

# 14강

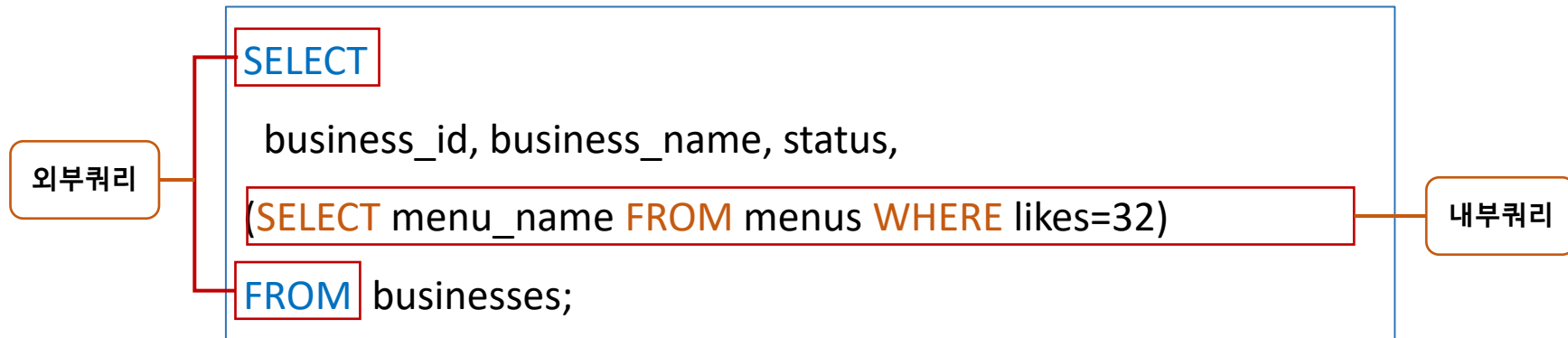
## 쿼리안의 서브쿼리





## 1. 비상관 서브쿼리

- 서브쿼리가 외부 쿼리의 어떤 것도 참조하지 않고 단독으로 사용되는 것을 **비상관 서브쿼리**라 한다.
- 내부 쿼리가 **우선 실행**되고, 그리고 나서 결과가 **외부쿼리의 WHERE 조건으로 이용**된다.
- 내부 쿼리는 외부 쿼리의 값과는 아무 상관이 없다. 단독으로도 쿼리가 실행될 수 있다.



	business_id	business_name	status	(select menu_name from menus where likes=32)
▶	1	화룡각	OPN	삼선짬뽕
	2	철구분식	OPN	삼선짬뽕
	3	알코렐라	RMD	삼선짬뽕
	4	바른떡볶이	OPN	삼선짬뽕
	5	복국냉면	OPN	삼선짬뽕
	6	보쌈마니아	OPN	삼선짬뽕
	7	에그사라다	VCT	삼선짬뽕
	8	달다방	OPN	삼선짬뽕
	9	마카오마카롱	OPN	삼선짬뽕
	10	김밥마라	OPN	삼선짬뽕
	11	소소스윗	OPN	삼선짬뽕
	12	사사서서소소...	VCT	삼선짬뽕
	13	림민짬뽕	CLS	삼선짬뽕
	14	파시조아	OPN	삼선짬뽕
	15	활매장국	CLS	삼선짬뽕
	16	노선이탈리아	OPN	삼선짬뽕
	17	커피엔코드	OPN	삼선짬뽕
	18	신림동백순대	VCT	삼선짬뽕




```
SELECT * FROM menus  
WHERE price < (SELECT AVG(price) FROM menus);
```

내부 쿼리 실행결과

	avg(price)
▶	7173.0769

Price의 평균값인 7173보다  
작은 price값만 가지는 레코  
드 출력



	menu_id	fk_business_id	menu_name	kilocalories	price	likes
▶	2	8	아메리카노	16.44	4500	6
	7	9	바닐라마카롱	160.62	1500	4
	11	14	단팥빵	225.88	1500	13
	12	1	간짜장	682.48	7000	3
	13	9	똥카롱	247.62	2000	8
	15	10	참치김밥	532.39	3000	0
	16	2	치즈떡볶이	638.42	5000	15
	17	11	플레인와플	299.31	6500	2
	18	2	참쌀순대	312.76	3000	-4
	20	4	국물떡볶이	483.29	4500	1
	21	10	돈가스김밥	562.72	4000	0
	24	4	라볶이	423.16	5500	0
	25	8	모카프라푸...	216.39	6000	8
	NULL	NULL	NULL	NULL	NULL	NULL



## 👉 여기서 잠깐 !!

<menus>테이블과 <businesses>테이블을 이용해 전체 메뉴 중 status가 'OPN'인 가장 높은 likes 값을 가진 메뉴 이름을 조회하는 필드명을 아래의 <출력 결과>와 같이 출력되도록 SQL문을 작성하시오.  
(단, 비상관 서브쿼리(select~where) 이용)

### 출력 결과

	business_name	status	max_likes
▶	화룡각	OPN	32
	철구분식	OPN	32
	바른떡볶이	OPN	32
	북극냉면	OPN	32
	보쌈마니아	OPN	32
	달다방	OPN	32
	마카오마카롱	OPN	32
	김밥마라	OPN	32
	소소스윗	OPN	32
	파시조아	OPN	32
	노선이탈리아	OPN	32
	커피앤코드	OPN	32

**SELECT**menu\_name, kilocalories, price, fk\_business\_id **FROM** menus**WHERE**CategoryID **IN** (**SELECT** business\_id **FROM** businesses **WHERE** can\_takeout=0);

내부 쿼리 실행결과

내부쿼리 실행결과 하나이상의 값을 반환하기때문에 = 연산자는 사용 불가능



	menu_name	kilocalories	price	fk_business_id
▶	물냉면	480.23	8000	5
	장국	387.36	8500	15
	비빔냉면	563.45	8000	5
	육개장	423.18	8500	15

	business_id
▶	5
	15
●	NULL



연산자	의미
~ ALL	서브쿼리의 <u>모든</u> 결과에 대해 일치하면 참
~ ANY	서브쿼리의 <u>하나라도</u> 결과에 대해 일치하면 참

```
SELECT * FROM menus
WHERE price > ALL (SELECT price FROM menus WHERE likes >= 10 );
```

내부쿼리 실행결과

내부쿼리 실행결과와 모든 값보다 큰 값을 추출한다는 것은 price의 max 값보다 큰 값을 추출하라는 의미와 같다. Price의 max 값은 12000 이다.



	menu_id	fk_business_id	menu_name	kilocalories	price	likes
▶	4	6	보쌈	1288.24	14000	2
	9	6	마늘보쌈	1320.49	16000	7
*	NULL	NULL	NULL	NULL	NULL	NULL

	price
▶	12000
	9000
	11000
	1500
	5500
	8000



```
SELECT
menu_name, kilocalories, price FROM menus
WHERE
fk_business_id = ANY ( SELECT business_id FROM businesses WHERE status='OPN' );
```

내부쿼리 실행결과의 모든 값 중 하나라도 만족하면 모두 추출하라는 의미 IN과 같은 의미임

내부쿼리 실행결과



menu_name	kilocalories	price
▶ 물냉면	480.23	9000
아메리카노	16.44	5000
고르곤졸라피자	1046.27	12000
보쌈	1288.24	14000
까르보나라	619.11	9000
백순대	681.95	11000
마늘보쌈	1320.49	16000
양념순대볶음	729.17	12000
단팔봉	225.88	1500
간짜장	682.48	7000
비빔냉면	563.45	9000
참치김밥	532.39	3000
치즈떡볶이	638.42	5500
플레인와플	299.31	6500
찰쌀순대	312.76	3000
국물떡볶이	483.29	4950
돈가스김밥	562.72	4000
삼선짬뽕	787.58	8000
수플레팬케익	452.37	9500
라볶이	423.16	5500
모카프라푸치노	216.39	6000
옛날팔빙수	382.35	8000

business_id
▶ 1
2
4
5
6
8
10
11
14
16
17



## 👉 여기서 잠깐 !!

<menus> 테이블, <businesses> 테이블, <sections> 테이블을 이용해 메뉴 가격이 분식 메뉴 가격보다 모두 낮은 메뉴 이름과 가격을 조회하는 SQL문을 아래의 <출력 결과>와 같이 출력되도록 작성하시오.

(단, 비상관 서브쿼리(select~where), ~ALL, JOIN 이용)

출력 결과

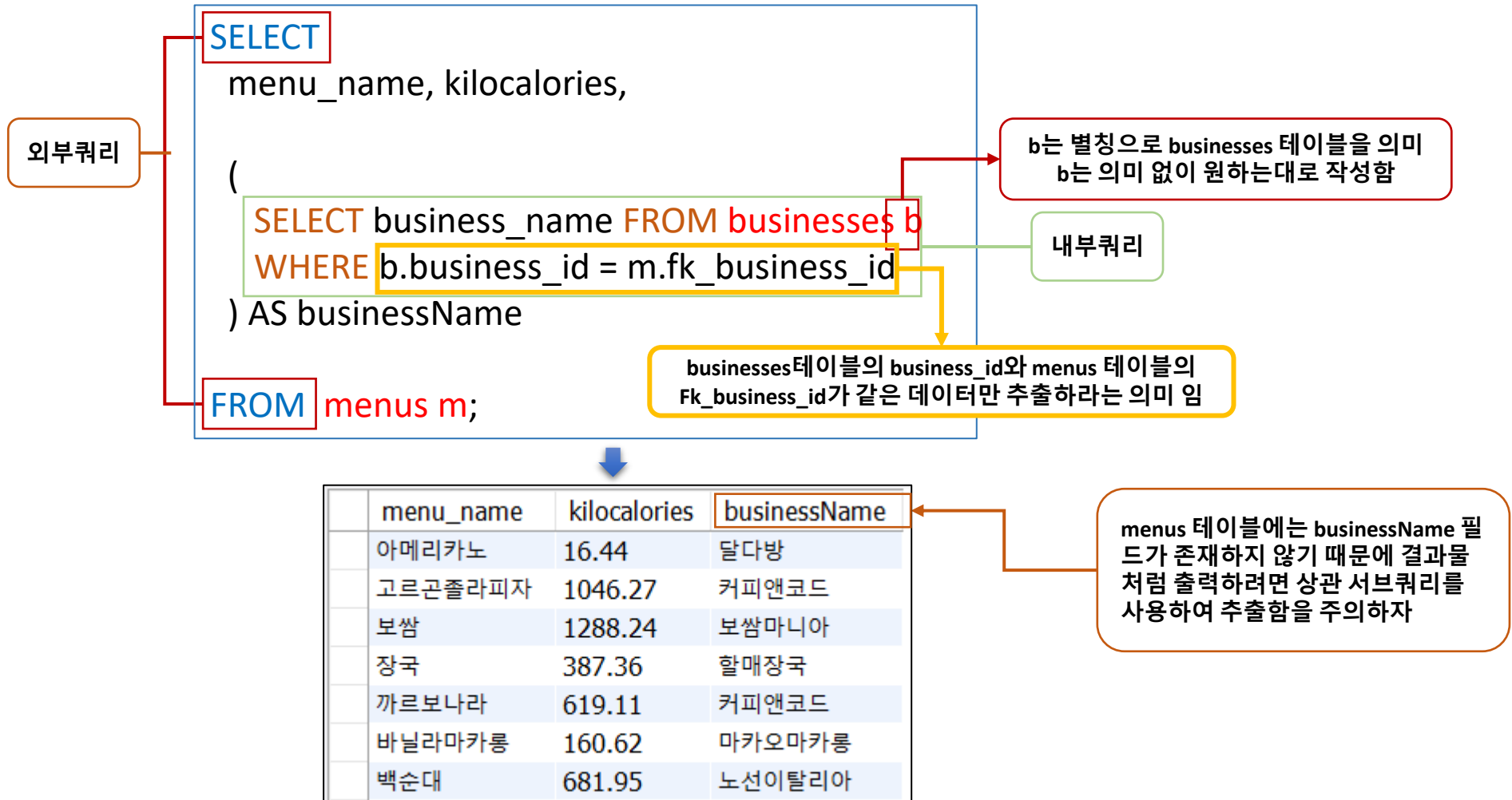
	menu_name	price
▶	바닐라마카롱	1500
	단팥빵	1500
	통카롱	2000





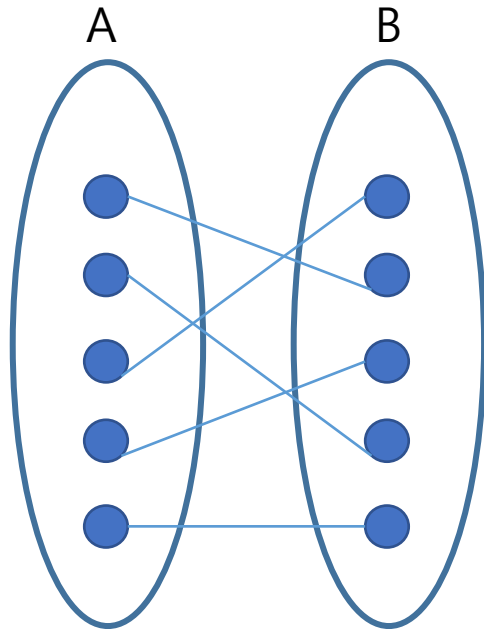
## 2. 상관 서브쿼리

- 내부 쿼리의 값이 결정되는데 **외부 쿼리에 의존한다.**
- 외부 쿼리의 **m.fk\_business\_id** 값이 있어야 내부 쿼리가 수행될 수 있다

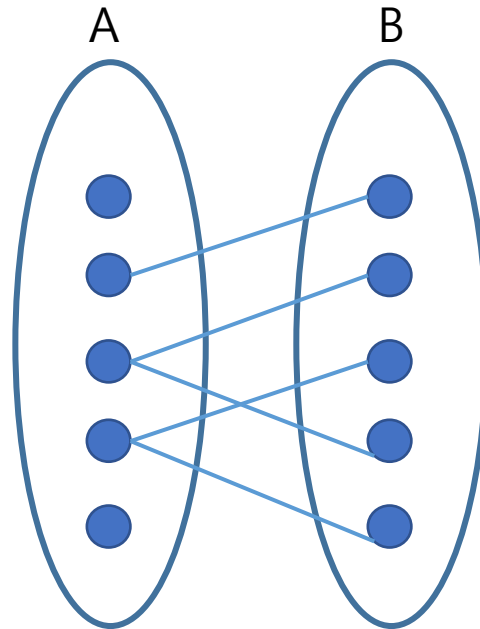




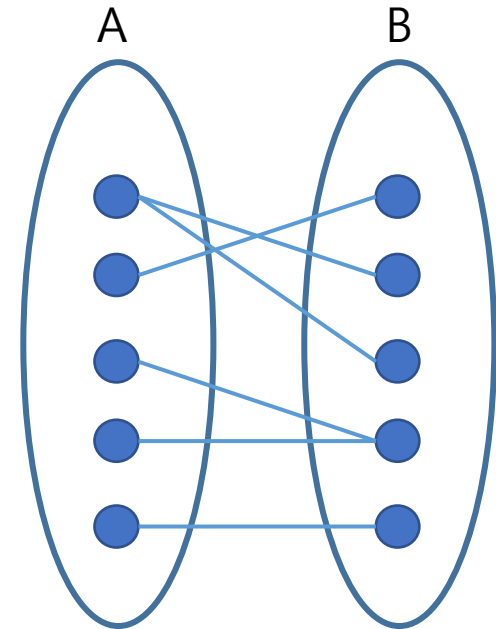
➤ 관계 형태(예 : 개체집합 A, 개체집합 B)



일대일(1:1)



일대다(1:N)



다대다(N:M)



```
SELECT business_name,  
( select menu_name from menus m where m.fk_business_id = b.business_id ) as MenuName  
FROM businesses b;
```



Error in SQL:

Subquery returns more than 1 row

businesses

business_id
10
16
8
13
9
4
6
5
12
11
18
3
7
2
17
14
15
1

menus

fk_business_id
5
8
17
6
15
17
9
16
6
16
14
1
9
5
10
2
11
2
15
4
10
1
11
4
8
14



- 위 SQL문이 오류가 출력되는 이유는 businesses 테이블과 menus 테이블이 1대 다의 관계 형태를 가지고 있다.
- 서브쿼리 결과 1개 이상의 레코드가 출력되기 때문이다.
- 이런 경우는 상관 서브쿼리가 아닌 **Join**을 이용하여 쿼리를 작성한다는 것에 유의하자!!
- 아니면 1대 1 관계는 businesses 는 내부쿼리로, menus 는 외부쿼리로 작성함을 주의하자



연산자	의미
<b>EXISTS</b>	조건을 만족하는 데이터가 <b>존재하면</b> TRUE
<b>NOT EXISTS</b>	조건을 만족하는 데이터가 <b>존재하지 않으면</b> TRUE

```

SELECT
  menu_name, price, fk_business_id
FROM menus m
WHERE EXISTS
  ( SELECT * -- 여기서 *는 무의미함
    FROM ratings r
    WHERE r.fk_business_id = m.fk_business_id
  );

```



	menu_name	price	fk_business_id
▶	물냉면	9000	5
	바닐라마카롱	1500	9
	백순대	11000	16
	양념순대볶음	12000	16
	단팥빵	1500	14
	동카롱	2000	9
	비빔냉면	9000	5
	치즈떡볶이	5500	2
	플레인와플	6500	11
	찰쌀순대	3000	2
	수플레팬케익	9500	11
	옛날팔빙수	8000	14

- 리뷰(ratings)가 하나라도 존재하는 가게의 메뉴를 조회하라
- 즉, "리뷰가 있는 가게의 메뉴만 가져오기"

EXISTS는 서브쿼리에 조건을 만족하는 행이 하나라도 존재하는지만 판단한다.

즉,  
서브쿼리가 1개라도 행을 반환하면 TRUE, 0개면 FALSE  
이 구조이기 때문에 서브쿼리 안의 SELECT 값이 어떻게 생겼는지는 전혀 중요하지 않다.



## ➤ 서브쿼리 사용하는 이유

서브쿼리는 '값을 계산 → 그 값을 기반으로 외부 쿼리 필터링' 하는 데 최적화된 도구  
그래서 다음과 같은 문제에서 매우 유용하다.

- ① 최댓값/최솟값/평균과 비교
- ② 특정 집합과의 비교 (ANY / ALL)
- ③ 특정 조건 존재 여부 확인 (EXISTS)
- ④ 조건에 따라 동적으로 값 결정
- ⑤ JOIN만으로 해결하기 어려운 조건을 처리하기 위해

# 14강

## 연습문제





문제1. 비상관 서브쿼리 (IN)을 이용해 별점(ratings)이 5점인 가게들의 메뉴 목록(menu\_name, price)을 조회하는 SQL문을 작성하시오.

<출력 결과>

	menu_name	price
▶	백순대	11000
	양념순대볶음	12000
	단팥빵	1500
	옛날팥빙수	8000



문제2. 비상관 서브쿼리 (AVG / >)이용해 전체 메뉴 평균 칼로리보다(kilocalories) 높은 칼로리 메뉴를 조회하는 SQL문을 작성하시오.

<출력 결과>

	menu_name	kilocalories
▶	고르곤졸라피자	1046.27
	보쌈	1288.24
	까르보나라	619.11
	백순대	681.95
	마늘보쌈	1320.49
	양념순대볶음	729.17
	간짜장	682.48
	비빔냉면	563.45
	치즈떡볶이	638.42
	돈가스김밥	562.72
	삼선짬뽕	787.58





문제3. 상관 서브쿼리 (EXISTS)를 이용해 리뷰가 존재하는(ratings 존재) 가게(businesses)의 가게 이름과 상태(status)를 조회하는 SQL문을 작성하시오.

<출력 결과>

	business_name	status
▶	철구분식	OPN
	노선이탈리아	OPN
	파시조아	OPN
	북극냉면	OPN
	소소스윗	OPN
	마카오마카롱	RMD



문제4. 상관 서브쿼리 (NOT EXISTS)이용해 리뷰가 하나도 없는 가게(businesses)의 business\_name, fk\_section\_id를 조회하는 SQL문을 작성하시오.

<출력 결과>

	business_name	fk_section_id
▶	화룡각	3
	알코렐라	5
	바른떡볶이	2
	보쌈마니아	1
	에그사라다	5
	달다방	6
	김밥마라	2
	사사서셔소쇼스시	4
	린민짬뽕	3
	할매장국	1
	커피앤코드	6
	신림동백순대	2



문제5. ANY 활용해 동일 가게의 메뉴들 중 가장 비싼 메뉴 가격보다 비싸지 않은 메뉴를 조회하는 SQL문을 작성하시오.  
(다시 말해, 자기 가게 최고가 메뉴의 가격보다 작거나 같은 메뉴)

### <출력 결과>

menu_name	price	fk_business_id
아메리카노	5000	8
고르곤졸라피자	12000	17
보쌈	14000	6
장국	8500	15
까르보나라	9000	17
바닐라마카롱	1500	9
백순대	11000	16
마늘보쌈	16000	6
양념순대볶음	12000	16
단팥빵	1500	14
간짜장	7000	1
동카롱	2000	9
비빔냉면	9000	5
참치김밥	3000	10
치즈떡볶이	5500	2
플레인와플	6500	11
참쌀순대	3000	2
육개장	8500	15
국물떡볶이	4950	4
돈가스김밥	4000	10
삼선짬뽕	8000	1
수플레팬케익	9500	11
라볶이	5500	4
모카프라푸치노	6000	8
옛날팔빙수	8000	14



문제6. ALL 활용해 모든 메뉴보다 칼로리가 가장 높은 메뉴를 조회하는 SQL문을 작성하시오.  
(전체 메뉴 중 최고 칼로리)

<출력 결과>

	menu_name	kilocalories
▶	마늘보쌈	1320.49



문제7. 상관 서브쿼리 (가게별 인기메뉴)이용해 각 가게별로 좋아요(likes)가 가장 높은 메뉴 이름과 좋아요 수를 조회하는 SQL문을 작성하시오.  
(가게별 최고 likes)

<출력 결과>

	menu_name	likes	fk_business_id
▶	고르곤졸라피자	12	17
	백순대	24	16
	마늘보쌈	7	6
	단팔빵	13	14
	통카롱	8	9
	비빔냉면	4	5
	참치김밥	0	10
	치즈떡볶이	15	2
	육개장	2	15
	국물떡볶이	1	4
	돈가스김밥	0	10
	삼선짬뽕	32	1
	수플레팬케익	5	11
	모카프라푸치노	8	8