

데이터분석 기초 SQL 부트캠프

목차

8. GROUP BY 구문

HAVING

9. IF / CASE

10. JOIN

8.

GROUP BY구문

HAVING

HAVING 절

- **HAVING** 절은 SQL의 GROUP BY 절과 함께 사용되며 그룹화된 결과에 조건을 적용하는 데 사용
- **WHERE** 절은 개별 테이블에 대한 조건을 적용하는 반면, **HAVING** 절은 그룹화된 결과의 집계 값에 대한 조건을 적용

HAVING

- SQL Code 처리 순서

```
SELECT  
    컬럼  
FROM  
    테이블  
WHERE  
    조건  
GROUP BY  
    조건  
HAVING  
    조건  
ORDER BY  
    조건
```

- HAVING : 그룹화 조건 확인.
- 항상 GROUP BY 뒤에 위치하며 GROUP BY 이후 그룹화 된 테이블에 조건 적용

HAVING

- SQL Code

```
SELECT productline,  
count(productline)  
FROM products  
GROUP BY productline  
HAVING COUNT(productline)  
>20  
;
```

- 결과

productline	count(productline)
Classic Cars	38
Vintage Cars	24

- Products 테이블에서
productLine 별 정보의 수가 20
이상

HAVING

- SQL Code

```
SELECT productline,  
avg(buyprice)  
FROM products  
GROUP BY productline  
HAVING AVG(BUYPRICE) < 50  
;
```

- 결과

products (4r × 2c)	
productline	avg(buyprice)
Planes	49.629167
Ships	47.007778
Trains	43.923333
Vintage Cars	46.06625

- Products 테이블에서
productLine 평균 가격이 \$50
미만

HAVING

- SQL Code

```
SELECT productcode,
SUM(quantityordered)
FROM orderdetails
WHERE orderlinenumber = 1
group BY productcode
HAVING PRODUCTCODE LIKE
'S10%'
;
```

- 결과

productcode	SUM(quantityordered)
S10_1678	254
S10_1949	120
S10_2016	106
S10_4698	76
S10_4757	51
S10_4962	48

- orderdetails 테이블에서 orderlinenumber
가 1인 제품들 중에서 productcode가
S10으로 시작하는 주문 수량의 합

HAVING

- SQL Code

```
SELECT productline,  
MAX(msrp)  
FROM products  
GROUP BY productline  
HAVING productline =  
'planes'  
;
```

- 결과

products (1r × 2c)	
productline	MAX(msrp)
Planes	157.69

- Products 테이블에서
productLine가 plane인 최고
msrp(권장 판매 가격)

HAVING 실습

- 문제 1: 'orders' 테이블에서 연도별 주문 건수가 100건을 초과하는 연도를 조회하세요.
(orderDate와 orderNumber 컬럼 사용)

- 정답

orders (2r × 2c)	
OrderYear	TotalOrders
2,003	111
2,004	151

HAVING 실습

- 문제 2: 'orderdetails' 테이블에서 상품별 연도별 총 주문량이 500개 이상인 상품 코드를 조회하세요. (productCode와 quantityOrdered 컬럼 사용)

- 정답

productCode	TotalQuantity
S10_1678	1,057
S10_1949	961
S10_2016	999
S10_4698	985
S10_4757	1,030
S10_4962	932
S12_1099	933
S12_1108	1,019
S12_1666	972
S12_2823	1,028

HAVING 실습

- 문제 3: 'payments'
테이블에서 고객별 총
결제 금액이
\$150,000을 초과하는
고객 번호를
조회하세요.
(customerNumber와
amount 컬럼 사용)

- 정답

payments (6r × 2c)	
customerNumber	TotalAmount
114	180,585.07
124	584,188.24
141	715,738.98
148	156,251.03
151	177,913.95
323	154,622.08

HAVING 실습

- 문제 4: 'customers' 테이블에서 국가별 고객 수가 10명 이상인 국가를 조회하세요.
(country와 customerNumber 컬럼 사용)

- 정답

customers (3r × 2c)	
country	TotalCustomers
France	12
USA	36
Germany	13

9.

IF / CASE구문

HAVING

IF

- IF(condition, value_if_true, value_if_false)
- IF 문은 SQL의 GROUP BY 절과 함께 사용되며 그룹화된 결과에 조건을 적용하는 데 사용
- WHERE 절은 개별 테이블에 대한 조건을 적용하는 반면, HAVING 절은 그룹화된 결과의 집계 값에 대한 조건을 적용

IF

- SQL Code

```
SELECT checkNumber,  
amount, IF(amount > 50000,  
'Large', 'Small') AS  
orderSize  
FROM payments  
;
```

- 결과

payments (273r × 3c)		
checkNumber	amount	orderSize
HQ336336	6,066.78	Small
JM555205	14,571.44	Small
OM314933	1,676.14	Small
B0864823	14,191.12	Small
HQ55022	32,641.98	Small
ND748579	33,347.88	Small
GG31455	45,864.03	Small
MA765515	82,261.22	Large

- payments 테이블에서 amount가 50000 초과인 경우 large, 이하인 경우 small 로 출력

IF 실습

- 문제 5. products' 테이블을 사용하여, 상품별로 가격이 \$100을 초과하면 'Expensive'로, 그렇지 않으면 'Cheap'으로 표시하는 쿼리를 작성하세요.
- (products.productName, products.buyPrice 컬럼 사용)

- 정답

products (110r × 2c)	
productName	PriceCategory
1969 Harley Davidson Ultimate Chopper	Cheap
1952 Alpine Renault 1300	Cheap
1996 Moto Guzzi 1100i	Cheap
2003 Harley-Davidson Eagle Drag Bike	Cheap
1972 Alfa Romeo GTA	Cheap
1962 LanciaA Delta 16V	Expensive
1968 Ford Mustang	Cheap
2001 Ferrari Enzo	Cheap

CASE

- CASE 문은 여러 조건을 테스트하고 여러 결과 중 하나를 반환

```
CASE expression
WHEN value1 THEN result1
WHEN value2 THEN result2
...
ELSE result
END as ~
```

CASE

- SQL Code

```
SELECT productName, buyPrice,
CASE
WHEN buyPrice < 20 THEN 'Cheap'
WHEN buyPrice BETWEEN 20 AND 50
THEN 'Moderate'
ELSE 'Expensive'
END AS priceCategory
FROM products;
```

- 결과

productName	buyPrice	priceCategory
1969 Harley Davidson Ultimate Chopper	48.81	Moderate
1952 Alpine Renault 1300	98.58	Expensive
1996 Moto Guzzi 1100i	68.99	Expensive
2003 Harley-Davidson Eagle Drag Bike	91.02	Expensive
1972 Alfa Romeo GTA	85.68	Expensive
1962 LanciaA Delta 16V	103.42	Expensive
1968 Ford Mustang	95.34	Expensive
2001 Ferrari Enzo	95.59	Expensive

- Buyprice 의 분류에 따라 cheap, moderate, expensive

CASE 실습

- 문제 6. employees' 테이블을 사용하여, 각 직원의 직책(jobTitle)에 따라 다음과 같이 분류하세요:
- 'Sales Rep': 'Sales Team'
- 'VP Sales': 'Management'
- 'VP Marketing': 'Management'
- 그 외: 'Other Positions'

정답

```
SELECT firstName, lastName, jobTitle,  
CASE jobTitle  
WHEN 'Sales Rep' THEN 'Sales Team'  
WHEN 'VP Sales' THEN 'Management'  
WHEN 'VP Marketing' THEN 'Management'  
ELSE 'Other Positions'  
END AS PositionCategory  
FROM employees;
```

CASE 실습

- 문제 6. employees' 테이블을 사용하여, 각 직원의 직책(jobTitle)에 따라 다음과 같이 분류하세요:
- 'Sales Rep': 'Sales Team'
- 'VP Sales': 'Management'
- 'VP Marketing': 'Management'
- 'Sales Manager (APAC)': 'Other Positions'
- 'Sale Manager (EMEA)': 'Other Positions'
- 'Sales Manager (NA)': 'Other Positions'
- 'Sales Rep': 'Sales Team'
- 'Sales Rep': 'Sales Team'
- 'Sales Rep': 'Sales Team'

정답

employees (23r x 4c)			
firstName	lastName	jobTitle	PositionCategory
Diane	Murphy	President	Other Positions
Mary	Patterson	VP Sales	Management
Jeff	Firrelli	VP Marketing	Management
William	Patterson	Sales Manager (APAC)	Other Positions
Gerard	Bondur	Sale Manager (EMEA)	Other Positions
Anthony	Bow	Sales Manager (NA)	Other Positions
Leslie	Jennings	Sales Rep	Sales Team
Leslie	Thompson	Sales Rep	Sales Team
Julie	Firrelli	Sales Rep	Sales Team

10.

JOIN구문

INNER JOIN, LEFT JOIN, RIGHT JOIN, UNION

JOIN

- JOIN 연산자는 두 테이블 간의 관계를 나타내기 위해 사용.
- 문법 :
- `SELECT * FROM TABLE A JOIN B ON 조건 ~`
- 예제) EX3 테이블과 EX4 테이블을 활용

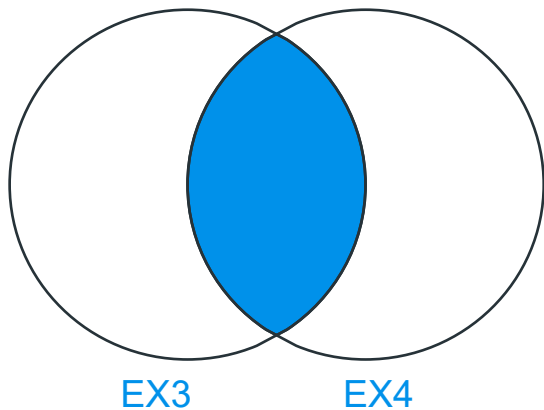
test.ex3: 3 행 (총) (대략적)

id	NAME	age
1	이상훈	34
2	박상훈	30
3	최상훈	20

test.ex4: 3 행 (총) (

id	region
1	서울
4	대구
5	부산

INNER JOIN



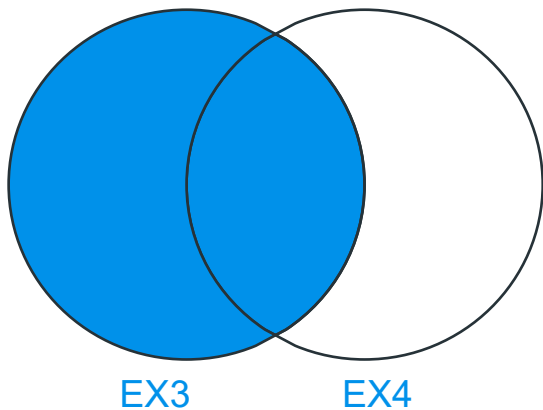
- SQL CODE

```
SELECT *  
FROM ex3  
JOIN ex4 ON ex3.id = ex4.id;
```

- 결과

결과 #1 (1r × 5c)				
id	NAME	age	id	region
1	이상훈	34	1	서울

LEFT JOIN(1/2)



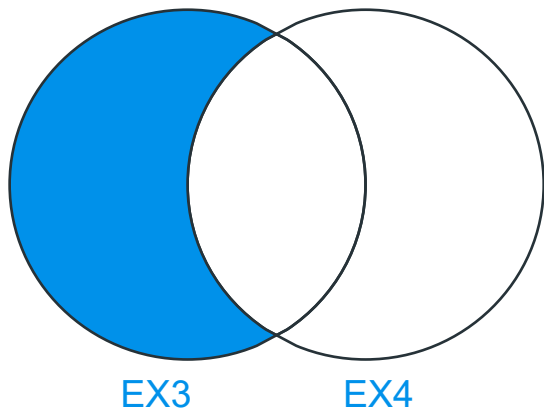
- SQL CODE

```
SELECT *
FROM ex3
LEFT JOIN ex4 ON ex3.id = ex4.id;
```

- 결과

결과 #1 (3r × 5c)				
id	NAME	age	id	region
1	이상훈	34	1	서울
2	박상훈	30	(NULL)	(NULL)
3	최상훈	20	(NULL)	(NULL)

LEFT JOIN(2/2)



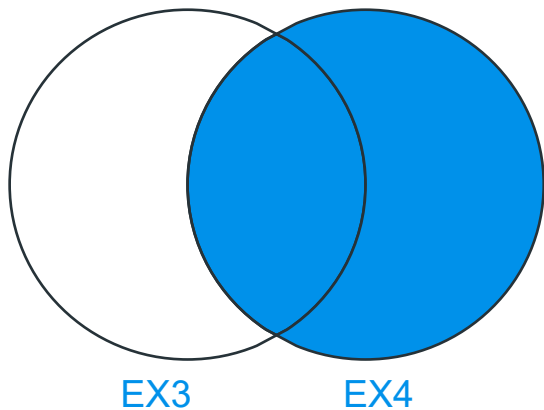
- SQL CODE

```
SELECT *
FROM ex3
left JOIN ex4 ON ex3.id = ex4.id
WHERE ex4.id IS null;
```

- 결과

결과 #1 (2r × 5c)				
id	NAME	age	id	region
2	박상훈	30	(NULL)	(NULL)
3	최상훈	20	(NULL)	(NULL)

RIGHT JOIN(1/2)



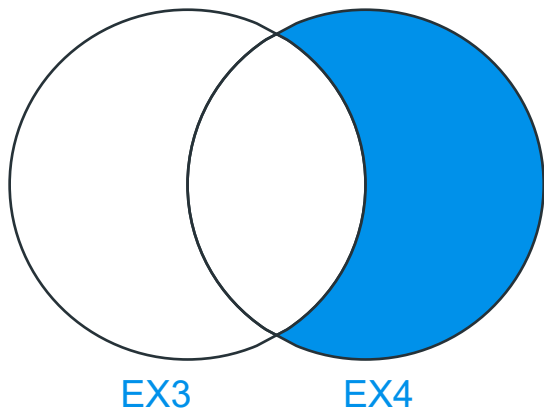
- SQL CODE

```
SELECT *
FROM ex3
RIGHT JOIN ex4 ON ex3.id = ex4.id;
```

- 결과

결과 #1 (3r × 5c)				
id	NAME	age	id	region
1	이상훈	34	1	서울
(NULL)	(NULL)	(NULL)	4	대구
(NULL)	(NULL)	(NULL)	5	부산

RIGHT JOIN(2/2)



- SQL CODE

```
SELECT *
FROM ex3
RIGHT JOIN ex4 ON ex3.id = ex4.id
WHERE ex3.id IS NULL;
```

- 결과

/ 결과 #1 (2r x 5c) \

id	NAME	age	id	region
(NULL)	(NULL)	(NULL)	4	대구
(NULL)	(NULL)	(NULL)	5	부산

UNION

- UNION 은 두 테이블의 데이터를 세로로 쭉 나열하는 역할.
- COLUMN의 수가 같아야 하며 중복은 제거.

- SQL CODE

```
SELECT id FROM ex3  
UNION  
SELECT id FROM ex4;
```

- 결과

id
1
2
3
4
5

UNION ALL

- UNION 은 중복을 그대로 표시

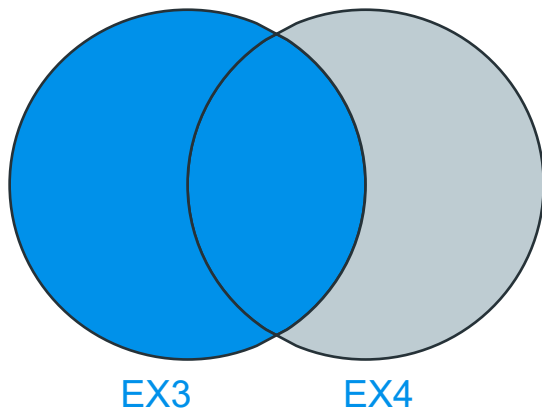
- SQL CODE

```
SELECT id FROM ex3  
UNION ALL  
SELECT id FROM ex4;
```

- 결과

id
1
2
3
1
4
5

FULL OUTER JOIN



- SQL CODE

```
SELECT ex3.id, ex3.name, ex3.age, ex4.id, ex4.region
FROM ex3
left JOIN ex4 ON ex3.id = ex4.id
union
SELECT ex3.id, ex3.name, ex3.age, ex4.id, ex4.region
FROM ex3
RIGHT JOIN ex4 ON ex3.id = ex4.id
WHERE ex3.id IS NULL;
```

- 결과

ex3 (5r × 5c)				
id	name	age	id	region
1	이상훈	34	1	서울
2	박상훈	30	(NULL)	(NULL)
3	최상훈	20	(NULL)	(NULL)
(NULL)	(NULL)	(NULL)	4	대구
(NULL)	(NULL)	(NULL)	가 5	부산

JOIN 실습

- 문제 7. 'customers' 테이블과 'orders' 테이블을 사용하여, 모든 고객의 이름과 주문 번호를 조회하세요.

정답

결과 #1 (326r × 2c)	
customerName	orderNumber 🔑
Atelier graphique	10,123
Atelier graphique	10,298
Atelier graphique	10,345
Signal Gift Stores	10,124
Signal Gift Stores	10,278
Signal Gift Stores	10,346
Australian Collectors, Co.	10,120
Australian Collectors, Co.	10,125
Australian Collectors, Co.	10,223
Australian Collectors, Co.	10,342
Australian Collectors, Co.	10,347

JOIN 실습

- 문제 8. 'products' 테이블과 'orderdetails' 테이블을 사용하여, 상품 이름과 주문된 수량을 조회하세요.

정답

결과 #1 (2,996r × 2c)	
productName	quantityOrdered
1969 Harley Davidson Ultimate Chopper	30
1969 Harley Davidson Ultimate Chopper	34
1969 Harley Davidson Ultimate Chopper	41
1969 Harley Davidson Ultimate Chopper	45
1969 Harley Davidson Ultimate Chopper	49
1969 Harley Davidson Ultimate Chopper	36
1969 Harley Davidson Ultimate Chopper	29
1969 Harley Davidson Ultimate Chopper	48

JOIN 실습

- 문제 9. 'Leslie'이라는 이름을 가진 직원이 담당하는 모든 고객의 이름을 조회하세요.
- 힌트. 'employees' 테이블과 'customers' 테이블을 사용

정답

customers (12r × 1c)	
customerName	
Mini Gifts Distributors Ltd.	
Mini Wheels Co.	
Technics Stores Inc.	
Corporate Gift Ideas Co.	
The Sharp Gifts Warehouse	
Signal Collectibles Ltd.	
Signal Gift Stores	
Toys4GrownUps.com	
Boards & Toys Co.	
Collectable Mini Designs Co.	
Men 'R' US Retailers, Ltd.	
West Coast Collectables Co.	

JOIN 실습

- 문제 10.
- San Francisco 사무실에서 근무하는 모든 직원의 이름을 조회하세요.
- 힌트. 'employees' 테이블과 'offices' 테이블을 사용

- 정답

employees (6r × 2c)	
firstName	lastName
Diane	Murphy
Mary	Patterson
Jeff	Firrelli
Anthony	Bow
Leslie	Jennings
Leslie	Thompson

JOIN 실습

- 문제 11. 주문 가격이 상품의 구매 가격보다 2.5배 높은 상품의 이름, 코드, 판매가격, 주문가격, 주문 수량을 조회하세요.
- 힌트. 'orderdetails' 테이블과 'products' 테이블을 사용


정답

결과 #1 (3r × 5c)				
productcode	productName	priceEach	buyPrice	quantityOrdered
524_4620	1961 Chevrolet Impala	80.84	32.33	23
524_4620	1961 Chevrolet Impala	80.84	32.33	22
524_4620	1961 Chevrolet Impala	80.84	32.33	41

JOIN 실습

- 문제 12. 2003년에 주문한 고객의 이름과 주문 번호를 조회하세요.
- 힌트. 'customers' 테이블과 'orders' 테이블을 JOIN

- 정답

결과 #1 (111r x 2c)	
customerName	orderNumber 
Atelier graphique	10,123
Signal Gift Stores	10,124
Australian Collectors, Co.	10,120
Australian Collectors, Co.	10,125
Baane Mini Imports	10,103
Baane Mini Imports	10,158
Mini Gifts Distributors Ltd.	10,113
Mini Gifts Distributors Ltd.	10,135
Mini Gifts Distributors Ltd.	10,142

JOIN 실습

- 문제 13. 2004년에
결제한 고객의 이름과
결제 금액을
조회하세요.
- 힌트. 'customers'
테이블과 'payments'
테이블을 JOIN

정답

결과 #1 (136r x 2c)	
customerName	amount
Atelier graphique	6,066.78
Atelier graphique	1,676.14
Signal Gift Stores	14,191.12
Signal Gift Stores	33,347.88
Australian Collectors, Co.	82,261.22
Australian Collectors, Co.	44,894.74
La Rochelle Gifts	19,501.82
La Rochelle Gifts	47,924.19
Baane Mini Imports	17,876.32

JOIN 실습

- 문제 14. 각 직원별로 담당한 고객의 수를 조회하세요.
- 힌트. 'employees' 테이블과 'customers' 테이블을 JOIN

정답

EmployeeName	NumberOfCustomers
Leslie Jennings	6
Leslie Thompson	6
Julie Firrelli	6
Steve Patterson	6
Foon Yue Tseng	7
George Vanauf	8
Loui Bondur	6
Gerard Hernandez	7
Pamela Castillo	10
Larry Bott	8
Barry Jones	9
Andy Fixter	5
Peter Marsh	5
Mami Nishi	5
Martin Gerard	6

JOIN 실습

- 문제 15. 2003년에 주문된 상품 이름과 해당 주문의 수량을 조회하세요.
- 힌트. 'orders' 테이블, 'orderdetails' 테이블, 'products' 테이블을 JOIN

- 정답

결과 #1 (1,052r × 2c)	
productName	quantityOrdered
1917 Grand Touring Sedan	30
1911 Ford Town Car	50
1932 Alfa Romeo 8C2300 Spider Sport	22
1936 Mercedes Benz 500k Roadster	49
1932 Model A Ford J-Coupe	25
1928 Mercedes-Benz SSK	26
1939 Chevrolet Deluxe Coupe	45
1938 Cadillac V-16 Presidential Limousine	46
1937 Lincoln Berline	39

JOIN 실습

- 문제 16. 각 고객별 총 주문 금액을 조회하세요.

- 정답

customerName	TotalOrderValue
Atelier graphique	22,314.36
Signal Gift Stores	80,180.98
Australian Collectors, Co.	180,585.07
La Rochelle Gifts	158,573.12
Baane Mini Imports	104,224.79
Mini Gifts Distributors Ltd.	591,827.34
Blauer See Auto, Co.	75,937.76
Mini Wheels Co.	66,710.56
Land of Toys Inc.	149,085.15
Euro+ Shopping Channel	820,689.54

- 힌트. 'customers' 테이블, 'orders' 테이블, 'orderdetails' 테이블을 JOIN

JOIN 실습

- 문제 17. 각 직원별로
담당한 고객들의 총
결제 금액을
조회하세요.
- 힌트. 'employees'
테이블, 'customers'
테이블, 'payments'
테이블을 JOIN

정답

EmployeeName	TotalPayments
Leslie Jennings	989,906.55
Leslie Thompson	347,533.03
Julie Firrelli	386,663.2
Steve Patterson	449,219.13
Foon Yue Tseng	488,212.67
George Vanauf	584,406.8
Loui Bondur	569,485.75
Gerard Hernandez	1,112,003.81
Pamela Castillo	750,201.87
Larry Bott	686,653.25

JOIN 실습

- 문제 18. 각 상품 라인별로 주문된 상품의 총 수량을 조회하세요.
- 힌트. 'products' 테이블, 'orderdetails' 테이블, 'productlines' 테이블을 JOIN

정답

productlines (7r × 2c)	
productLine	TotalQuantity
Classic Cars	35,582
Motorcycles	12,778
Planes	11,872
Ships	8,532
Trains	2,818
Trucks and Buses	11,001
Vintage Cars	22,933

JOIN 실습

- 문제 19. 2004년에 가장 많이 판매된 상위 5개 상품 이름과 해당 상품의 총 판매 수량을 조회하세요.
- 힌트. 'orders' 테이블, 'orderdetails' 테이블, 'products' 테이블을 JOIN

- 정답

products (5r × 2c)	
productName	TotalQuantity
1992 Ferrari 360 Spider red	789
1980s Black Hawk Helicopter	567
2001 Ferrari Enzo	566
The USS Constitution Ship	541
1941 Chevrolet Special Deluxe Cabriolet	536

11.

WINDOW 함수

SUM, AVG, MIN, MAX, COUNT
ROW_NUMBER, RANK, DENSE_RANK, LAG, LEAD,
FIRST_VALUE, LAST_VALUE
OVER (PARTITION BY ~ ORDER BY ~)

WINDOW 함수

- **SELECT** 구문에서 사용되며 분석 구간을 변동시키는 역할
- EX)누적합
- **SUM(COLUMN1) OVER(PARTITION BY COLUMN2
ORDER BY COLUMN3) AS NEW_COLUMN**

누적합

- `SUM(COLUMN1) OVER(PARTITION BY COLUMN2 ORDER BY COLUMN3) AS NEW_COLUMN`
- `PARTITION BY` : `GROUP BY` 와 비슷한 역할로, 그룹별로 누적 합계를 구할 수 있다. `GROUP BY` 는 집계 결과로 조회가 되는 반면 `PARTITION BY` 는 본래의 TABLE 그대로 출력.
- 생략 가능

누적합

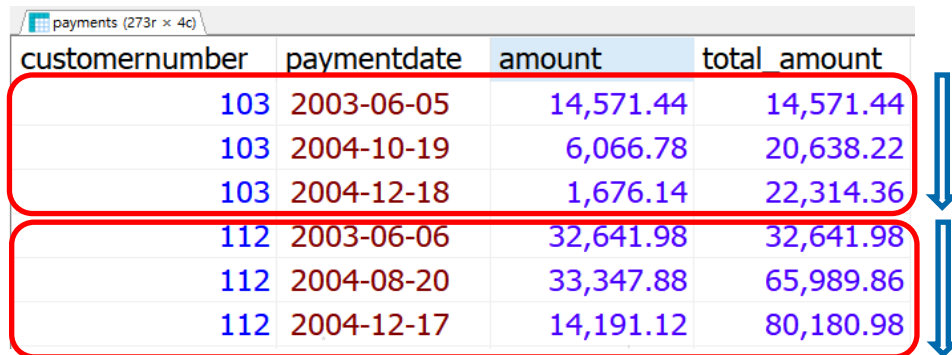
- `SUM(COLUMN1) OVER(PARTITION BY COLUMN2
ORDER BY COLUMN3) AS NEW_COLUMN`
- `ORDER BY` : 계산을 하는 순서를 정해준다.
- 생략 하면 누적 합이 계산되지 않는다.

누적합(over partition by order by 사용)

- SQL CODE

```
SELECT customernumber, paymentdate, amount,  
sum(amount) OVER(PARTITION BY CUSTOMERNUMBER  
ORDER BY PAYMENTDATE) AS total_amount  
FROM payments;
```

- 결과



customernumber	paymentdate	amount	total amount
103	2003-06-05	14,571.44	14,571.44
103	2004-10-19	6,066.78	20,638.22
103	2004-12-18	1,676.14	22,314.36
112	2003-06-06	32,641.98	32,641.98
112	2004-08-20	33,347.88	65,989.86
112	2004-12-17	14,191.12	80,180.98

누적합(over partition by 사용)

- SQL CODE

```
SELECT customernumber, paymentdate, amount,  
sum(amount) OVER( PARTITION BY  
CUSTOMERNUMBER) AS total_amount  
FROM payments;
```

- 결과(order by 생략으로 누적 합계 순서 x)

payments (273r x 4c)

customernumber	paymentdate	amount	total amount
103	2004-10-19	6,066.78	22,314.36
103	2003-06-05	14,571.44	22,314.36
103	2004-12-18	1,676.14	22,314.36
112	2004-12-17	14,191.12	80,180.98
112	2003-06-06	32,641.98	80,180.98
112	2004-08-20	33,347.88	80,180.98

누적합(over order by 사용)

- SQL CODE

```
SELECT customernumber, paymentdate, amount,
sum(amount) OVER(ORDER BY PAYMENTDATE) AS
total_amount
FROM payments;
```

- 결과(partition by 생략으로 그룹화 사라짐)

payments (273r x 4c)

customernumber	paymentdate	amount	total_amount
363	2003-01-16	10,223.83	10,223.83
128	2003-01-28	10,549.01	20,772.84
181	2003-01-30	5,494.78	26,267.62
121	2003-02-16	50,218.95	76,486.57
145	2003-02-20	53,959.21	130,445.78
141	2003-02-25	40,206.2	170,651.98
278	2003-03-02	52,151.81	222,803.79

누적합(over partition by 생략 order by 생략)

- SQL CODE

```
SELECT customernumber, paymentdate, amount,
sum(amount) OVER( ) AS total_amount
FROM payments;
```

- 결과(partition by, order by 생략으로 전체 sum 출력)

customernumber	paymentdate	amount	total_amount
103	2004-10-19	6,066.78	8,853,839.23
103	2003-06-05	14,571.44	8,853,839.23
103	2004-12-18	1,676.14	8,853,839.23
112	2004-12-17	14,191.12	8,853,839.23
112	2003-06-06	32,641.98	8,853,839.23
112	2004-08-20	33,347.88	8,853,839.23

over(partition by ~ order by ~) 실습

- 문제 20. 문제:
"orderdetails"
테이블에서 각
주문별로 주문된
제품의 평균
수량("quantityOrdered")을 계산하세요..

정답

orderNumber	productCode	avg_quantity_per_order
10,100	S18_1749	37.75
10,100	S18_2248	37.75
10,100	S18_4409	37.75
10,100	S24_3969	37.75
10,101	S18_2325	35.5
10,101	S18_2795	35.5
10,101	S24_1937	35.5
10,101	S24_2022	35.5

over(partition by ~ order by ~) 실습

- 문제21: "orders"
테이블에서 각
고객별로 주문 날짜에
따라서 지금까지의
주문 횟수를
계산하세요.

정답

customerNumber	orderNumber	orderDate	order_count_so_far
103	10,123	2003-05-20	1
103	10,298	2004-09-27	2
103	10,345	2004-11-25	3
112	10,124	2003-05-21	1
112	10,278	2004-08-06	2
112	10,346	2004-11-29	3
114	10,120	2003-04-29	1
114	10,125	2003-05-21	2
114	10,223	2004-02-20	3
114	10,342	2004-11-24	4
114	10,347	2004-11-29	5

WINDOW 함수

- LEAD / LAG 함수
- LEAD : 다음 행 데이터를 가져온다.
- LAG : 이전 행 데이터를 가져온다.

LEAD/LAG(over partition by order by 사용)

- SQL CODE

```
SELECT orderNumber, customerNumber, orderDate,
LAG(orderDate) OVER (PARTITION BY customerNumber ORDER BY orderDate)
AS prev_order_date,
LEAD(orderDate) OVER (PARTITION BY customerNumber ORDER BY
orderDate) AS next_order_date
FROM orders;
```

- 결과

orders (326r × 5c)

orderNumber	customerNumber	orderDate	prev_order_date	next_order_date
10,123	103	2003-05-20	(NULL)	2004-09-27
10,298	103	2004-09-27	2003-05-20	2004-11-25
10,345	103	2004-11-25	2004-09-27	(NULL)
10,124	112	2003-05-21	(NULL)	2004-08-06
10,278	112	2004-08-06	2003-05-21	2004-11-29
10,346	112	2004-11-29	2004-08-06	(NULL)

LEAD/LAG(over order by 사용)

- SQL CODE

```
SELECT orderNumber, customerNumber, orderDate,
LAG(orderDate) OVER ( ORDER BY orderDate) AS prev_order_date,
LEAD(orderDate) OVER ( ORDER BY orderDate) AS next_order_date
FROM orders;
```

- 결과

orders (326r × 5c)				
orderNumber	customerNumber	orderDate	prev_order_date	next_order_date
10,100	363	2003-01-06	(NULL)	2003-01-09
10,101	128	2003-01-09	2003-01-06	2003-01-10
10,102	181	2003-01-10	2003-01-09	2003-01-29
10,103	121	2003-01-29	2003-01-10	2003-01-31
10,104	141	2003-01-31	2003-01-29	2003-02-11
10,105	145	2003-02-11	2003-01-31	2003-02-17
10,106	278	2003-02-17	2003-02-11	2003-02-24

LEAD/LAG(over partition by 사용)

- SQL CODE

```
SELECT orderNumber, customerNumber, orderDate,
LAG(orderDate) OVER ( PARTITION BY customerNumber) AS
prev_order_date,
LEAD(orderDate) OVER ( PARTITION BY customerNumber) AS
next_order_date
FROM orders;
```

- 결과

orders (326r × 5c)

orderNumber	customerNumber	orderDate	prev_order_date	next_order_date
10,123	103	2003-05-20	(NULL)	2004-09-27
10,298	103	2004-09-27	2003-05-20	2004-11-25
10,345	103	2004-11-25	2004-09-27	(NULL)
10,124	112	2003-05-21	(NULL)	2004-08-06
10,278	112	2004-08-06	2003-05-21	2004-11-29
10,346	112	2004-11-29	2004-08-06	(NULL)

The table shows the results of the SQL query. Red boxes highlight the data for customer 103 and customer 112. Blue arrows indicate the LAG function mapping the current row's orderDate to the previous row's orderDate. Red arrows indicate the LEAD function mapping the current row's orderDate to the next row's orderDate.

LEAD/LAG(over order by 사용)

- SQL CODE

```
SELECT orderNumber, customerNumber, orderDate,
LAG(orderDate) OVER ( ) AS prev_order_date,
LEAD(orderDate) OVER ( ) AS next_order_date
FROM orders;
```

- 결과

orderNumber	customerNumber	orderDate	prev_order_date	next_order_date
10,100	363	2003-01-06	(NULL)	2003-01-09
10,101	128	2003-01-09	2003-01-06	2003-01-10
10,102	181	2003-01-10	2003-01-09	2003-01-29
10,103	121	2003-01-29	2003-01-10	2003-01-31
10,104	141	2003-01-31	2003-01-29	2003-02-11
10,105	145	2003-02-11	2003-01-31	2003-02-17
10,106	278	2003-02-17	2003-02-11	2003-02-24

over(partition by ~ order by ~) 실습

- 문제 22 :
"orderdetails"
테이블에서 각 제품
코드별로 주문된
수량(ORDERNUMBE
R)을 기준으로
정렬했을 때, 주문
수량의 증분을
계산하시오.

정답

orderNumber	productCode	quantityOrdered	quantity_difference
10,107	S10_1678	30	(NULL)
10,121	S10_1678	34	4
10,134	S10_1678	41	7
10,145	S10_1678	45	4
10,159	S10_1678	49	4
10,168	S10_1678	36	-13
10,180	S10_1678	29	-7
10,188	S10_1678	48	19

순위 함수

- **ROW_NUMBER** : 중복 없이 고유한 순위 부여
- **RANK** : 중복값에 같은 순위 부여, 중복된 숫자만큼 건너뛰기(1,1,1,4,5,6)
- **DENSE_RANK** : RANK와 유사하지만 중복된 숫자를 건너뛰지 않음(1,1,1,2,3,4)

순위 함수

- SQL CODE

```
SELECT customername, creditlimit,  
ROW_NUMBER() OVER ( ORDER BY creditlimit ASC) AS row_number_,  
RANK() OVER ( ORDER BY creditlimit ASC) AS rank_,  
DENSE_RANK() OVER ( ORDER BY creditlimit ASC) AS dense_rank_  
FROM customers  
ORDER BY creditlimit ASC;
```

순위 함수

결과

customername	creditlimit	row_number_	rank_	dense_rank_
Euro+ Shopping Channel	227,600.0	1	1	1
Mini Gifts Distributors Ltd.	210,500.0	2	2	2
Vida Sport, Ltd	141,300.0	3	3	3
Muscle Machine Inc	138,500.0	4	4	4
AV Stores, Co.	136,800.0	5	5	5
Saveley & Henriot, Co.	123,900.0	6	6	6
Marta's Replicas Co.	123,700.0	7	7	7
L'ordine Souvenirs	121,400.0	8	8	8
Heintze Collectables	120,800.0	9	9	9
Toms Spezialitäten, Ltd	120,400.0	10	10	10
Rovelli Gifts	119,600.0	11	11	11
La Rochelle Gifts	118,200.0	12	12	12
Australian Collectors, Co.	117,300.0	13	13	13
Scandinavian Gift Ideas	116,400.0	14	14	14
Land of Toys Inc.	114,900.0	15	15	15
Online Diecast Creations Co.	114,200.0	16	16	16
Amica Models & Co.	113,000.0	17	17	17
Kelly's Gift Shop	110,000.0	18	18	18
Anna's Decorations, Ltd	107,800.0	19	19	19
Collectable Mini Designs Co.	105,000.0	20	20	20
Corporate Gift Ideas Co.	105,000.0	21	20	20
Corrida Auto Replicas, Ltd	104,600.0	22	22	21

순위 함수

- **First_value():** 가장 첫번째 오는 row 조회
- **Last_value():** 가장 마지막에 오는 row 조회
- 위 두 함수는 order by의 활용에 따라 결과가 달라짐
- Ex) first_value(column) over (partition by ~ order by ~)

over(partition by ~ order by ~) 실습

- 문제 23 : "products"
테이블에서 각 제품
라인별로 가장 비싼
제품의 이름과 가장
싼 제품의 이름을
조회하세요.

정답

products (110r x 5q)				
productLine	productName	buyprice	cheapest_product	most_expensive_product
Classic Cars	1962 LanciaA Delta 16V	103.42	1958 Chevy Corvette Limited...	1962 LanciaA Delta 16V
Classic Cars	1998 Chrysler Plymouth Pro...	101.51	1958 Chevy Corvette Limited...	1962 LanciaA Delta 16V
Classic Cars	1952 Alpine Renault 1300	98.58	1958 Chevy Corvette Limited...	1962 LanciaA Delta 16V

WINDOW FRAME(윈도우 프레임)

- **ROW:** 행의 개수로 윈도우 프레임을 정의
- **RANGE:** 정렬의 기준이 되는 행의 값을 기준으로 정의
- **PRECEDING:** 현재 행보다 전에 있는 행들을 의미
- **FOLLOWING:** 현재 행보다 다음에 있는 행들을 의미
- **UNBOUNDED PRECEDING:** 현재 파티션의 첫 번째 행부터 현재 행까지의 범위를 의미
- **UNBOUNDED FOLLOWING:** 현재 행부터 현재 파티션의 마지막 행까지의 범위를 의미
- **CURRENT ROW:** 현재 행

WINDOW FRAME(윈도우 프레임)

- SQL CODE

```
SELECT orderNumber, productCode, quantityOrdered,  
AVG(quantityOrdered) OVER (ORDER BY orderNumber ROWS BETWEEN 1  
PRECEDING AND 1 FOLLOWING) AS moving_avg_quantity_1,  
AVG(quantityOrdered) OVER (ORDER BY orderNumber ROWS BETWEEN  
CURRENT ROW AND 1 FOLLOWING) AS moving_avg_quantity_2,  
AVG(quantityOrdered) OVER (ORDER BY orderNumber ROWS BETWEEN 1  
PRECEDING AND CURRENT ROW) AS moving_avg_quantity_3,  
AVG(quantityOrdered) OVER (ORDER BY orderNumber RANGE BETWEEN 1  
PRECEDING AND 1 FOLLOWING) AS moving_avg_quantity_4  
FROM orderdetails  
;
```

WINDOW FRAME(윈도우 프레임)

- SQL CODE

orderNumber	productCode	quantityOrdered	moving_avg_quantity_1	moving_avg_quantity_2	moving_avg_quantity_3	moving_avg_quantity_4
10,100	S18_1749	30	40.0	40.0	30.0	36.625
10,100	S18_2248	50	34.0	36.0	40.0	36.625
10,100	S18_4409	22	40.3333	35.5	36.0	36.625
10,100	S24_3969	49	32.0	37.0	35.5	36.625
10,101	S18_2325	25	33.3333	25.5	37.0	37.3
10,101	S18_2795	26	32.0	35.5	25.5	37.3
10,101	S24_1937	45	39.0	45.5	35.5	37.3
10,101	S24_2022	46	43.3333	42.5	45.5	37.3
10,102	S18_1342	39	42.0	40.0	42.5	34.6818
10,102	S18_1367	41	35.3333	33.5	40.0	34.6818
10,103	S10_1949	26	36.3333	34.0	33.5	34.3226

over(partition by ~ order by ~) 실습

- 문제 24 : 직원별로 담당하는 고객 수를 계산하고, 각 직원별 담당 고객 수의 누적 합계를 계산하세요.

정답

employeeNumber	firstName	lastName	customerCount	cumulativeCustomerCount
1,165	Leslie	Jennings	6	6
1,166	Leslie	Thompson	6	12
1,188	Julie	Firrelli	6	18
1,216	Steve	Patterson	6	24
1,286	Foon Yue	Tseng	7	31
1,323	George	Vanauf	8	39
1,337	Loui	Bondur	6	45
1,370	Gerard	Hernandez	7	52
1,401	Pamela	Castillo	10	62
1,501	Larry	Bott	8	70
1,504	Barry	Jones	9	79
1,611	Andy	Fixter	5	84
1,612	Peter	Marsh	5	89
1,621	Mami	Nishi	5	94
1,702	Martin	Gerard	6	100