



SQL EDA

문법 리뷰 : select

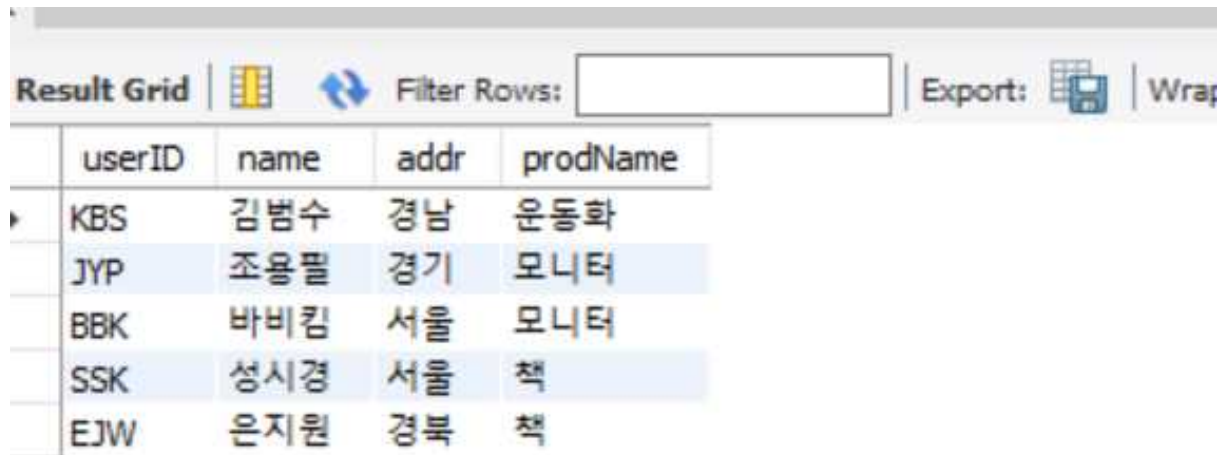
- 컬럼 조회
 - Select 호출하려는컬럼 from DB명.테이블명;
- 집계 함수
 - Select 집계함수 from DB명.테이블명
- * 모든 결과 조회
- AS 특정컬럼 및 결과를 다른 이름으로 변경 → where에서는 사용 못함(having)
- Distinct
 - Select distinct 유니크하게체크하려는컬럼들 from DB명.테이블명
 - 주의!!!!) select distinct col1, col2, col3 from table; 인 경우에 col1만 적용하기 위해서는 distinct가 적용이 안 됨!!! Col1, col2, col3의 조합의 유니크한 것으로 동작을 함
→ 1개 컬럼만 하기 위해서는 group by 로 해야함!!!!
- 이외 같이 사용되는 구문들 : from, where, group by, having, order by, limit etc
- 참고) 문자열을 나타낼 때 "abcd", 'abcd'를 사용을 하고, 컬럼이름에 대해서는 `backtick`을 사용을 한다. 그래서 이 둘이 정확히 사용이 안 되면 원하지 않은 결과가 나오니 늘 이 둘에 대한 사용에 유의할 것!! 단, 컬럼 이름을 나타내는 것에서 `backtick`을 사용을 안 하고, 그냥 컬럼명을 작성을 해도 된다(단, 컬럼 명에 공란이 없을 때!!!) → 컬럼 명에 공란은 무조건 backtick!!!!!!

문법 리뷰 : select

Result Grid				
Filter Rows:				
	userID	name	addr	prodName
▶	BBK	바비킴	서울	모니터
	BBK	바비킴	서울	메모리
	BBK	바비킴	서울	운동화
	EJW	은지원	경북	책
	EJW	은지원	경북	청바지
	JYP	조용필	경기	모니터
	KBS	김범수	경남	운동화
	KBS	김범수	경남	노트북
	KBS	김범수	경남	청바지
	SSK	성시경	서울	책

Distinct 를 해도 userID 컬럼 1개로 걸리지 않음!!!!

문법 리뷰 : select



userID	name	addr	prodName
KBS	김범수	경남	운동화
JYP	조용필	경기	모니터
BBK	바비킴	서울	모니터
SSK	성시경	서울	책
EJW	은지원	경북	책

보고자 하는 컬럼은 여러가지인데,
1개의 컬럼에 대한 distinct를 할 때에는
위와 같이 group by 등을 활용하면 된다!!!!

문법 리뷰 : select + from + where + group by에서 카운팅!!

- 자주 사용되는 경우가 group 별로 묶으면서 카운팅을 해야하는 경우가 있음!!

43 • `select * from buytbl order by userID;`

Result Grid | Filter Rows:

	num	userID	prodName	groupName	
▶	4	BBK	모니터	전자	2
	6	BBK	메모리	전자	8
	10	BBK	운동화	NULL	3
	12	BBK	운동화	NULL	3
	8	EJW	책	서적	1
	9	EJW	청바지	의류	5
	11	EJW	책	서적	1
	3	JYP	모니터	전자	2
	1	KBS	운동화	NULL	3
	2	KBS	노트북	전자	1
	5	KBS	청바지	의류	50
	7	SSK	책	서적	15
*	NULL	NULL	NULL	NULL	NULL

44 • `select userID, count(num) as "구매횟수" from buytbl group by userID;`

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	userID	구매 횟수
▶	BBK	4
	EJW	3
	JYP	1
	KBS	3
	SSK	1

Quiz) sqldb에서 buytbl에서 groupName이 전자인 경우에 대해서 ID별로 구매한 횟수를 구하세요...(구매한 수량이 아니라, 구매한 횟수!!)

			Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	userID	구매 횟수					
▶	BBK	2					
	JYP	1					
	KBS	1					

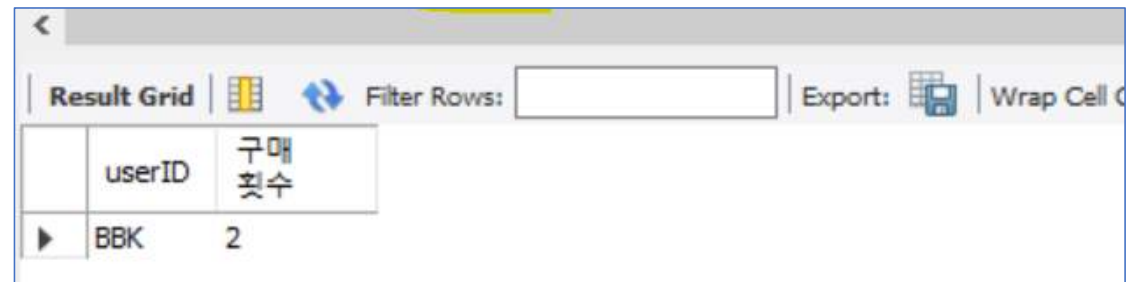
Quiz) sqldb에서 buytbl에서 groupName이 전자인 항목을 구매한 경우에서 userID별로 구매횟수가 3회이상인 사람들만 찾아서, userID와 구매횟수를 출력하세요..

- 이 문제의 포인트는 조건에 대한 필터의 사용!! Where, having에 대한 명확한 구분을 할 수 있는지가 포인트!!!



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	userID	구매 횟수
▶	BBK	2
	JYP	1
	KBS	1



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	userID	구매 횟수
▶	BBK	2

Quiz) 아래 두 쿼리문이 동일한데, 무슨 차이???

- A) having에서 집계 처리된 컬럼 이름이기에 단순 문자열이 아닌 컬럼이름이라고 해야하기에 backtick으로 사용!! Or 그냥 컬럼명이 공백이 없으니 그냥 사용



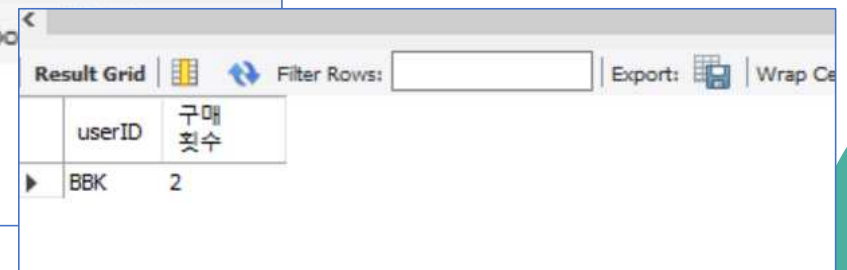
Result Grid | Filter Rows: | Export: | Wrap Cell C

	userID	구매 횟수
--	--------	----------



Result Grid | Filter Rows: | Export: | Wrap Cell C

	userID	구매 횟수
▶	BBK	2



Result Grid | Filter Rows: | Export: | Wrap Cell C

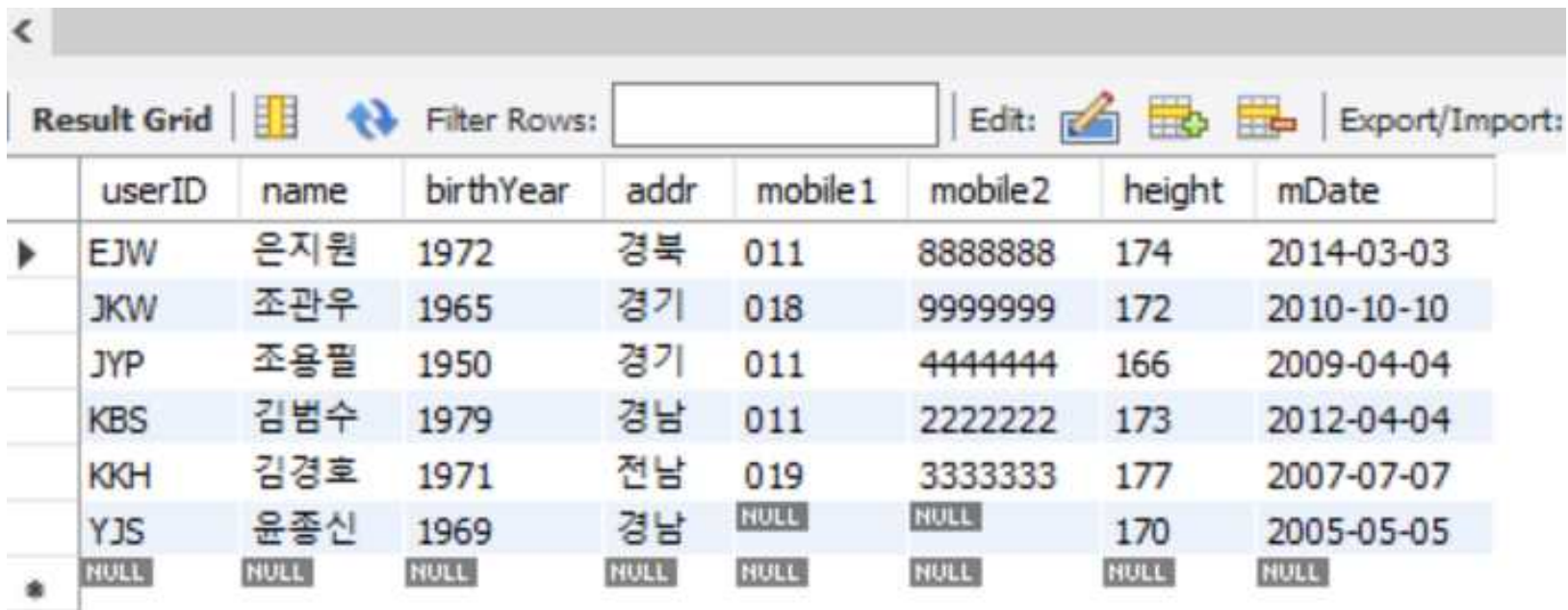
	userID	구매 횟수
▶	BBK	2

Where과 같이 주로 사용되는 연산자들

- Between : 시작점~끝점인 데이터만 출력
 - `Select * from 테이블명 where 컬럼 between 시작점 and 끝점`
- 대소관계 : `=, >, <, >=, <=, !=(<>)`
- In : or의 확장판
 - `Select 컬럼들 from 테이블명 where 컬럼명 in (값1, 값2, etc);`
- Not in : 앞의 in을 제외해서 처리할 때 → 제조국이 미국과 영국 제외하고 볼 때
- Is null : 특정 컬럼의 값이 비어 있는지 체크
- Is not null : 특정 컬럼에서 값이 있는 것들만 볼 때
- Like "%text%" : text가 들어 있는 것을 찾을 때 -> 거주지 주소에 "부산 " 이 들어가 있는 것들 찾을 때 사용!!!
 - `Select * from 테이블 where 주소 like "%부산%";`

Where에서 자주 사용하는 쿼리들

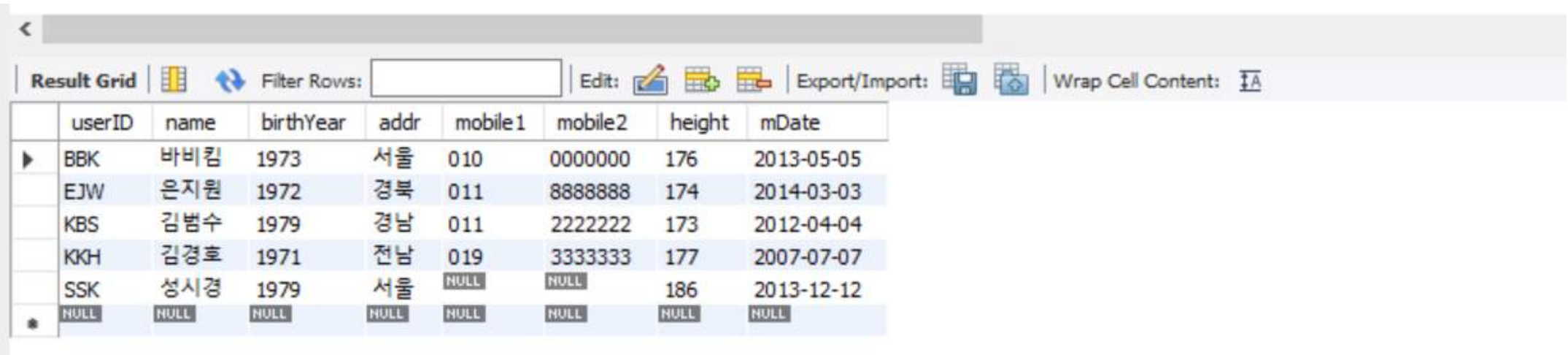
- 특정 값이 아닌 경우만 조회 : sqldb에서 usertbl 테이블에서 지역이 서울을 제외하고 볼 때..(!=, <> 모두 가능)



	userID	name	birthYear	addr	mobile 1	mobile2	height	mDate
▶	EJW	은지원	1972	경북	011	8888888	174	2014-03-03
	JKW	조관우	1965	경기	018	9999999	172	2010-10-10
	JYP	조용필	1950	경기	011	4444444	166	2009-04-04
	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
	KKH	김경호	1971	전남	019	3333333	177	2007-07-07
	YJS	윤종신	1969	경남	NULL	NULL	170	2005-05-05
★	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Where에서 자주 사용하는 쿼리들

- 사이 값 조회 : sqldb에서 usertbl 테이블에서 출생연도가 1970~1980사이인 사용자만 조회 (between)

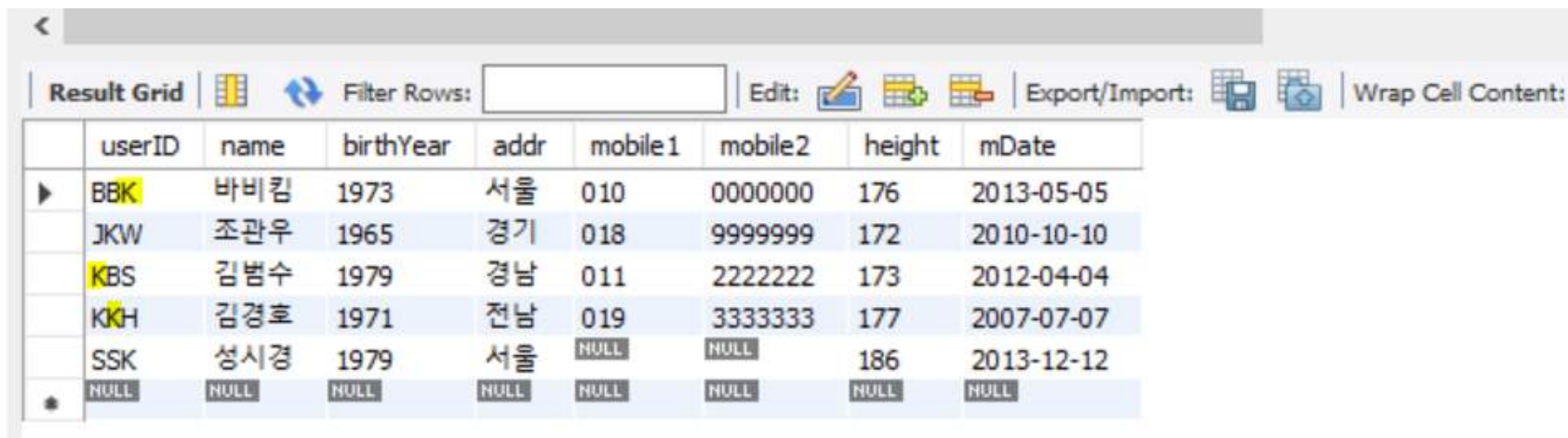


The screenshot shows a database application interface. At the top, there's a toolbar with icons for 'Result Grid', 'Filter Rows' (with a search box), 'Edit', 'Export/Import', and 'Wrap Cell Content'. Below the toolbar is a table with 9 columns: userID, name, birthYear, addr, mobile1, mobile2, height, and mDate. The table contains 6 rows of data, with the last row being a NULL row. The first five rows represent users with various attributes like birth year, address, and mobile numbers.

	userID	name	birthYear	addr	mobile1	mobile2	height	mDate
▶	BBK	바비킴	1973	서울	010	0000000	176	2013-05-05
	EJW	은지원	1972	경북	011	8888888	174	2014-03-03
	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
	KKH	김경호	1971	전남	019	3333333	177	2007-07-07
	SSK	성시경	1979	서울	NULL	NULL	186	2013-12-12
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Where에서 자주 사용하는 쿼리들

- 문자열 속에 특정 문자가 있는지 조회 : sqldb의 usertbl에서 사용자id 중에서 K를 사용하는 사람들은...(like)



The screenshot shows a database query result grid with the following columns: userID, name, birthYear, addr, mobile1, mobile2, height, and mDate. The data is as follows:

	userID	name	birthYear	addr	mobile1	mobile2	height	mDate
▶	BBK	바비킴	1973	서울	010	0000000	176	2013-05-05
	JKW	조관우	1965	경기	018	9999999	172	2010-10-10
	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
	KKH	김경호	1971	전남	019	3333333	177	2007-07-07
	SSK	성시경	1979	서울	NULL	NULL	186	2013-12-12
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Case when

- Case when 구문은 조건에 따른 다른 값을 출력하고 싶을 때 사용.

- 조건1을 만족할 때에는 결과1을 출력하고, 조건2를 만족하게 될 때에는 결과2를 출력하고, 조건1과 조건2를 모두 만족하지 않으면 결과3을 내도록 할 때.
- Select case when 조건1 then 결과1 when 조건2 then 결과2 else 결과3 end from 테이블명;

- 예 : 연령대

- 아래의 경우는 28세는 20대로 출력하고, 54세는 50대로 출력을 하고, 없는 45세는 NULL값이 나타나게 된다!!!
- Select case when 나이컬럼 between 20 and 29 then "20대" when 나이컬럼 between 50 and 59 then "50대" end from 테이블명;

- 예 : 수도권과 수도권 이외 지역 → 수도권 : 서울, 경기 , 비수도권

- Select 지역, case when 지역 in ("서울", "경기") then "수도권" else "비수도권" end as 수도권구분 from 테이블

- Case when 함수 : 필요한 조건만 집계할 수 있음!!!

- 지역 항목 중에서 서울만 집계하는 방법
 - Select sum(case when 지역 = "서울" then 1 else 0 end) as SEOUL from 테이블;
- 지역 항목 중에서 서울의 비율을 집계하는 방법
 - Select sum(case when 지역 = "서울" then 1 else 0 end) as CNT_SEOUL, sum(case when 지역 = "서울" then 1 else 0 end) / count(*) as RATIO_SEOUL from 테이블;

- 예 : 앞의 예제에서 수도권, 비수도권으로 하고, 각기 지역에 거주하는 고객의 수 까지 같이 볼 때.

- Select case when 지역 in ("서울", "경기") then "수도권" else "비수도권" end as 수도권구분,
count(고객) N_CUSTOMER from 테이블



group by case when 지역 in ("서울", "경기") then "수도권" else "비수도권" end

- Select case when 지역 in ("서울", "경기") then "수도권" else "비수도권" end as 수도권구분,
count(고객) N_CUSTOMER from 테이블

group by 1 → 참고로 뒤에 groupby에 있지만, group by에서 숫자로 select 이후 선택한 컬럼을 의미하는 경우가 있음!!! 여기서는 case when으로 처리된 수도권구분이라는 컬럼을 기준으로 묶는다는 의미!!!

Case when 사례

- 예) 나이대와 유사한 년대를 통해서 사례를 보자. Sqlldb의 usertbl에서 출생년도에 대한 세대를 아래와 같이 구분해보자..

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	userID	name	세대
▶	BBK	박비킴	70년대
	EJW	은지원	70년대
	JKW	조관우	60년대
	JYP	조용필	50년대
	KBS	김범수	70년대
	KKH	김경호	70년대
	LJB	임재범	60년대
	LSG	이승기	80년대
	SSK	성시경	70년대
	YJS	윤종신	60년대

Case when 사례

- Sqlldb의 usertbl에서 앞서서와 같이 세대별로 몇 명의 데이터가 있는지 아래와 같이 표시해보자..(문자열, 컬러명에서 ", backtick 등 사용에 유의!!!)



The screenshot shows a database query result grid. The grid has two columns: '세대' (Generation) and '명수' (Count). The data is as follows:

세대	명수
70년대	5
60년대	3
50년대	1
80년대	1

The interface includes a 'Result Grid' tab, a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' toggle.

Case when 사례

- Sqlldb의 usertbl에서 회원들중에서 3명 이상이 존재하는 세대를 찾아보세요!!!!



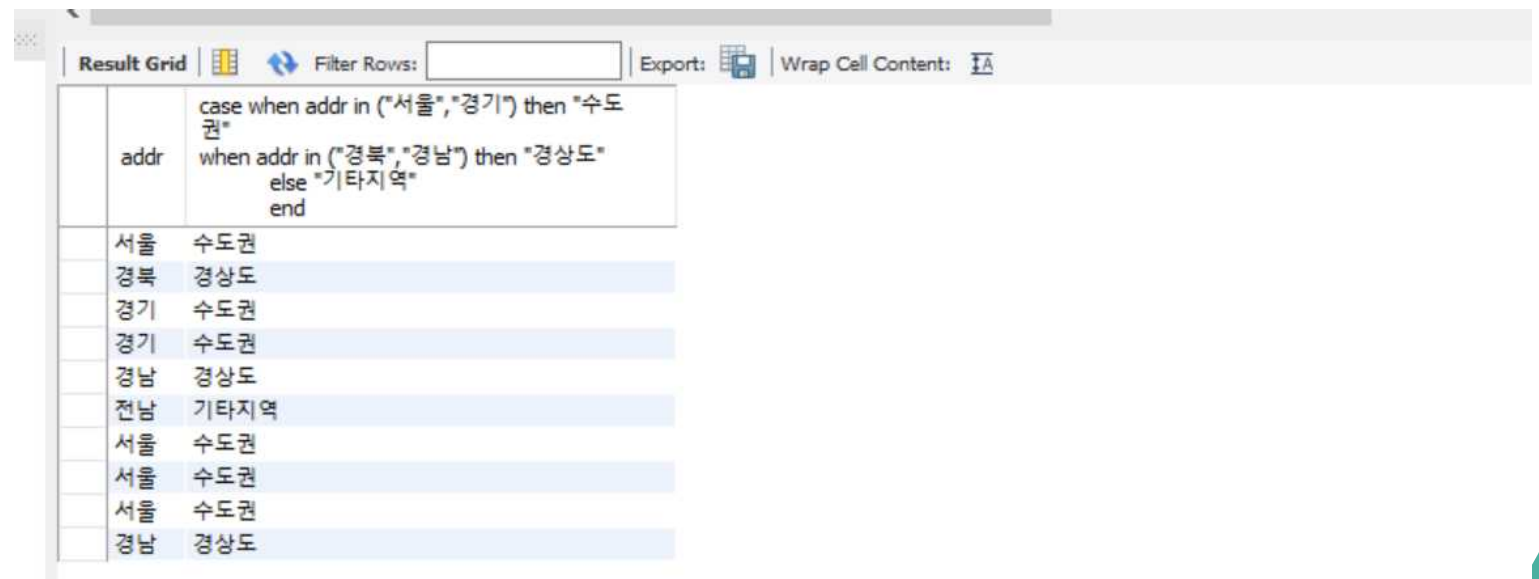
The screenshot shows a SQL query result grid with the following data:

	세대	명수
▶	70년대	5
	60년대	3

The interface includes a toolbar with options like 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'.

Case when 사례

- Sqlldb의 usertbl에서 지역에 대한 정보를 바탕으로 수도권, 경상도, 그외 지역으로 재구분 하세요.

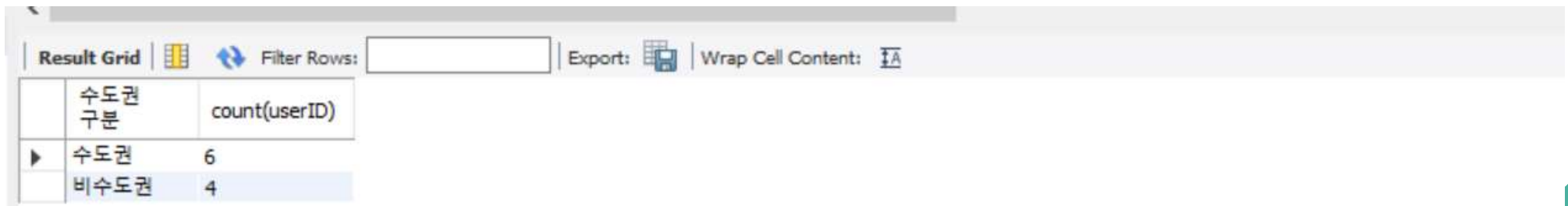


The screenshot shows a database interface with a 'Result Grid' tab. A SQL query is entered in the top bar, and the results are displayed in a table below. The query uses a CASE WHEN statement to categorize addresses into '수도권' (Seoul Capital Area), '경상도' (Gyeongsang-do), or '기타지역' (Other regions) based on the 'addr' column values.

addr	case when addr in ("서울", "경기") then "수도권" when addr in ("경북", "경남") then "경상도" else "기타지역" end
서울	수도권
경북	경상도
경기	수도권
경기	수도권
경남	경상도
전남	기타지역
서울	수도권
서울	수도권
서울	수도권
경남	경상도

Case when 사례

- Sqlldb의 usertbl에서 앞서서와 달리 지역을 수도권/ 비수도권으로 나눠서 그 지역에 해당하는 고객의 수를 같이 표시하세요!!!



Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	수도권 구분	count(userID)
▶	수도권	6
	비수도권	4

Case when에서 1개만 포커싱할때도 사용함!!!!

- Sqldb에서 usertbl에서 지역이 서울인 사람들만 몇 명인지 볼 때..

	userID	name	birthYear	addr	mobile1	mobile2	height	mDate
▶	BBK	바비킴	1973	서울	010	0000000	176	2013-05-05
	EJW	은지원	1972	경북	011	8888888	174	2014-03-03
	JKW	조관우	1965	경기	018	9999999	172	2010-10-10
	JYP	조용필	1950	경기	011	4444444	166	2009-04-04
	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
	KKH	김경호	1971	전남	019	3333333	177	2007-07-07
	LJB	임재범	1963	서울	016	6666666	182	2009-09-09
	LSG	이승기	1987	서울	011	1111111	182	2008-08-08
	SSK	성시경	1979	서울	NULL	NULL	186	2013-12-12
	YJS	윤종신	1969	경남	NULL	NULL	170	2005-05-05
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Group by

- 주로 여러 집계 함수와 같이 사용하게 된다!
- 집계값과 보통 같이 사용되다 보니 조건에 대해서는 having인지 where인지 명
확히 잘 구분해서 해야함!!!
- Group by 에서의 컬럼명은 select에 나열한 컬럼의 순서인 숫자로 표시가 가능
하다!!!!!!

Group by에서 숫자로 기준 표시 예제

- 앞에서 한 예제를 다음과 같이 할 수 있음!!!

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	수도권 구분	count(userID)
▶	수도권	6
	비수도권	4

```
125 • select case when addr in ("서울","경기") then "수도권"  
126         else "비수도권"  
127     end as "수도권구분", count(userID)  
128     from usertbl  
129     group by 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	수도권 구분	count(userID)
▶	수도권	6
	비수도권	4

join

- 내가 필요한 데이터가 여러 테이블에 있을 때 여러 테이블에서 필요한 정보들을 가지고 와야 함.
- Outer join
 - Left outer join
 - Right outer join
 - Full은 없어서 union으로 left outer join + left outer join 해야함
 - Exclusive 조인은 없으니 is null을 활용해서 해야함.
- Inner join

서브쿼리

- In, where 이외에도 from, join 등 여러 곳에서 사용이 된다.
- 주로 많이 사용되는 from, join에 대해서 보자.
 - From에 서브쿼리를 사용하면 서브쿼리의 결과가 하나의 테이블로 사용된다.
 - From, join에 서브쿼리를 사용하는 경우에는 항상 서브쿼리의 마지막에 A와 같은 문자열을 입력을 해야 해당 테이블은 A라는 명칭으로 쿼리 내부에서 사용이 된다!!! (참고 : 오라클은 안 해도 됨;;;π → MYSQL은 꼭 해야함!)
 - Select 고객수 from (select 고객수 from 고객테이블 where 지역 = "서울 ") A
 - Sleect 주문번호 from 주문테이블 where 고객번호 in (select 고객번호 from 고객테이블 where 지역 = "서울 ")

중요사항) sub query 를 from에서 사용할 때 주의!!!

- 서브 쿼리는 값이 올 수 있는 곳 어느곳이라면 다 올 수 있음!
- 다음과 같은 부분에 주로 나타난다.
 - Where 절 : 조건에 사용할 값을 찾는다.
 - Select 절의 필드 목록 : 출력할 값을 찾는다. (join의 대체 표현식이나, 데이터의 양이 많으면 너무 느리게 되어서 거의 사용안함!!!)
 - From 절 : 출력 대상 테이블을 생성한다.
- Where, select 절의 서브쿼리는 둘 다 값을 리턴을 한다. → 그 값이 1개일 수 있고, 아니면 여러 개일 수 있음!!!
- **From 절은 조회 대상 테이블을 명시하는 문장이니 from 다음의 서브쿼리는 테이블과 자격이 같다.**
- From 절의 서브쿼리를 특별히 "인라인뷰 Inline View"라고 부른다.
- 서브쿼리가 FROM 절에 오면 select문까지 중첩이 된다,,,,; select * from (select * from tCity) A;
- FROM절에 있는 안쪽의 select문이 리턴하는 도시 목록은 하나의 테이블이니 결과셋부터 또 select 명령을 실행할 수 있다. 이후 인라인뷰의 필드를 칭하기 위해 당장 쓰지않더라도 병명을 붙여야 한다!!!!!!!

중요사항) sub query 를 from에서 사용할 때 주의!!!

```
146 • select * from (select * from usertbl);
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 56	00:15:42	select * from usertbl	10 row(s) returned	0.000 sec / 0.00
✗ 57	00:22:22	select *, (select userID from buytbl) from usertbl	Error Code: 1242. Subquery returns more than 1 row	0.000 sec
✗ 58	00:23:05	select *, (select userID from buytbl where buytbl.userID = usertbl.userID) from usertbl	Error Code: 1242. Subquery returns more than 1 row	0.000 sec
✗ 59	00:23:47	select (select userID from buytbl where buytbl.userID = usertbl.userID) from usertbl	Error Code: 1242. Subquery returns more than 1 row	0.000 sec
✗ 60	00:24:29	select * from (select * from usertbl)	Error Code: 1248. Every derived table must have its own alias	0.000 sec
✗ 61	00:24:37	select * from (select * from usertbl)	Error Code: 1248. Every derived table must have its own alias	0.000 sec

```
146 • select * from (select * from usertbl) AA;
```

Result Grid



Filter Rows:

Export:



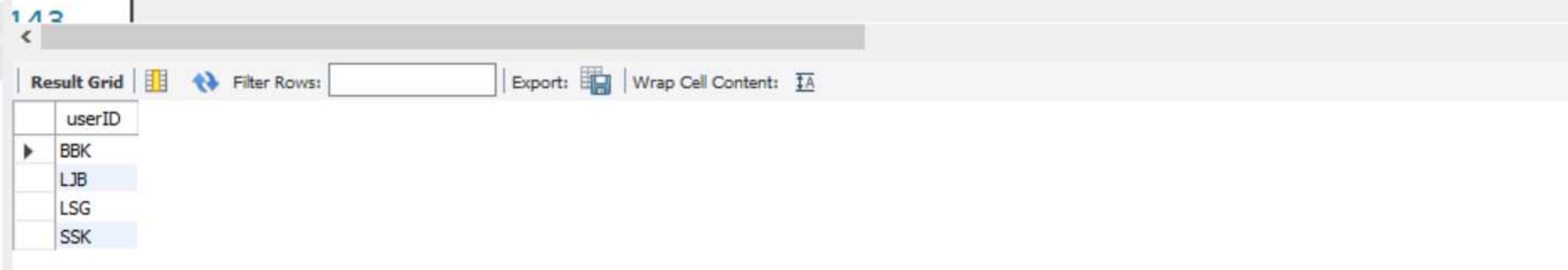
Wrap Cell Content:

	userID	name	birthYear	addr	mobile1	mobile2	height	mDate
▶	BBK	바비킴	1973	서울	010	0000000	176	2013-05-05
	EJW	은지원	1972	경북	011	8888888	174	2014-03-03
	JKW	조관우	1965	경기	018	9999999	172	2010-10-10
	JYP	조용필	1950	경기	011	4444444	166	2009-04-04
	KBS	김범수	1979	경남	011	2222222	173	2012-04-04
	KKH	김경호	1971	전남	019	3333333	177	2007-07-07
	LJB	임재범	1963	서울	016	6666666	182	2009-09-09
	LSG	이슬기	1987	서울	011	1111111	182	2008-08-08
	SSK	성시경	1979	서울	NULL	NULL	186	2013-12-12
	YJS	윤종신	1969	경남	NULL	NULL	170	2005-05-05

중요사항) sub query 를 from에서 사용할 때 주의!!!

- 아래 두 쿼리문의 결과는 동일하다.
- 다만 서브쿼리 형식으로 from 뒤에 테이블 자리에 하면 반듯이 약칭을 해야함!!

```
141 • select userID from usertbl where addr = "서울";  
142 • select userID from (select userID from usertbl where addr = "서울") a;
```



userID
BBK
LJB
LSG
SSK

순위 관련 : rank, dense_rank, row_number + partition by ~

◦ 순위관련

- Row_number() : 동일한 값에 대해 서로 다른 순위 반환
 - (100,150, 200,200,300 -> 1등 2등 3등, 4등 5등)
- Rank() : 동일한 값에 대해 고유한 순위 반환
 - (100,150, 200,200,300 -> 1등 2등 3등, 3등 5등)
- Dense_rank() : rank()와 흡사하지만, 다음 숫자가 이어진다.
 - (100,150, 200,200,300 -> 1등 2등 3등, 3등 4등)
- + 추가적인 옵션으로 partition by
- 용법
 - Over (order by 열 ASC or DESC)
 - Over (partition by 열 order by 열' ASC or DESC)

◦ 순위는 아니지만 어느 영역대인지 : ntile

순위 관련 : rank, dense_rank, row_number + partition by ~

- 참고) <https://doorbw.tistory.com/221>

1. RANK 중복만큼 skip하고 건너 뛴!!!

RANK 함수는 중복 값들에 대해서 동일 순위로 표시하고, 중복 순위 다음 값에 대해서는 중복 개수만큼 떨어진 순위로 출력하도록 하는 함수입니다.

```
SELECT empNo, empName, salary,
RANK() OVER (ORDER BY salary DESC) RANK등수
FROM employee;
```

	empNo	empName	salary	RANK등수
1	1009	최상범	1000	1
2	1004	최상우	600	2
3	1013	한태범	560	3
4	1007	문서연	520	4
5	1008	장웅	500	5
6	1003	장태훈	500	5
7	1010	이명근	500	5
8	1006	송원철	480	8
9	1005	변봉중	450	9
10	1001	최범우	300	10
11	1012	이철진	300	10
12	1011	서은혜	280	12
13	1014	김광우	250	13
14	1002	김범수	250	13

2. DENSE_RANK

중복순위하고, 그 다음은 그 다음 수로 → 수가 중간에 비는 경우가 없음!!!

DENSE_RANK 함수는 중복 값들에 대해서 동일 순위로 표시하고, 중복 순위 다음 값에 대해서는 중복 값 개수와 상관없이 순차적인 순위 값을 출력하도록 하는 함수입니다.

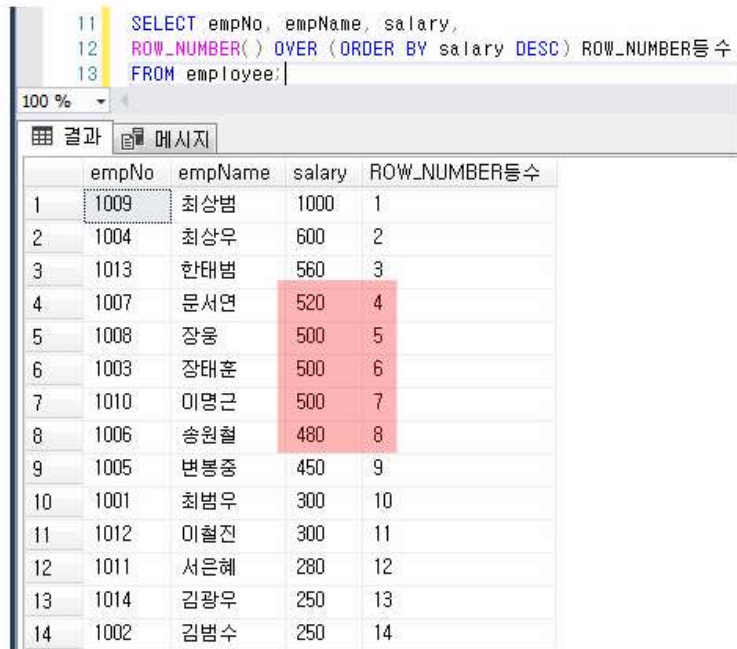
```
SELECT empNo, empName, salary,
DENSE_RANK() OVER (ORDER BY salary DESC) DENSE_RANK등수
FROM employee;
```

	empNo	empName	salary	DENSE_RANK등수
1	1009	최상범	1000	1
2	1004	최상우	600	2
3	1013	한태범	560	3
4	1007	문서연	520	4
5	1008	장웅	500	5
6	1003	장태훈	500	5
7	1010	이명근	500	5
8	1006	송원철	480	6
9	1005	변봉중	450	7
10	1001	최범우	300	8
11	1012	이철진	300	8
12	1011	서은혜	280	9
13	1014	김광우	250	10
14	1002	김범수	250	10

3. ROW_NUMBER

ROW_NUMBER 함수는 중복 값들에 대해서도 순차적인 순위를 표시하도록 출력하는 함수입니다.

```
SELECT empNo, empName, salary,  
ROW_NUMBER() OVER (ORDER BY salary DESC) ROW_NUMBER등수  
FROM employee;
```



	empNo	empName	salary	ROW_NUMBER등수
1	1009	최상범	1000	1
2	1004	최상우	600	2
3	1013	한태범	560	3
4	1007	문서연	520	4
5	1008	장웅	500	5
6	1003	장태훈	500	6
7	1010	이명근	500	7
8	1006	송원철	480	8
9	1005	변봉중	450	9
10	1001	최범우	300	10
11	1012	이철진	300	11
12	1011	서은혜	280	12
13	1014	김광우	250	13
14	1002	김범수	250	14

이것은 기본적으로 row를 기반으로 하고
있어서 동일한 값이 나타나도 순서에 의
해서 서로 다른 수를 부여하고 있음!!!!!!
→ 그냥 줄줄줄


```

SELECT empNo, empName, salary,
RANK() OVER (ORDER BY salary DESC) RANK등수,
DENSE_RANK() OVER (ORDER BY salary DESC) DENSE_RANK등수,
ROW_NUMBER() OVER (ORDER BY salary DESC) ROW_NUMBER등수
FROM employee;

```

```

11 SELECT empNo, empName, salary,
12 RANK() OVER (ORDER BY salary DESC) RANK등수,
13 DENSE_RANK() OVER (ORDER BY salary DESC) DENSE_RANK등수,
14 ROW_NUMBER() OVER (ORDER BY salary DESC) ROW_NUMBER등수
15 FROM employee;

```

100 %

결과 메시지

	empNo	empName	salary	RANK등수	DENSE_RANK등수	ROW_NUMBER등수
1	1009	최상범	1000	1	1	1
2	1004	최상우	600	2	2	2
3	1013	한태범	560	3	3	3
4	1007	문서연	520	4	4	4
5	1008	장웅	500	5	5	5
6	1003	장태훈	500	5	5	6
7	1010	이명근	500	5	5	7
8	1006	송원철	480	8	6	8
9	1005	변봉중	450	9	7	9
10	1001	최범우	300	10	8	10
11	1012	이철진	300	10	8	11
12	1011	서은혜	280	12	9	12
13	1014	김광우	250	13	10	13
14	1002	김범수	250	13	10	14

순위 관련 실습

- Employees 데이터베이스에는 연봉관련 테이블 `sal`과 근로자관련 정보 테이블 `employees`가 있다. → 이것을 바탕으로 `temp`라는 테이블을 만드는데, 근로자사번, 근로자 성, 근로자 이름, 성별, 연봉에 대한 테이블을 만드세요...(근로자 정보 테이블을 기준으로 연봉 정보가 있으면 가지고 와야 함!!)
- 참고) 데이터가 많아서 `with`로 임시로 하기에는 그래서 `temp`라는 테이블에 만들려고 함..

emp_no	salary	from_date	to_date	emp_no	birth_date	first_name	last_name	gender	hire_date
10001	60117	1986-06-26	1987-06-26	10001	1953-09-02	Georgi	Facello	M	1986-06-26
10001	62102	1987-06-26	1988-06-25	10001	1953-09-02	Georgi	Facello	M	1986-06-26
10001	66074	1988-06-25	1989-06-25	10001	1953-09-02	Georgi	Facello	M	1986-06-26
10001	66596	1989-06-25	1990-06-25	10001	1953-09-02	Georgi	Facello	M	1986-06-26
10001	66961	1990-06-25	1991-06-25	10001	1953-09-02	Georgi	Facello	M	1986-06-26

일단 간단히 조인해서 정보들 체크..

이렇게 하면 컬럼 명이 중복이 생기기에 서브쿼리시에 볼 것만 추림..

```
create table temp (select * from salaries S left join  
employees E on S.emp_no=E.emp_no);
```

72	00:33:40	create table temp (select * from salaries S left join employees E on S.emp_no=E.e...	Error Code: 1060. Duplicate column name 'emp_no'
73	00:33:47	select * from salaries S left join employees E on S emp_no=E emp_no	2844047 row(s) returned

앞에서 대략적인 것을 바탕으로 바로 create table로 해서
생성하게 되면, 전체에 대해서 하고, 양쪽 테이블에서 동일
한 컬럼이름이 존재하기에 위와 같이 에러가 발생할 함!!!

따라서 명확하게 정리를 하고 진행을 해야함!!!

Result Grid					
Filter Rows: <input type="text"/>					
	emp_no	first_name	last_name	gender	salary
▶	10001	Georgi	Facello	M	60117
	10001	Georgi	Facello	M	62102
	10001	Georgi	Facello	M	66074
	10001	Georgi	Facello	M	66596
	10001	Georgi	Facello	M	66961
	10001	Georgi	Facello	M	71046
	10001	Georgi	Facello	M	74333
	10001	Georgi	Facello	M	75286
	10001	Georgi	Facello	M	75994
	10001	Georgi	Facello	M	76884
	10001	Georgi	Facello	M	80013
	10001	Georgi	Facello	M	81025
	10001	Georgi	Facello	M	81097
	10001	Georgi	Facello	M	84917
	10001	Georgi	Facello	M	85112
	10001	Georgi	Facello	M	85097
	10001	Georgi	Facello	M	88058

** 이제 이 테이블에서
순위 관련 예제를 해보
려고 함 **

	250701	Satori	Greenwald	140700	60
	44465	Brigham	Teitelbaum	146719	61
	493158	Lidong	Meriste	146703	62
	102962	Chirstian	Kobara	146655	63
	493158	Lidong	Meriste	146588	64
	102962	Chirstian	Kobara	146546	65
	109334	Tsutomu	Alameldin	146531	66
▶	279776	Mohammed	Moehrke	146531	66
	53402	Houman	Worfolk	146507	68
	492164	Ghassan	Birta	146430	69
	263955	Rance	Chinin	146414	70

	102962	Chirstian	Kobara	146655	63
	493158	Lidong	Meriste	146588	64
	102962	Chirstian	Kobara	146546	65
▶	109334	Tsutomu	Alameldin	146531	66
	279776	Mohammed	Moehrke	146531	66
	53402	Houman	Worfolk	146507	67
	492164	Ghassan	Birta	146430	68
	263955	Rance	Chinin	146414	69
	246120	Arnd	Junot	146292	70

	102962	Chirstian	Kobara	146655	63
	493158	Lidong	Meriste	146588	64
	102962	Chirstian	Kobara	146546	65
	109334	Tsutomu	Alameldin	146531	66
	279776	Mohammed	Moehrke	146531	67
▶	53402	Houman	Worfolk	146507	68
	492164	Ghassan	Birta	146430	69
	263955	Rance	Chinin	146414	70

Result Grid							
Filter Rows:							
Export: Wrap Cell Content:							
	emp_no	first_name	last_name	salary	rank	denserank	row_rank
	493158	Lidong	Meriste	146588	64	64	64
	102962	Chirstian	Kobara	146546	65	65	65
	109334	Tsutomu	Alameldin	146531	66	66	66
	279776	Mohammed	Moehrke	146531	66	66	67
	53402	Houman	Worfolk	146507	68	67	68
▶	492164	Ghassan	Birta	146430	69	68	69
	263955	Rance	Chinin	146414	70	69	70
	246120	Arnd	Junot	146292	71	70	71
	66793	Lansing	Kambil	146281	72	71	72

5. NTILE

NTILE 함수도 순위함수로 사용되지만 위에서 사용된 함수들과는 약간 다르게 느껴질 수 있습니다.

NTILE함수는 뒤에 함께 적어주는 숫자 만큼으로 등분을 하는 함수 입니다.

만약 직원들 데이터에 대해서 salary 순서를 기준으로 4등분을 하고자 한다면 다음과 같습니다.

```
SELECT empNo, empName, salary,  
NTILE(4) OVER (ORDER BY salary DESC) NTILE등분  
FROM employee;
```

	empNo	empName	salary	NTILE등분
1	1009	최상범	1000	1
2	1004	최상우	600	1
3	1013	한태범	560	1
4	1007	문서연	520	1
5	1008	장웅	500	2
6	1003	장태훈	500	2
7	1010	이명근	500	2
8	1006	송원철	480	2
9	1005	변봉중	450	3
10	1001	최범우	300	3
11	1012	이철진	300	3
12	1011	서은혜	280	4
13	1014	김광우	250	4
14	1002	김범수	250	4

	emp_no	first_name	last_name	salary	ntile-5
▶	43624	Tokuyasu	Pesch	158220	1
	43624	Tokuyasu	Pesch	157821	1
	254466	Honesty	Mukaidono	156286	1
	47978	Xiahua	Whitcomb	155709	1
	253939	Sanjai	Luders	155513	1
	109334	Tsutomu	Alameldin	155377	1
	109334	Tsu			
	109334	Tsu			
	emp_no	first_name	last_name	salary	ntile-5
	480112	Kazuhide	Frolund	68557	2
	488506	Caolyn	Fortenba...	68557	2
	484477	Renny	Fetvedt	68557	2
	475334	Dietrich	Hiltgen	68557	2
	475563	Tomoyuki	Montemayor	68556	2
	475810	Martial	Rellone	68	
	15830	Yurij	Narwekar	38812	5
	281546	Chuanyi	Kuhnemann	38786	5
	49239	Fumiya	Unno	38735	5
	253406	Olivera	Baek	38623	5

Result 71

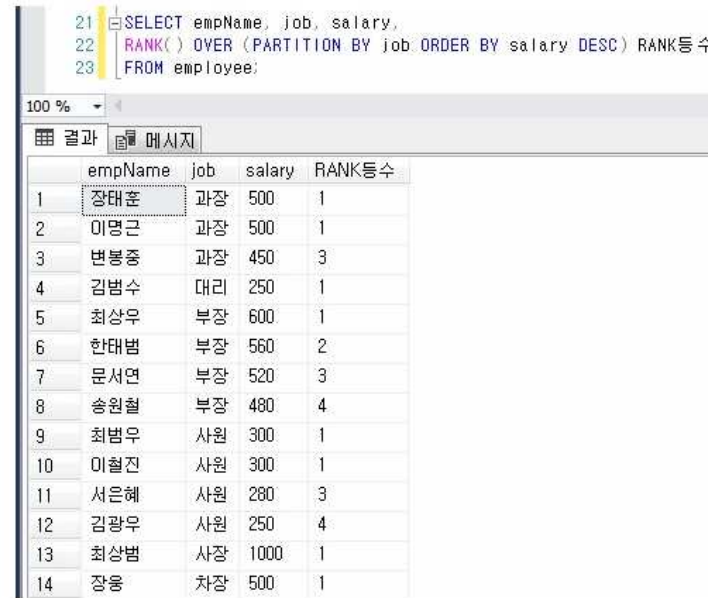
6. PARTITION BY

PARTITION BY 절 또한 어렵지 않습니다.

만약 위와 같은 데이터에서 단순히 모든 사람의 salary를 순위 매기고 싶은 것이 아니라, 직급별 순위를 매기고 싶다면 어떻게 할까요?

직급 별로 구분을 해서 순위를 매기면 됩니다. 이렇게 특정 속성 별로 구분을 하고자 할 때 PARTITION BY절을 사용하면 됩니다.

```
SELECT empName, job, salary,  
RANK() OVER (PARTITION BY job ORDER BY salary DESC) RANK등수  
FROM employee;
```



The screenshot shows a SQL query execution window with the following SQL code:

```
21 SELECT empName, job, salary,  
22 RANK() OVER (PARTITION BY job ORDER BY salary DESC) RANK등수  
23 FROM employee;
```

The results are displayed in a table with columns: empName, job, salary, and RANK등수. The data is sorted by job and then by salary in descending order within each job group.

	empName	job	salary	RANK등수
1	장태훈	과장	500	1
2	이명근	과장	500	1
3	변봉중	과장	450	3
4	김범수	대리	250	1
5	최상우	부장	600	1
6	한대범	부장	560	2
7	문서연	부장	520	3
8	송원철	부장	480	4
9	최범우	사원	300	1
10	이철진	사원	300	1
11	서은혜	사원	280	3
12	김광우	사원	250	4
13	최상범	사장	1000	1
14	장웅	차장	500	1

- 동일한 사번을 가진 사람의 연봉이 높은 순서에 대해서 사번별로 랭킹을 할 때...

	emp_no	first_name	last_name	salary	rank
▶	10001	Georgi	Facello	88958	1
	10001	Georgi	Facello	85112	2
	10001	Georgi	Facello	85097	3
	10001	Georgi	Facello	84917	4
	10001	Georgi	Facello	81097	5
	10001	Georgi	Facello	81025	6
	10001	Georgi	Facello	80013	7
	10001	Georgi	Facello	76884	8
	10001	Georgi	Facello	75994	9
	10001	Georgi	Facello	75286	10
	10001	Georgi	Facello	74333	11
	10001	Georgi	Facello	71046	12
	10001	Georgi	Facello	66961	13
	10001	Georgi	Facello	66596	14
	10001	Georgi	Facello	66074	15
	10001	Georgi	Facello	62102	16
	10001	Georgi	Facello	60117	17
	10002	Bezalel	Simmel	72527	1
	10002	Bezalel	Simmel	71963	2
	10002	Bezalel	Simmel	69366	3
	10002	Bezalel	Simmel	67534	4
	10002	Bezalel	Simmel	65909	5
	10002	Bezalel	Simmel	65828	6