이 1~10 사이인 데이터를 조회하면, 우리가 처음 구하려고 했던 Top 10(재구매율) 상품 번호를 추출할 수 있다.

```

SELECT *  

FROM  

(SELECT *,  

ROW_NUMBER() OVER(ORDER BY RET_RATIO DESC) RNK  

FROM  

(SELECT PRODUCT_ID,  

SUM(CASE WHEN REORDERED =1 THEN 1 ELSE 0 END)/COUNT(*) RET_RATIO  

FROM INSTACART.ORDER_PRODUCTS__PRIOR

```

```
GROUP  
BY 1) A) A  
WHERE RNK BETWEEN 1 AND 10  
;
```

	PRODUCT_ID	RET_RATIO	RNK
1	12220	1	1
2	48763	1	2
3	20039	1	3
4	16839	1	4
5	30735	1	5
6	35788	1	6
7	33381	1	7
8	31371	1	8
9	34450	1	9
10	23631	1	10

[그림 6-33 쿼리 실행 결과]

10) Department별 재구매율이 가장 높은 상품 10개

앞에서 상품별로 재구매율을 구한 뒤, 재구매율이 가장 높은 상품 10개를 추출했다. Department별로 재구매율이 높은 상위 10개 상품을 추출하려면 어떻게 해야 할까?

Department별로 재구매율이 높은 상품을 추출하려면, Department별, 상품별 재구매율을 계산한다. 다음 Department로 Partition을 생성한 뒤 랭크를 계산하면 된다. 이후 원하는 조건(RNK: 1~10)을 생성하면, Department별 재구매율 상위 10개의 데이터를 추출할 수 있다.

앞의 예제에서 추가된 내용은 ORDER_PRODUCTS_PRIOR에 PRODUCTS 테이블을 조인해 DEPARTMENT를 가져와야 한다는 점과 DEPARTMENT로 PARTITION을 생성해 순위를 매겨야 한다는 점이다.

다음과 같은 방법으로 DEPARTMENT별 재구매율 상위 10개 상품을 출력할 수 있다.

```
SELECT *
FROM
(SELECT *,  
ROW_NUMBER() OVER(ORDER BY RET_RATIO DESC) RNK
FROM
(SELECT C.DEPARTMENT,  
PRODUCT_ID,  
SUM(CASE WHEN REORDERED =1 THEN 1 ELSE 0 END)/COUNT(*) RET_RATIO  
FROM INSTACART.ORDER_PRODUCTS__PRIOR A
LEFT
JOIN INSTACART.PRODUCTS B
ON A.PRODUCT_ID = B.PRODUCT_ID
LEFT
JOIN INSTACART.DEPARTMENTS C
ON B.DEPARTMENT_ID = C.DEPARTMENT_ID
GROUP
BY 1,2) A)
WHERE RNK BETWEEN 1 AND 10;
```

	DEPARTMENT	PRODUCT_ID	RET_RATIO	RNK
1	frozen	178	1	1
2	dairy eggs	28730	1	2
3	frozen	932	1	3
4	frozen	937	1	4
5	dairy eggs	29666	1	5
6	frozen	1188	1	6
7	frozen	1542	1	7
8	frozen	1560	1	8
9	frozen	1618	1	9
10	frozen	1620	1	10

[그림 6-34 쿼리 실행 결과]

4 구매자 분석

박 대리님, 정리해 주신 지표 감사합니다. 차후 분석은 구매자와 관련된 분석을 진행하고자 합니다. 10분위 분석, 고객의 주요 구매 카테고리, 고객 Segment를 통해 해당 서비스의 구매자에 대해 더 깊게 이해하고자 합니다. 그럼 부탁드립니다.

1) 10분위 분석

이전까지 매출과 관련된 지표들을 주로 살펴보았다면, 이제는 구매자에 집중해 데이터를 살펴보고자 한다. 먼저 10분위 분석을 통해 서비스의 주문 수가 VIP 고객에게 얼마나 집중되어 있는지 살펴보자.

고객별로 주문 건수를 계산한 뒤, 주문 건수를 기준으로 각 고객이 어떤 그룹에 속하는지 구한다.

그룹	비중
1	0~10%
2	10~20%
3	20~30%
4	30~40%
5	40~50%
6	50~60%
7	60~70%
8	70~80%
9	80~90%
10	90~100%

[표 6-2 각 분위별 비중(%)]

다음으로 각 분위 위수별 주문 건수의 합을 구하면, 전체 주문 건이 어떤 그룹에 얼마나 집중되어 있는지 계산할 수 있다.

- 10분위 분석이란?

전체를 10분위로 나누어 각 분위 수에 해당하는 집단의 성질을 나타내는 방법.

10분위 분석을 진행하려면 먼저 각 구매자의 분위 수를 구해야 한다. 우리는 고객들의 주문 건수를 기준으로 분위 수를 나눈다.

먼저 다음과 같은 방법으로 주문 건수에 따른 Rank를 생성한다.

```
SELECT *,  
ROW_NUMBER() OVER(ORDER BY F DESC) RNK  
FROM  
(SELECT USER_ID,  
COUNT(DISTINCT ORDER_ID) F  
FROM INSTACART.ORDERS  
GROUP  
BY 1) A
```

	USER_ID	F	RNK
1	32099	3	1
2	2610	2	2
3	10132	2	3
4	10972	2	4
5	12435	2	5
6	22379	2	6
7	25449	2	7
8	33244	2	8
9	37499	2	9
10	37801	2	10

[그림 6-35 쿼리 실행 결과]

고객별 주문 건수에 따라 순위가 매겨진다. 고객별로 분위 수를 매기려면 전체 고

객이 몇 명인지 알아야 한다. 다음과 같은 방법으로 전체 고객 수를 계산한다.

```
SELECT COUNT(DISTINCT USER_ID)
FROM
(SELECT USER_ID,
COUNT(DISTINCT ORDER_ID) F
FROM INSTACART.ORDERS
GROUP
BY 1) A
```

전체 고객 수는 3,159명으로 등수별 분위 수는 다음과 같이 계산된다.

분위 수	포함 등수
1	1~316
2	317~632
3	633~948
4	949~1,264
5	1,265~1,580
6	1,581~1,895
7	1,896~2,211
8	2,212~2,527
9	2,528~2,843
10	2,844~3,159

[표 6-3 각 분위별 포함 등수]

각 등수에 따른 분위 수는 CASE WHEN 구문을 이용해 다음과 같은 방법으로 설정 할 수 있다.

```

SELECT *,  

CASE WHEN RNK BETWEEN 1 AND 316 THEN 'Quantile_1'  

WHEN RNK BETWEEN 317 AND 632 THEN 'Quantile_2'  

WHEN RNK BETWEEN 633 AND 948 THEN 'Quantile_3'  

WHEN RNK BETWEEN 949 AND 1264 THEN 'Quantile_4'  

WHEN RNK BETWEEN 1265 AND 1580 THEN 'Quantile_5'  

WHEN RNK BETWEEN 1581 AND 1895 THEN 'Quantile_6'  

WHEN RNK BETWEEN 1896 AND 2211 THEN 'Quantile_7'  

WHEN RNK BETWEEN 2212 AND 2527 THEN 'Quantile_8'  

WHEN RNK BETWEEN 2528 AND 2843 THEN 'Quantile_9'  

WHEN RNK BETWEEN 2844 AND 3159 THEN 'Quantile_10' END quantile  

FROM  

(SELECT *,  

ROW_NUMBER() OVER(ORDER BY F DESC) RNK  

FROM  

(SELECT USER_ID,  

COUNT(DISTINCT ORDER_ID) F  

FROM INSTACART.ORDERS  

GROUP  

BY 1) A) A

```

	USER_ID	F	RNK	quantile
1	32099	3	1	Quantile_1
2	2610	2	2	Quantile_1
3	10132	2	3	Quantile_1
4	10972	2	4	Quantile_1
5	12435	2	5	Quantile_1
6	22379	2	6	Quantile_1
7	25449	2	7	Quantile_1
8	33244	2	8	Quantile_1
9	37499	2	9	Quantile_1
10	37801	2	10	Quantile_1

[그림 6-36 쿼리 실행 결과]

위 CASE WHEN 구문에서는 각 분위 수별로 등수를 지정했는데, 모든 등수를 기재 할 필요 없이 다음과 같은 방법으로도 처리 가능하다.

```

SELECT *,  

CASE WHEN RNK <= 316 THEN 'Quantile_1'  

WHEN RNK <= 632 THEN 'Quantile_2'  

WHEN RNK <= 948 THEN 'Quantile_3'  

WHEN RNK <= 1264 THEN 'Quantile_4'  

WHEN RNK <= 1580 THEN 'Quantile_5'  

WHEN RNK <= 1895 THEN 'Quantile_6'  

WHEN RNK <= 2211 THEN 'Quantile_7'  

WHEN RNK <= 2527 THEN 'Quantile_8'  

WHEN RNK <= 2843 THEN 'Quantile_9'  

WHEN RNK <= 3159 THEN 'Quantile_10' END quantile  

FROM  

(SELECT *,  

ROW_NUMBER() OVER(ORDER BY F DESC) RNK  

FROM  

(SELECT USER_ID,  

COUNT(DISTINCT ORDER_ID) F  

FROM INSTACART.ORDERS  

GROUP  

BY 1) A) A

```

	USER_ID	F	RNK	quantile
1	32099	3	1	Quantile_1
2	2610	2	2	Quantile_1
3	10132	2	3	Quantile_1
4	10972	2	4	Quantile_1
5	12435	2	5	Quantile_1
6	22379	2	6	Quantile_1
7	25449	2	7	Quantile_1
8	33244	2	8	Quantile_1
9	37499	2	9	Quantile_1
10	37801	2	10	Quantile_1

[그림 6-37 쿼리 실행 결과]

이제 각 분위 수별 특성을 파악해 보려고 한다. 각 분위 수별로 평균 Recency를 파악한다. 먼저 위의 조회 결과를 하나의 테이블로 생성해 user_id별 분위 수 정보를 생성한다.

```
CREATE TEMPORARY TABLE INSTACART.USER_QUANTILE AS
SELECT *,
CASE WHEN RNK <= 316 THEN 'Quantile_1'
WHEN RNK <= 632 THEN 'Quantile_2'
WHEN RNK <= 948 THEN 'Quantile_3'
WHEN RNK <= 1264 THEN 'Quantile_4'
WHEN RNK <= 1580 THEN 'Quantile_5'
WHEN RNK <= 1895 THEN 'Quantile_6'
WHEN RNK <= 2211 THEN 'Quantile_7'
WHEN RNK <= 2527 THEN 'Quantile_8'
WHEN RNK <= 2843 THEN 'Quantile_9'
WHEN RNK <= 3159 THEN 'Quantile_10' END quantile
FROM
(SELECT *,  
ROW_NUMBER() OVER(ORDER BY F DESC) RNK
FROM
(SELECT USER_ID,  
COUNT(DISTINCT ORDER_ID) F
FROM INSTACART.ORDERS
GROUP
BY 1) A) A
```

이제 다음과 같은 방법으로 각 분위 수별 전체 주문 건수의 합을 구할 수 있다.

```
SELECT QUANTILE,
SUM(F) F
FROM INSTACART.USER_QUANTILE
GROUP
BY 1
;
```

	QUANTILE	F
1	Quantile_1	377
2	Quantile_2	316
3	Quantile_3	316
4	Quantile_4	316
5	Quantile_5	316
6	Quantile_6	315
7	Quantile_7	316
8	Quantile_8	316
9	Quantile_9	316
10	Quantile_10	316

[그림 6-38 쿼리 실행 결과]

전체 주문 건수를 계산하고, 각 분위 수의 주문 건수를 전체 주문 건수로 나누어 보자.

```
SELECT SUM(F) FROM INSTACART.USER_QUANTILE;  
;
```

	SUM(F)
1	3220

[그림 6-39 쿼리 실행 결과]

```
SELECT QUANTILE,  
SUM(F)/3220 F  
FROM INSTACART.USER_QUANTILE  
GROUP  
BY 1  
;
```

	QUANTILE	F
1	Quantile_1	0.1171
2	Quantile_2	0.0981
3	Quantile_3	0.0981
4	Quantile_4	0.0981
5	Quantile_5	0.0981
6	Quantile_6	0.0978
7	Quantile_7	0.0981
8	Quantile_8	0.0981
9	Quantile_9	0.0981
10	Quantile_10	0.0981

[그림 6-40 쿼리 실행 결과]

결과를 보면 각 분위 수별로 주문 건수가 거의 균등하게 분포되어 있다. 즉 해당 서비스는 매출이 VIP에게 집중되지 않고, 전체 고객에 고르게 분포되어 있음을 알 수 있다.

5 상품 분석

Instacart라는 서비스의 지표와 분위 수에 따른 주문 건수의 비중을 계산해 보았다. 이제 재구매를 많이 하는 상품을 알아보고, 각 상품의 판매 특성에 대해 살펴보자.

먼저 재구매 비중이 높은 상품을 찾아본다. 상품별 재구매 비중(%)과 주문 건수를 계산한다.

```
SELECT PRODUCT_ID,
       SUM(REORDERED)/SUM(1) REORDER_RATE,
       COUNT(DISTINCT ORDER_ID) F
  FROM INSTACART.ORDER_PRODUCTS__PRIOR
 GROUP BY PRODUCT_ID
 ORDER BY REORDER_RATE DESC
```

	PRODUCT_ID	REORDER_RATE	F
1	4133	1	1
2	47809	1	1
3	29077	1	1
4	41591	1	1
5	47953	1	1
6	10586	1	1
7	16732	1	2
8	4557	1	1
9	29437	1	1
10	10779	1	2

[그림 6-41 쿼리 실행 결과]

주문 건수가 일정 건수(10건) 이하인 상품은 제외하고 보자. HAVING을 이용하면 일정 건수 이하인 상품들을 쉽게 제외할 수 있다.

```
SELECT A.PRODUCT_ID,
       SUM(REORDERED)/SUM(1) REORDER_RATE,
       COUNT(DISTINCT ORDER_ID) F
  FROM INSTACART.ORDER_PRODUCTS__PRIOR A
 LEFT JOIN INSTACART.PRODUCTS B
    ON A.PRODUCT_ID = B.PRODUCT_ID
 GROUP BY PRODUCT_ID
 HAVING COUNT(DISTINCT ORDER_ID) > 10
```

PRODUCT_ID	REORDER_RATE	F
1	45	0.7778
2	196	0.6857
3	260	0.7241
4	432	0.8108
5	651	0.6154
6	890	0.5
7	1158	0.6
8	1215	0.7692
9	1244	0.4545
10	1463	0.8571

[그림 6-42 쿼리 실행 결과]

• HAVING vs WHERE

SQL을 처음 배울 때 HAVING과 WHERE의 차이점에 대해 혼란스러워하는 경향이 있다. 2가지 구문의 차이점은 무엇일까?

먼저 WHERE 절은 FROM에 위치한 테이블에만 조건을 걸 수 있다. SELECT 문에서 새롭게 생성한 칼럼에 조건을 걸어야 하는 경우가 있는데, 많은 초급자가 SELECT에서 새롭게 생성한 칼럼을 WHERE에서 사용하는 실수를 한다.

반면 HAVING은 그룹핑한 데이터에 조건을 생성하고 싶을 때 사용한다.

```

SELECT A.PRODUCT_ID,
B.PRODUCT_NAME,,
COUNT(DISTINCT ORDER_ID) F
FROM INSTACART.ORDER_PRODUCTS__PRIOR A
LEFT
JOIN INSTACART.PRODUCTS B
ON A.PRODUCT_ID = B.PRODUCT_ID
GROUP
BY PRODUCT_ID,
B.PRODUCT_NAME
HAVING COUNT(DISTINCT ORDER_ID) > 10
;

```

앞의 예제를 보면 PRODUCT_ID로 데이터를 그룹핑한 뒤, 주문 번호를 카운트했다. 이처럼 그룹핑한 데이터에 조건을 생성하고 싶은 경우 HAVING을 이용하면 된다. 결국 위 쿼리를 실행하면 주문 건수가 10보다 큰 데이터만 출력된다.

어떤 상품들이 재구매율이 높은지 보기 위해 PRODUCTS 테이블을 Join해 살펴본다.

```
SELECT A.PRODUCT_ID,  
B.PRODUCT_NAME,  
SUM(REORDERED)/SUM(1) REORDER_RATE,  
COUNT(DISTINCT ORDER_ID) F  
FROM INSTACART.ORDER_PRODUCTS__PRIOR A  
LEFT  
JOIN INSTACART.PRODUCTS B  
ON A.PRODUCT_ID = B.PRODUCT_ID  
GROUP  
BY PRODUCT_ID,  
B.PRODUCT_NAME  
HAVING COUNT(DISTINCT ORDER_ID) > 10
```

PRODUCT_ID	PRODUCT_NAME	REORDER_RATE	F
1 45	European Cucumber	0.7778	18
2 196	Soda	0.6857	35
3 260	Cantaloupe	0.7241	29
4 432	Vanilla Almond Breeze Almond Milk	0.8108	37
5 651	Organic Salted Butter	0.6154	13
6 890	Organic Diced Tomatoes	0.5000	24
7 1158	Mango Chunks	0.6000	15
8 1215	Kidz All Natural Baked Chicken Nuggets	0.7692	13
9 1244	Organic Orange Bell Pepper	0.4545	11
10 1463	Organic Milk	0.8571	28

[그림 6-43 쿼리 실행 결과]

6 다음 구매까지의 소요 기간과 재구매 관계

재구매와 관계가 있는 변수는 무엇이 있을까? 커머스 사이트에서 자주 재구매하는 상품에는 어떤 종류가 있는가? 가장 대표적으로 생수, 세제, 휴지와 같은 생활 필수품이 있을 것이다. 만약 가정에 아이가 있다면, 기저귀 같은 상품도 일정한 주기로 구매할 것이다.

‘고객이 자주 재구매하는 상품은 그렇지 않은 상품보다 일정한 주기를 가질 것이다.’라는 가정을 세우고 수치를 살펴보자. 재구매율이 높은 순서대로 상품을 10가지 그룹으로 구분하고, 각 그룹에서의 구매 소요 기간의 분산(Variance)을 구해 보자.

분산은 그 확률 변수가 기댓값에서 얼마나 떨어진 곳에 분포하는지를 나타내는 값이다. 즉 분산이 낮을수록 데이터가 평균에 모이게 되고, 분산이 클수록 관측치는 평균에서 멀리 분포한다.

먼저 다음과 같은 방법으로 상품별 재구매율을 계산하고, 가장 높은 순서대로 순위를 매긴다.

```
SELECT *
FROM
(SELECT *,  
ROW_NUMBER() OVER(ORDER BY RET_RATIO DESC) RNK
FROM
(SELECT PRODUCT_ID,  
SUM(CASE WHEN REORDERED =1 THEN 1 ELSE 0 END)/COUNT(*) RET_RATIO  
FROM INSTACART.ORDER_PRODUCTS__PRIOR  
GROUP  
BY 1) A) A
```

	PRODUCT_ID	RET_RATIO	RNK
1	12220	1	1
2	48763	1	2
3	20039	1	3
4	16839	1	4
5	30735	1	5
6	35788	1	6
7	33381	1	7
8	31371	1	8
9	34450	1	9
10	23631	1	10

[그림 6-44 쿼리 실행 결과]

고객 분석에서 했던 10분위 분석과 동일한 방법으로 각 상품을 10개의 그룹으로 나눈다.

분위 수	포함 등수
1	1~929
2	930~1,858
3	1,859~2,786
4	2,787~3,715
5	3,716~4,644
6	4,645~5,573
7	5,574~6,502
8	6,503~7,430
9	7,431~8,359
10	8,360~9,288

[표 6-4 분위별 등수]

```

CREATE TEMPORARY TABLE INSTACART.PRODUCT_REPURCHASE_QUANTILE AS
SELECT A.PRODUCT_ID,
CASE WHEN RNK <= 929 THEN 'Q_1'
WHEN RNK <= 1858 THEN 'Q_2'
WHEN RNK <= 2786 THEN 'Q_3'
WHEN RNK <= 3715 THEN 'Q_4'
WHEN RNK <= 4644 THEN 'Q_5'

```

```

WHEN RNK <= 5573 THEN 'Q_6'
WHEN RNK <= 6502 THEN 'Q_7'
WHEN RNK <= 7430 THEN 'Q_8'
WHEN RNK <= 8359 THEN 'Q_9'
WHEN RNK <= 9288 THEN 'Q_10' END RNK_GRP
FROM
(SELECT *,  

ROW_NUMBER() OVER(ORDER BY RET_RATIO DESC) RNK
FROM
(SELECT PRODUCT_ID,  

SUM(CASE WHEN REORDERED =1 THEN 1 ELSE 0 END)/COUNT(*) RET_RATIO  

FROM INSTACART.ORDER_PRODUCTS__PRIOR  

GROUP  

BY 1) A) A
GROUP
BY 1,2

```

	PRODUCT_ID	RNK_GRP
1	12220	Q_1
2	48763	Q_1
3	20039	Q_1
4	16839	Q_1
5	30735	Q_1
6	35788	Q_1
7	33381	Q_1
8	31371	Q_1
9	34450	Q_1
10	23631	Q_1

[그림 6-45 쿼리 실행 결과]

위의 쿼리로 PRODUCT_ID별 분위 수를 계산할 수 있다. 계산된 상품별 분위 수는 PRODUCT_REPURCHASE_QUANTILE 테이블에 생성된다. 이제 각 분위 수별로 재구매 소요 시간의 분산을 구해 본다.

```
CREATE TEMPORARY TABLE INSTACART.PRODUCT_REPURCHASE_QUANTILE AS
SELECT A.PRODUCT_ID,
CASE WHEN RNK <= 929 THEN 'Q_1'
WHEN RNK <= 1858 THEN 'Q_2'
WHEN RNK <= 2786 THEN 'Q_3'
WHEN RNK <= 3715 THEN 'Q_4'
WHEN RNK <= 4644 THEN 'Q_5'
WHEN RNK <= 5573 THEN 'Q_6'
WHEN RNK <= 6502 THEN 'Q_7'
WHEN RNK <= 7430 THEN 'Q_8'
WHEN RNK <= 8359 THEN 'Q_9'
WHEN RNK <= 9288 THEN 'Q_10' END RNK_GRP
FROM
(SELECT *,  
ROW_NUMBER() OVER(ORDER BY RET_RATIO DESC) RNK
FROM
(SELECT PRODUCT_ID,  
SUM(CASE WHEN REORDERED =1 THEN 1 ELSE 0 END)/COUNT(*) RET_RATIO
FROM INSTACART.ORDER_PRODUCTS__PRIOR
GROUP  
BY 1) A) A
GROUP
BY 1,2
```

각 분위 수별 재구매 소요 기간의 분산을 구하려면 다음과 같은 정보를 결합해 구해야 한다.

- 상품별 분위 수: PRODUCT_REPURCHASE_QUANTILE
- 주문 소요 시간: INSTACART.ORDERS
- 주문 번호와 상품 번호: INSTACART.ORDER_PRODUCTS__PRIOR

먼저 INSTACART.ORDER_PRODUCTS__PRIOR에 ORDERS 테이블을 결합해 PRODUCT_ID의 DAYS_SINCE_PRIOR_ORDER를 구한다.

```
CREATE TEMPORARY TABLE INSTACART.ORDER_PRODUCTS__PRIOR2 AS  
SELECT PRODUCT_ID,  
DAYS_SINCE_PRIOR_ORDER  
FROM INSTACART.ORDER_PRODUCTS__PRIOR A  
INNER  
JOIN INSTACART.ORDERS B  
ON A.ORDER_ID = B.ORDER_ID;
```

다음으로 결합한 테이블에서 분위수, 상품별 구매 소요 기간의 분산을 계산한다.

```
SELECT A.RNK_GRP,  
A.PRODUCT_ID,  
VARIANCE(DAYS_SINCE_PRIOR_ORDER) VAR_DAYS  
FROM INSTACART.PRODUCT_REPURCHASE_QUANTILE A  
LEFT  
JOIN INSTACART.ORDER_PRODUCTS__PRIOR2 B  
ON A.PRODUCT_ID = B.PRODUCT_ID  
GROUP  
BY 1,2  
ORDER  
BY 1;
```

RNK_GRP	PRODUCT_ID	VAR_DAYS
1	Q_1	29
2	Q_1	32
3	Q_1	47
4	Q_1	128
5	Q_1	137
6	Q_1	138
7	Q_1	172
8	Q_1	209
9	Q_1	273
10	Q_1	446

[그림 6-46 쿼리 실행 결과]

각 분위 수, 상품의 구매 소요 기간의 분산을 계산할 수 있다. 우리의 가정은 다음과 같았다. ‘재구매율이 높은 상품군은 구매 주기가 일정할 것이다.’

이를 확인하기 위해서 재구매율에 따라 상품을 10가지 그룹으로 분할했고, 각 분위 수의 상품별 구매 소요 기간의 분산을 계산했다.

이제 각 분위 수의 상품 소요 기간 분산의 중위 수를 계산한다. 계산한 결과를 보고 분위 수별 구매 소요 기간에 차이가 존재하는지 확인할 수 있다.

MySQL에서는 Median 함수를 제공하지 않으므로 avg() 함수를 이용해 평균으로 대체한다.

```
SELECT RNK_GRP,
AVG(VAR_DAYS) AVG_VAR_DAYS
FROM
(SELECT A.RNK_GRP,
A.PRODUCT_ID,
VARIANCE(days_since_prior_order) VAR_DAYS
FROM PRODUCT_REPURCHASE_QUANTILE A
LEFT
JOIN INSTACART.ORDER_PRODUCTS__PRIOR B
ON A.PRODUCT_ID = B.PRODUCT_ID
LEFT
JOIN INSTACART.ORDERS C
ON B.ORDER_ID = C.ORDER_ID
GROUP
BY 1,2) A
GROUP
BY 1
ORDER
BY 1
```

RNK_GRP	PRODUCT_ID	VAR_DAYS
1	Q_1	29
2	Q_1	32
3	Q_1	47
4	Q_1	128
5	Q_1	137
6	Q_1	138
7	Q_1	172
8	Q_1	209
9	Q_1	273
10	Q_1	446

[그림 6-47 쿼리 실행 결과]

결과를 살펴보면, 분위 수에 따라 재구매 주기의 분산에 차이가 없는 것으로 확인된다. 즉 이 결과를 통해 재구매율이 높은 상품이라고 해서 구매 주기가 평균에 집중되지 않음을 확인할 수 있다.