<데이터베이스 실습 복습 정리>

160912~160922(Number 함수~Group 함수)

(문장은 간략하게 작성하는 목적상 소문자로 작성하겠습니다. 구분이 필요할 때만 대문자를 작성합니다.)

4. 숫자, 날짜, 데이터 형변환 함수 at 160912

1) 숫자 관련 함수들

1-1) Round() 함수, Trunc() 함수

이 함수는 초딩때 배운 반올림과 버림과 같다고 생각을 하면 된다. 함수의 구조는 다음과 같다.

Round(숫자 칼럼(혹은 숫자), 자리 수)

Trunc(숫자 칼럼(혹은 숫자), 자리 수)

여기서 원하는 자리 수에서 햇깔리는 경우가 발생하게 된다. 그래서 다음과 같이 문제가 나오는 경우에는 x에 들어갈 값을 자리 수로 써 먹으면 된다.

천의 자리까지만 반올림 출력, 백의 자리에서 반올림, 천의 자리 단위만 반올림 출력

10^{-x} 여기서 천의 자리를 기준으로 하기 때문에 이의 값이 1000을 만족하는 x의 값은 결국 -3이다. 그래서 다음과 같이 작성하면 된다.

Round(808123.101, -3)

이처럼 작성하게 되면 808000이 나오게 된다.(굳이 반올림에 관련된 정의는 다들 알 것으로 판단하고 넘어간다.)

이번에는 소수점이다. 소수점은 버림으로 실행해 볼 것이다. 그래서 다음과 같이 문제가 나오면 x에 들어갈 숫자에 대해 생각을 하면서 풀면 쉽게 풀 수 있다.

소수 둘 째 자리까지만 버림 출력, 소수 셋 째 자리에서 버림, 소수 둘 째 자리 단위 버림 출력

Trunc(808123.101, 2)

이처럼 작성을 하게 되면 소수 둘 째 자리까지만 출력되니 808123.1(본래는 808123.10인데 뒤에 0은 굳이 있을 필요가 없으니 0이 안 나온다.)이 나오게 된다.

1-2) Mod()함수

Mod() 함수는... 그렇다. 나머지 값을 출력하는데 쓰인다. 함수의 구조는 다음과 같다.

Mod(숫자 칼럼(혹은 숫자), 나눌 숫자)

다음과 같이 작성을 해보도록 하자.

Mod(100, 3) 그러면 결과 값은 1이 나오게 된다. 또한 소수로도 작성을 해도 소수를 포함해서 나오니 숙지해두도록 한다.

1-3) Ceil() 함수, Floor() 함수

이는 우리가 고등학교 때 가우스 함수에 대해 배웠을 것이다. 이를 SQL에서 적용한 것으로 생각하면 쉽다. 구조는 달랑 숫자(소수인 숫자면 용이함) 하나만 있으면 된다. 그러나 Ceil 함수와 Floor() 함수에 대해 햇깔리는 경우가 있다. 가령 소수가 123.456이 있다고 가정을 하자. 그러면 123.456은 123과 124 사이에 있다. 123<123.456<124를 이용하게 된다면 Ceil 함수를 입력하게 되면 124가 나오고, Floor 함수를 입력하게 되면 123이 나오게 된다.

Floor(123.45)<123.45<Ceil(123,45) 그래서 두 함수를 햇깔리지 않게 외우기 위해서는 FC 순으로 외우면 편하다. FC의 키워드인 패미컴이나 풋볼클럽, 후라이드 치킨 등 여러분이 알아서 연상하길 바란다.

2) 날짜 관련 함수들

2-1) Sysdate 함수

Sysdate 함수는 쉽게 이야기해서 현재 시간을 호출하는 함수이다. 그냥 아무런 데이터 없이 Sysdate 딱 하나만 작성해주면 된다. 여기서 하나 더 알아둔다면, Sysdate는 dual 테이블에서만 쓰는 것이 아니라 모든 테이블에서 적용이 되는 것을 숙지해 두길 바란다.(모든 테이블에도 현재 시간이 필요로 하는 경우가 있기 때문이라고 생각하자.)

<추가> 또한 Session을 바꿔서 날짜의 출력 방식에 대해 교체를 할 수 있다. 다음과 같은 문장을 참조해 보도록 하자.

Alter Session

set NLS_Date_Format='YYYY.MM.DD. HH:MI:SS';

이는 먼저 작성을 하고 마지막에 ;(세미콜론)으로 끝내줘야 한다. 그대로 작성하게 되면 에러가 걸리게 된다. 여기서 NLS는 National Language Service(국가별 언어 지원)의 줄임말이다. Date_Format는 날짜 형식으로 생각하면 된다. 예를 들어 대한민국의 언어를 기준으로 해서 날짜를 출력하게끔 해주는 기능이라고 생각하면 된다.(1994년 뭐월 무슨일 뭐요일 이런 식으로)

2-2) Months_Between 함수

이는 날짜와 날짜 사이의 개월이 몇 개월 차이인지에 대해 출력하는 것이다. 그러나 이는 소수점으로 나오는 경우가 존재한다. 왜냐 같은 1개월이라도 28일, 29일, 30일, 31일 이렇게 존재하는 것으로 간주하고 계산이 되기 때문이다. 그래서 어느 기법에 따라서 반올림이나 버림을 해주길 바란다.

함수의 구조는 다음과 같다.

Months_Between(최근 날짜, 옛날 날짜)

Months_Between(sysdate, TO_DATE('2014.09.18.', 'YYYY.MM.DD'))

이처럼 작성을 하게 되면 오늘 날짜가 2016년 9월 18일인 것으로 가정을 하면 결과 값은 24가 나오게 된다.

2-3) Add_Months 함수

Add_Months(날짜, 개월 수)

개월 수에 숫자를 입력하게 되면 그 개월만큼 지난 날짜에 대해 출력을 하는 함수이다. 구조는 간단하나 만일 12월 31일이 2개월 지난 뒤에 출력되는 날짜가 무엇인지에 대해 궁금해 할 것이다. 뭐 2월 31일은 없으니깐 결국 출력되는 날짜는 2월 28일이다.(윤달에 따라 다르다.)

2-4) Next_Day 함수

비와이의 Day Day라는 노래를 들어 봤을 것이다. 뭐 연관성은 없지만, 디비에서 Day는 요일을 뜻하는 키워드이다. 한국어를 기준으로서는 무슨 요일로 구성이 되어 있다.

함수 구성은 다음과 같다.

Next_Day(날짜컬럼/날짜, 요일(괄호 처리 꼭))

가령 작성한 날짜가 9월 14일 수요일이라고 가정을 하자.

일	월	화	수	목	금	토
....	12	13	14	15	16	17
18	19	20	21	22	23	...

그래서 첫 번째 경우는 Next_Day를 이용해서 다음 금요일은 언제인지에 대한 출력, 또 하나는 다음 화요일이 언제인지에 대해 출력하는 함수를 작성해 보겠다.
Select To_char(Next_Day(sysdate, '금요일'), 'YYYY/MM/DD')
이처럼 작성하게 되면 결과 값은 금주 금요일인 9월 16일이 나오게 된다. 허나 금요일을 화요일로 바꾸면 다음 주 화요일인 9월 20일이 출력하게 된다.

2-5) Last_Day() 함수
이는 그 달에 해당된 마지막 날을 출력하는 함수이다. 가령 오늘 날짜가 12월 25일(무려 성탄절)일 때 Last_Day(sysdate)를 해주면 출력되는 값은 12월 31일, 연말로서 제야의 종소리를 들으러 갈 준비를 하면 된다.

2-6) Round, Trunc 날짜에서 적용하기
이도 Last_Day처럼 날짜라는 변수 하나만 있으면 된다. Round는 정오를 기준으로 오전이면 오늘, 오후이면 내일을 출력하는 개념이고, Trunc는 시간이 어찌됐든 무조건 오늘을 출력하는 함수이다.
가령 sysdate가 2016년 10월 10일 오후 2:30이라고 가정을 하자.
Round(Sysdate) 이렇게 작성하면 2016년 10월 11일 오전 00:00으로 나오게 된다.
Trunc(Sysdate) 는 2016년 10월 10일 오전 00:00으로 나오게 된다.

3) 데이터 형 변환 함수
이는 디비 개론때도 잠깐 숙달을 한 적이 있다. 그래서 이에 관련된 개념은 잠시 복습하는 차원해서 공부해봤으면 좋겠다.

3-1) To_Char 함수
이는 가령 날짜를 출력하는데 국가별 언어 서비스에서 날짜 형식에 따라서 출력하는 것을 무시하는 경우에 쓰거나 숫자에 소수점이나 달러, 쉼표를 추가할 때 쓰는 함수이다. 숫자를 쓰는 경우, 날짜를 쓰는 경우로 나누어서 알아보자.

3-1-1) 숫자를 문자열로 변환하는 경우
숫자를 문자열로 변환을 할 경우에는 다음과 같은 기호들을 참조해보자.
To_Char(숫자/칼럼, 형식)
999999 : 9가 자리 수만큼 채워주는 역할을 해준다. 만일 예를 들어 999999 형태로 출력할 때 숫자 5400을 출력하면 이상 없이 5400이 출력하게 되고, 1234560을 출력하게 된다면 #######(9 입력 개수의 하나 더)를 출력하게 되고, 음수도 같이 적용된다.
099999 : 0을 붙어주게 되면 빈 공간에 대해서는 0으로 지정하게 된다. 물론 그 이상을 입력하게 되면 똑같이 #######을 출력 하는 건 다 알거라 믿는다.
$: 앞에 달러를 붙어준다. 앞에만 붙을 수 있다. 뒤에는 못 붙인다...(물론 띄어쓰기도 해선 안 된다.)
. : 소수점 이하 표시. 9999.999로 설정해둬서 920.32를 표현하면, 920.320이 출력된다.
, : 천 단위 끊어주기. 99,999 형식으로 1000을 출력하면 1,000으로 나오게 된다.

3-1-2) 날짜를 문자열로 형 변환하기
TO_Char(날짜/날짜칼럼, 형식)
형식에는 다음과 같은 요소들을 이용해서 적용을 할 수가 있다.
YY/YYYY : 년도. 쉽게 이야기해서 그냥 년도다.
RR/RRRR : 이것도 연도인데, 가령 현재 년도(2016년)에 RR이 04인 경우에는 2004년이 되고, RR이 94인 경우에는 1994년을 가리키게 된다. 아니면 과거 년도(1995년)인 경우에는 RR이 04이면 2004년이고, RR이 98이면 1998년이 된다.
Year : 연도의 영문 이름. 1994년이면 Nineteen Ninety-Four가 출력됨.

MM : 2글자로 월을 표현. 9, 10, 11....
MON : 영어 3글자로 원을 표현 Sep, Oct, Nov....
MONTH : 풀 영어로 표시한다. September, October, November....

DD : 2글자로 일을 표현. 10, 20, 30
DAY : 요일을 표현. 한국은 월요일, 화요일 이런 식으로 출력된다.
DDTH : 10th, 11st, 12nd 이런 식으로 일을 표기한다. 참고로 미국이나 영국가면 날짜의 일을 이렇게 표현하니깐 그냥 이해하자.

HH24 : 24시간으로 시를 표현
HH : 12시간으로 시를 표현
AM, PM : 그냥 붙여주면 오전이면 AM, 오후면 PM을 출력. 아무거나 붙이고 가도 상관 없다.
MI : 분, 00~59 // SS : 초, 00~59

3-1-3) TO_DATE, TO_NUMBER 함수
이는 문자열을 날짜나 숫자로 다시 반환시켜주는 함수이다. 그래서 다음과 같이 이용을 할 수 있으니 참고하자.
TO_DATE(날짜/칼럼, '형식')
이는 날짜를 서로 비교할 때 많이 쓰이니 숙지하고 있으면 좋다. 비록 예를 들어 날짜가 2010년 3월 1일 이전에 입사한 사람을 뽑는 문장에 대해 소개하겠다.
```
Select employee_id, last_name
from employees
where hire_date<TO_DATE('2010/3/1',
 'YYYY/MM/DD')
```
이처럼 비교할 때 날짜를 임의로 설정하였을 때 날짜를 다시 비교하게끔 도와주니 참조하자.

또한 TO_NUMBER로 문자열을 숫자로 다시 바꿔준다. 가령 TO_NUMBER('10000')을 입력하면 다시 숫자가 10000이 뜨게 되는 것으로 종결되는 것이다.

(추가) 날짜와 날짜 사이를 뺀다면??
날짜와 날짜를 빼면 무슨 값이 나올까? 필자가 제작년 중간고사 문제를 푸면서 깨달았었는데 날짜와 날짜 사이를 빼면 그 사이에 지나간 일수를 출력하게 된다. 가령 다음과 같은 문장을 작성해보자.
```
Select Sysdate-Hire_Date "날짜일수차이"
from employees;
```
이처럼 작성하게 되면 소수로 블라블라하게 나오지만, 반올림이나 버림을 해주면 적당하게 나오니 이를 나누어서 7로 나누면 주단위 근무기간, 30으로 나누면 월단위 근무기간, 365로 나누면 연간 근무기간으로 적용을 할 수 있으니 제작년 중간고사 5번 문제를 풀 때 참고하길 바란다.

<참고> 묵시적 형 변환
숫자를 문자열로 하였을 경우에는 자바에서는 서로

다르게 생각했을 것이다. 그러나 SQL에서는 그렇지 않다. 2이든 '2'이든 둘 다 같은 값으로 인식을 하게 된다. 이것이 바로 묵시적 형 변환이다. 이전 버전에서는 이런 건 없었는지 모르지만, 버전이 업데이트되면서 추가 되었다고 보면 된다. 그래서 2+'2'를 하면 22가 아닌 4가 나오게 된다.

---

5. 다양한 일반 함수들 at 160919

1) NVL(), NVL2() 함수

NVL은 Not Value인 경우에 다시 값을 추가하게끔 해주는 역할을 해준다. 그래서 가령 예를 들어 봉급이 Null인 사람을 0으로 설정하는 경우에는 다음과 같이 해주면 된다.

```
Select NVL(salary, 0)
from employees;
```

그러면 봉급이 null인 사람은 0으로 설정이 된다.

NVL2인 경우에는 다음과 같이 설정하게 되면 값이 존재하면 밑줄 친 쪽의 왼쪽, null인 경우에는 밑줄 친 쪽의 오른쪽을 출력하게 된다.

```
Select NVL(salary, '봉급숙지함', '봉급을모름')
from employees;
```

참고로 이는 차후에 배울 Group by 함수에서 AVG 함수를 다룰 때 써먹을 테니 기억해두길 바란다.

2) Decode 함수

Decode는 우리가 자바에서 썼던 IF문과 같은 개념이지만 주의할 점은 그 값이 정확한 것들만(예를 들면 문자열이나 직책번호 등등) 적용이 가능하다. 월급이 10000이상이나 그런 것들은 Case 문을 이용하도록 하고 어떻게 이용하는지에 대해서 경우를 나누어 설명하겠다.

2-1) 학년이 1학년인 경우 신입생, 아니면 재학생을 출력하기(Grade는 Varchar2 데이터로 친다.)

```
Decode(grade, '1학년', '신입생', '재학생')
```

이 문장은 가령 예를 들어 grade가 1학년인 경우에는 신입생을 출력하는 것이고, 아닌 경우에는 재학생을 출력하는 것이다.

이 Decode문의 구조는 다음과 같다.

```
Decode(컬럼이름, 확인할 문자열, 맞으면, 틀리면)
```

틀리면에 null로도 지정이 가능하다. 물론 틀리면을 안 써도 알아서 null로 지정이 된다.

2-2) 학년이 1학년인 경우 신입생, 2학년이면 열공생을 출력하기(Grade는 number 데이터로 친다.)

```
Decode(Grade, '1학년', '신입생', '2학년', '열공생', '고학년')
```

이 문장은 쉽게 이야기 하면 grade가 1학년이면 신입생, 2학년이면 열공생, 그 이외이면 고학년을 출력하게끔 해준다. 조건을 중복해서 다는 경우엔 구조가 햇갈리게 된다. 다음과 같은 구조로 숙달을 하도록 하자.

```
Decode(컬럼이름, 확인할 문자열, 맞으면, 다른 확인할 문자열, 이것이 맞으면, 아무 것도 안 맞으면)
```

2-3) 위 조건을 중첩해서 사용하기(Semi는 학기이다.)

```
Decode(Grade, '2학년', Decode(Semi, '1학기', '자바', '운영체제'))
```

이 문장은 쉽게 이야기하면 학년이 2학년인 사람들 중에서 학기가 1학기인 사람이면 자바, 1학기가 아닌 사람은 운영체제를 뽑아내는 것이다. 이는 Grade가 2학년인 사람만 시행하는 문장으로 생각하면 된다.

(쉽게 숙달하기) Decode 문은 생각보다 많이 외우기 까다롭다. 그래서 다음과 같은 방법으로 숙지를 해두는 것을 추천한다. 아니면 굳이 Decode 문보다 Case When Then 문이 있으니 이걸 더 사용해 보는 것도 나쁘진 않다고 생각한다.

```
Decode(컬럼, (조건, 맞으면), (조건, 맞으면), ... , (조건, 맞으면), 아무 것도 아니면)
```

3) Case 함수

SQL계의 진정한 IF문이다. 위의 Decode 문은 정확한 값에 관련해서 다루곤 하였지만, Case 문은 조건을 통해서 분류를 해주는 큰 역할을 해주기 때문에 꼭 알아두길 바라는 문장이다. 문장의 구조는 다음과 같다.

```
Case When 조건1 Then 결과물
     When 조건2 Then 결과물
...  Else 아무것도 아닐 때 결과물 End "컬럼명"
```

이처럼 작성을 해주게 되면 컬럼명은 그대로 컬럼명이 될 것이다. 컬럼명을 생략하게 되면 컬럼의 이름이 진짜로 부산행이 될 것(즉 컬럼 이름이 장난 아니게 길어진다는 뜻)이니깐 써주는 편을 추천한다. 다음 문장을 살펴보자.

```
Case When Salary Between 1000 and 2000
      Then '신입사원'
      When Salary Between 2000 and 10000
Then Case When Salary Between 2000 and 5000
           Then '보통사원'
          When Salary Between 5000 and 8000
            Then '되는사원'
          When Salary Betwenn 8000 and 10000
            Then '슈퍼사원' End
      Else '하이퍼사원' End "사원분류"
```

이 문장을 살펴보게 되면 처음에 봉급이 1000에서 2000 사이인 사원이 바로 신입 사원이 되는 것이다. 그러나 다음 조건이 2000에서 10000 사이 봉급을 받는 사원들은 또 세 단계로 나뉘어서 2000~5000은 보통사원, 5000~8000은 되는사원, 8000~10000은 슈퍼사원이 되고, 이외에 10000을 넘기는 사원은 결국 하이퍼사원을 출력하게 된다. 이도 마찬가지로 Decode 문처럼 중첩이 된다는 사실을 알아두길 바란다. 이를 이용해서 여러 정보들의 분류를 적극적으로 활용하는데 큰 도움이 될 수 있다는 점을 알아두도록 하자.

4) 복수 행 함수

복수 행 함수는 하나의 튜플을 다루는 것이 아니라 여러 개의 튜플을 집계를 내는 경우나 통계를 내는 경우가 대다수이다. 그래서 복수 행 함수에 관련해서 알아둬야 할 필요가 있다.

4-1) Count() 함수

Count(*)를 이용하게 되면 조건이나 전체에서 해당되는 튜플들의 개수에 대해 출력하는 함수이다. 또한 * 에 튜플이 들어갈 수가 있다. 그래서 다음과 같이 작성했을 때 차이에 대해 숙지해 둘 필요가 있다.

```
Select Count(commission_pct) // 1 결과는 35
, Count(NVL(Commission_Pct, 0)) // 2 결과는 107
from employees;
```

1번과 같이 작성하게 되면 사원 중에 보너스 비율이 존재하는 사원들의 수만 출력을 하게 된다. 2번처럼 작성하게 되면 사원들 중에서 보너스 비율이 없다면 0으로 설정을 하게끔 해주고 난 뒤에 실행하는 함수로서 전체의 개수가 나오게 된다.

4-2) Sum() 함수
Sum()은 말 그대로 합계이다. Sum(합계를 낼 컬럼)이 처럼 작성해서 쓰면 된다. 만일 사원들 중에서 부서 번호가 60번에서 90번까지인 사원들의 월급의 합을 알고 싶다면 다음과 같이 작성하면 된다.

```
Select Sum(Salary)
From Employees
Where Department_ID Between 60 And 90;
```

4-3) Avg() 함수
AVeraGe(평균)의 개념이다. 원래 Group함수는 값이 없는 것은 예외하고 뽑아가는 원리라서 지금처럼 Avg(commission_PCT)를 쓰면 보너스 비율이 있는 사람들로만 계산이 되는 개념으로 보면 된다. 그렇지만 AVG(NVL(commission_pct, 0))을 이용하게 되면 보너스 비율이 없는 사람은 0으로 간주하고 총 평균을 구하게 된다. 그래서 보너스 비율을 있는 사람은 그대로, 없는 사람은 0으로 간주하고 쓰고 싶다면 NVL 라는 함수를 이용해보도록 하자.

4-4) Max(), Min() 함수
말 그대로 최댓값, 최솟값을 매기는 함수이다. 그렇지만 애석하게도 가장 느린 함수이기 때문에 인덱스를 같이 이용해서 이 함수를 이용해 보는 방안도 좋은 방법이다. 인덱스는 차후에 어떻게 쓰는지에 대해 알아보되, Max(칼럼), Min(칼럼)을 하면 숫자면 뻔하게 최댓값, 최솟값이 나오는건 알지만... 날짜인 경우는 Max(날짜컬럼)은 최근 날짜가 나오게 되고, Min(날짜컬럼)은 오래된 날짜가 나오게 된다.

5) Group By 절
물론 방금 위의 함수는 Group By 절이 없더라도 전체를 기준으로 사용하면 작동하는데 이상은 없다. 그렇지만 Group By로 묶어서 풀이를 하게 된다면 결과는 달라진다. 하나로도 묶을 수 있고, 둘 이상으로도 묶을 수 있다. Employees의 Department_id, Job_id 2개로 설명을 하겠다.

```
select Department_id, job_id, Sum(Salary), count(*)
from employees
group by Department_id, job_id
Order by 1, 2;
```

이처럼 작성하게 되면 부서별 번호대로 나뉘어서 또 직책별로 나뉘어서 통계를 내주는 역할을 해준다. 허나 애석하게도 Group By는 직접 정렬하는 개념이 아니라 알아서 정렬하는 개념이다. 그래서 정렬은 마지막에 Order by로 해결하도록 하자.

<Group By 절 사용 시 주의사항>
- Select 절에 사용된 그룹 함수 이외에 컬럼, 표현식은 반드시 Group By 절에 사용되어야 한다. 그니깐 쉽게 이야기해서 Select Emp_id, Job_id, Sum(Salary)처럼 되었지만 마지막 Group By 절에서 Group By Emp_id만 작성하게 되면 오류가 난다고 생각하면 된다.
- Group By절에 이용된 컬럼은 Select 절에 사용하지 않아도 상관은 없다. 그러나 결과에 대해서는 장담을 못한다.
- Group By 절에는 컬럼명만 이용이 가능하다. Order By 절에서 Order By 1, 2처럼 사용 하거나 Alias에서 별도로 지어낸 컬럼명으로 Group By를 쓴다고 생각하면 절대로 안 된다.

6) Having 절
우리가 Where 절에서는 컬럼에 관련해서 다루게끔 하였다. 그렇지만 Where절과 Having 절의 차이를 숙지해둬야 한다. 어떤 차이인지에 대해 알아두자.
Where : 컬럼에 관련해서 쓰는 조건문이다. 허나 Group 함수에 관련되어서는 쓸 수가 없다.

Having : 일번 컬럼에 관련해서 쓰는 조건문도 된다. 또한 Group 함수에 관련되어서도 쓸 수가 있다.

물론 숙지해둬야 하는 점은 조건문 절은 Group By 이전에 작성이 된다는 점을 햇깔려서는 안 된다. 여기서 Where절과 Having 절을 통해서 알아보도록 하자.

6-1)

```
Select department_ID, Count(*)
from employees
where last_name Like '송%'
Group By Department_id;
```

이처럼 작성을 하게 된다면 쉽게 이야기해서 성이 송씨인 사람을 부서별로 세어서 결과를 출력하는 문장이라고 생각하면 된다. 아니면 이번에는 다음 문장을 살펴보자.

6-2)

```
Select Department_ID
from employees
where SUM(Salary)>=10000(X)
group By Department_id;
```

이처럼 작성하게 되면 어디서 오류가 걸리는지는 다 숙지하였을 것이다. Where절에는 절대로 Group By 함수가 들어올 수가 없다는 점을 잊어서는 안 된다. 그래서 여기서는 Where를 Having으로 바꿔서 써야 된다는 점을 알아두자.

7) Cube, Rollup, Grouping Sets 함수
여기서는 간략히 작성하겠다. 다만 Grouping 함수 실습을 참조해보길 바란다.
Cube 함수는 모든 컬럼에 대해서 각자 참조하게끔 도와주는 함수이다. 비록 다음과 같이 함수를 적용하게 되면 어떻게 될까?

Cube(Dep_id, job_id)
이처럼 작성하게 되면 부서 내 직급 별/부서 별/직급 별/전체 참조 형식으로 출력을 하게 된다.

Rollup(Dep_id, Job_id)
이 함수는 밑줄 친 것은 살려두고(즉 밑줄 친 값의 Grouping_id를 처음에는 0으로 해 두고 뒤의 Grouping_ID가 모두 1인 경우 마지막에 전체 출력을 위해 1로 변환해서 묶어준다고 생각하면 된다.) 마지막에 죽여둔다고 생각하면 쉽다. 결과는 부서 내 직급 별, 부서 별, 전체 이렇게만 참조된다고 생각하자.

Grouping Sets(Dep_id, Job_id)
이렇게는 각 컬럼 별로만 통계를 낸다고 생각하면 된다. 쉽게 생각해서 부서 별 통계, 직급 별 통계 이렇게민 낸다고 생각하면 된다.

위 함수에 관련된 자세한 문장에 대해 숙지하고 싶다면 필자가 작성한 160921_Grouping함수도 또한 보내줬으니 이를 참조하도록 하자.

8) Listagg 함수(11G부터 가능합니다.)
Listagg(컬럼, 구분문자) Within Group(정렬문 Order By) “새로운_컬럼명”
이처럼 작성을 하게 된다면 이는 Group By로 되어 있어야만 함수를 적용을 할 수 있다. 가령 예를 들어 부서 별로 묶을 때 이런 경우만 가능하다고 생각하면 된다. 물론 여러 개의 컬럼으로 그룹을 하게 되는 방법도 가능하니 참조하도록 하자. 정렬문은 Order By로 쓰면 되니깐 참조하고 아래의 문장대로 참조하자.

```
select department_id, Listagg(last_name, ', ') Within Group(Order by last_name)
from employees
where salary<10000
Group by department_id
order by 1;
```

이처럼 작성하게 되면 봉급이 10000 달러 미만인 직원들의 성에 대해서 이름 별로 부서 별로 출력이 된다고 생각하면 된다.
