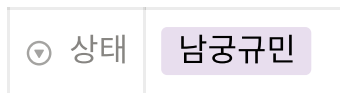
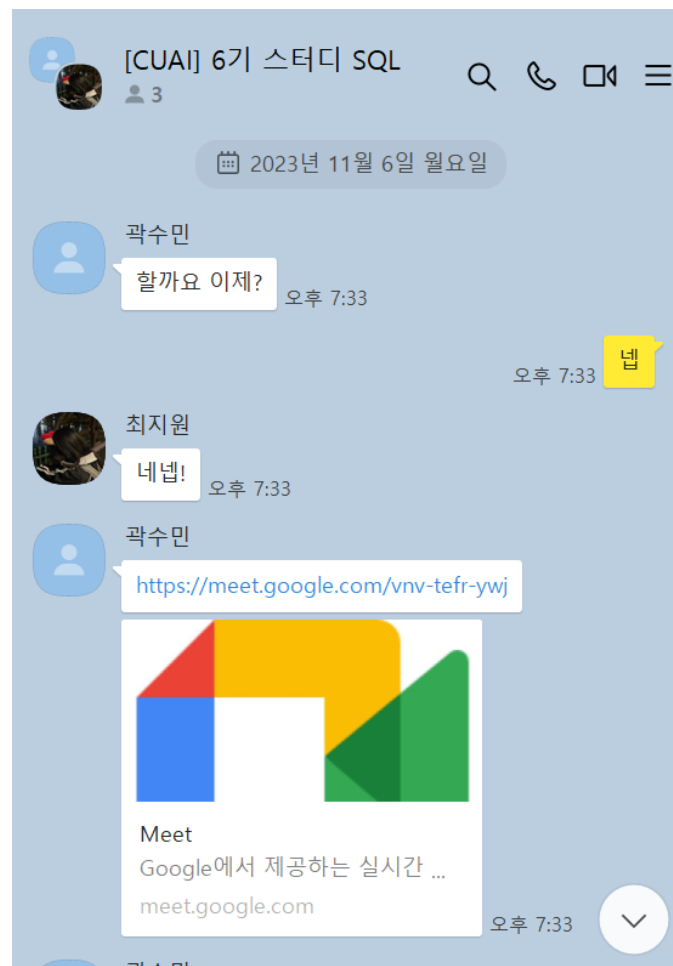


8~9강



스터디 인증 회의 사진 캡처를 까먹어서 카톡방으로 대체합니다...!!



▼ 8강. 여러 개의 테이블 조작하기

여러 개의 테이블을 세로로 결합하기

```
SELECT 'APP1' AS APP, USER_ID, NAME, EMAIL FROM app1_mst_users
UNION ALL
SELECT 'APP2' AS APP, USER_ID, NAME, NULL AS EMAIL FROM APP2_MST_USER
```

	app text	user_id character varying (255)	name character varying (255)	email character varying
1	APP1	U001	Sato	sato@example.com
2	APP1	U002	Suzuki	suzuki@example.com
3	APP2	U001	Ito	[null]
4	APP2	U002	Tanaka	[null]

UNION ALL; 테이블을 세로로 결합

- 열 이름이 동일해야 함
- EMAIL과 같이 한 테이블에만 있는 데이터는 다른 테이블에서 NULL로 선택

여러 개의 테이블을 가로로 정렬하기

```
-- 단순 JOIN
SELECT
  M.CATEGORY_ID, M.NAME
  , S.SALES
  , R.PRODUCT_ID AS SALE_PRODUCT
FROM
  MST_CATEGORIES AS M
JOIN
  CATEGORY_SALES AS S
  ON M.CATEGORY_ID=S.CATEGORY_ID
JOIN
  PRODUCT_SALE_RANKING AS R
  ON M.CATEGORY_ID=R.CATEGORY_ID;
```

	category_id integer	name character varying (255)	sales integer	sale_product character varying
1	1	dvd	850000	D001
2	1	dvd	850000	D002
3	1	dvd	850000	D003
4	2	cd	500000	C001
5	2	cd	500000	C002
6	2	cd	500000	C003

- 단순 JOIN → 결손 데이터(마스터 테이블의 ID 3번), 중복 데이터 발생(DVD,CD 이름,가격)

- 마스터 테이블의 행 수가 변경됨

```
-- LEFT JOIN
SELECT
  M.CATEGORY_ID, M.NAME
  , S.SALES
  , R.PRODUCT_ID AS SALE_PRODUCT
FROM
  MST_CATEGORIES AS M
  LEFT JOIN
  CATEGORY_SALES AS S
  ON M.CATEGORY_ID=S.CATEGORY_ID
  LEFT JOIN
  PRODUCT_SALE_RANKING AS R
  ON M.CATEGORY_ID=R.CATEGORY_ID
  AND R.RANK=1;
```

	category_id integer	name character varying (255)	sales integer	sale_product character varying (255)
1	1	dvd	850000	D001
2	2	cd	500000	C001
3	3	book	[null]	[null]

- LEFT JOIN으로 마스터 테이블의 결손값이 없도록 함
- 순위 테이블에서 순위가 1인 데이터들만 조인시켜 중복 데이터가 없도록 함

```
-- 서브쿼리 이용
SELECT
  M.CATEGORY_ID, M.NAME
  , (SELECT S.SALES
    FROM CATEGORY_SALES AS S
    WHERE M.CATEGORY_ID=S.CATEGORY_ID) AS SALES
  , (SELECT R.PRODUCT_ID
    FROM PRODUCT_SALE_RANKING AS R
    WHERE M.CATEGORY_ID=R.CATEGORY_ID
    ORDER BY SALES DESC
    LIMIT 1) AS TOP_SALE_PRODUCT
FROM
  MST_CATEGORIES AS M;
```

	category_id integer	name character varying (255)	sales integer	top_sale_product character varying (255)
1	1	dvd	850000	D001
2	2	cd	500000	C001
3	3	book	[null]	[null]

- SELECT 절에서 서브쿼리 이용
- WHERE로 ID가 같은 경우만 선택
- 가장 판매액이 높은 상품열은 WHERE 다음 정렬 후 1개만 선택

조건 플래그를 0과 1로 표현하기

```
SELECT
  M.USER_ID, M.CARD_NUMBER
  , COUNT(P.USER_ID) AS PURCHASE_COUNT
  , CASE WHEN M.CARD_NUMBER IS NOT NULL THEN 1 ELSE 0 END AS HAS_CARD
  , SIGN(COUNT(P.USER_ID)) AS HAS_PURCHASED
FROM
  MST_USERS_WITH_CARD_NUMBER AS M
  LEFT JOIN
  PURCHASE_LOG AS P
  ON M.USER_ID=P.USER_ID
GROUP BY M.USER_ID, M.CARD_NUMBER;
```

	user_id character varying (255)	card_number character varying (255)	purchase_count bigint	has_card integer	has_purchased double precision
1	U002	[null]	2	0	1
2	U003	5678-xxxx-xxxx-xxxx	0	1	0
3	U001	1234-xxxx-xxxx-xxxx	3	1	1

- 마스터 테이블에 구매 로그 LEFT JOIN
- 마스터 테이블의 USER_ID, CARD_NUMBER로 그룹핑하고
- COUNT로 구매 테이블 USER 개수 카운트 ⇒ 구매 이력
- 카드번호 유무 CASE WHEN, 구매이력여부 SIGN으로 계산

계산한 테이블에 이름 붙여 재사용하기

CTE: Common Table Expression 공통 테이블식⇒ 일시적 테이블에 이름 붙여 사용
→ 가독성 꺼

=WITH <테이블 이름> AS (SELECT~)

```
--WITH 구문으로 임시테이블 만들기
WITH
PRODUCT_SALE_RANKING AS (
  SELECT
    CATEGORY_NAME
    , PRODUCT_ID
    , SALES
    , ROW_NUMBER() OVER(PARTITION BY CATEGORY_NAME
                        ORDER BY SALES DESC) AS RANK
  FROM
    PRODUCT_SALES
)
SELECT * FROM PRODUCT_SALE_RANKING
```

	category_name character varying (255) 🔒	product_id character varying (255) 🔒	sales integer 🔒	rank bigint 🔒
1	book	B001	20000	1
2	book	B002	15000	2
3	book	B003	10000	3
4	book	B004	5000	4
5	cd	C001	30000	1
6	cd	C002	20000	2
7	cd	C003	10000	3
8	dvd	D001	50000	1
9	dvd	D002	20000	2
10	dvd	D003	10000	3


- 카테고리별 순위 매기기

```
--WITH 구문에 테이블 여러 개 넣기
WITH
PRODUCT_SALE_RANKING AS (
  SELECT
    CATEGORY_NAME
    , PRODUCT_ID
```

```

, SALES
, ROW_NUMBER() OVER(PARTITION BY CATEGORY_NAME
ORDER BY SALES DESC) AS RANK
FROM
PRODUCT_SALES
), MST_RANK AS (
SELECT DISTINCT RANK
FROM PRODUCT_SALE_RANKING)
SELECT * FROM MST_RANK;

```

	rank bigint 
1	4
2	2
3	3
4	1

- WITH 구문에 여러 개의 테이블을 넣을 때에는 쉼표로 구분
- PRODUCT_SALE_RANKING의 RANK에서 고유값만 (DISTINCT) 출력

```

WITH
PRODUCT_SALE_RANKING AS (
SELECT
CATEGORY_NAME
, PRODUCT_ID
, SALES
, ROW_NUMBER() OVER(PARTITION BY CATEGORY_NAME
ORDER BY SALES DESC) AS RANK
FROM
PRODUCT_SALES
), MST_RANK AS (
SELECT DISTINCT RANK
FROM PRODUCT_SALE_RANKING)
SELECT
M.RANK
, R1.PRODUCT_ID AS DVD
, R1.SALES AS DVD_SALES
, R2.PRODUCT_ID AS CD
, R2.SALES AS CD_SALES
, R3.PRODUCT_ID AS BOOK
, R3.SALES AS BOOK_SALES
FROM
MST_RANK AS M

```

```

LEFT JOIN
  PRODUCT_SALE_RANKING AS R1
  ON M.RANK=R1.RANK
  AND R1.CATEGORY_NAME='dvd'
LEFT JOIN
  PRODUCT_SALE_RANKING AS R2
  ON M.RANK=R2.RANK
  AND R2.CATEGORY_NAME='cd'
LEFT JOIN
  PRODUCT_SALE_RANKING AS R3
  ON M.RANK=R3.RANK
  AND R3.CATEGORY_NAME='book'
ORDER BY M.RANK;

```

	rank bigint	dvd character varying (255)	dvd_sales integer	cd character varying (255)	cd_sales integer	book character varying (255)	book_sales integer
1	1	D001	50000	C001	30000	B001	20000
2	2	D002	20000	C002	20000	B002	15000
3	3	D003	10000	C003	10000	B003	10000
4	4	[null]	[null]	[null]	[null]	B004	5000

!!문자 데이터는 대소문자 구분..... 🐱😭

- 고유 순위 테이블을 기준으로 카테고리 이름이 dvd,cd,book인 데이터를 각각 left join

유사 테이블 만들기

```

--SELECT 구문 UNION ALL로 합쳐서 만드는 방법
WITH
  MST_DEVICES AS (
    SELECT 1 AS DEVICE_ID, 'PC' AS DEVICE_NAME
    UNION ALL SELECT 2 AS DEVICE_ID, 'SP' AS DEVICE_NAME
    UNION ALL SELECT 3 AS DEVICE_ID, '어플리케이션' AS DEVICE_NAME)
SELECT *
FROM MST_DEVICES;

```

	device_id integer	device_name text
1	1	PC
2	2	SP
3	3	어플리케이션

- UNION ALL은 처리가 비교적 무거움

```
--MST_USERS와 조인해서 나타냄
WITH
MST_DEVICES AS (
    SELECT 1 AS DEVICE_ID, 'PC' AS DEVICE_NAME
    UNION ALL SELECT 2 AS DEVICE_ID, 'SP' AS DEVICE_NAME
    UNION ALL SELECT 3 AS DEVICE_ID, '어플리케이션' AS DEVICE_NAME)
SELECT
    U.USER_ID
    ,D.DEVICE_NAME
FROM
    MST_USERS AS U
    LEFT JOIN
        MST_DEVICES AS D
    ON U.REGISTER_DEVICE=D.DEVICE_ID ;
```

	user_id character varying (255) 🔒	device_name text 🔒
1	U001	PC
2	U002	SP
3	U003	어플리케이션

```
--VALUES 구문 이용해서 유사 테이블 만들기
WITH
MST_DEVICES(DEVICE_ID, DEVICE_NAME) AS (
VALUES
    (1, 'PC')
    ,(2, 'SP')
    ,(3, '애플리케이션')
)
SELECT * FROM MST_DEVICES;
```


	device_id integer	device_name text
1	1	PC
2	2	SP
3	3	애플리케이션

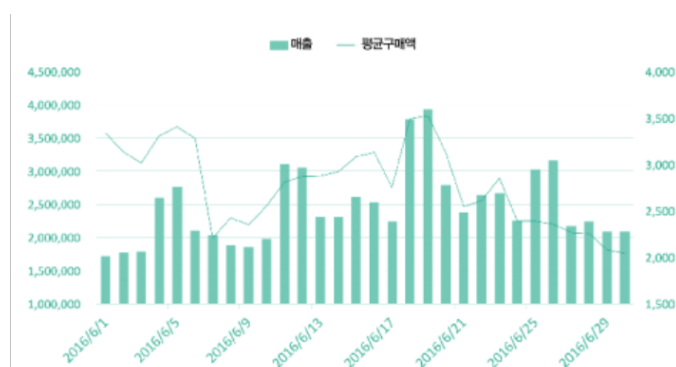
```
--순번 만들기
WITH
SERIES AS (
    SELECT GENERATE_SERIES(1,5) AS IDX)
SELECT * FROM SERIES;
```

	idx integer
1	1
2	2
3	3
4	4
5	5

- GENERATE_SERIES(1,5)

▼ 9강. 시계열 기반으로 데이터 집계하기

날짜별 매출 집계하기



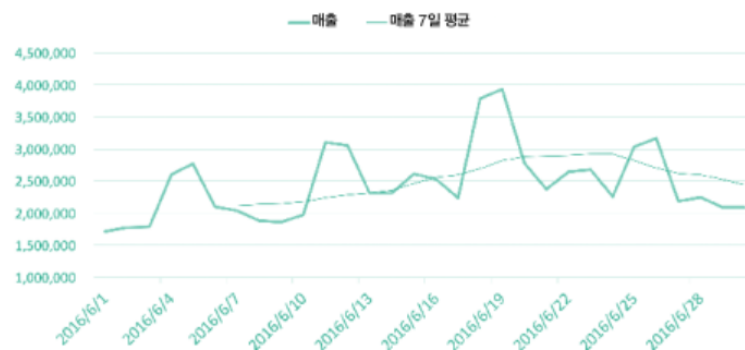
매출 집계시에는 날짜-금액 표현 그래프 사용

```
SELECT
  DT
  , COUNT(*) AS PURCHASE_COUNT
  , SUM(PURCHASE_AMOUNT) AS TOTAL_AMOUNT
  , AVG(PURCHASE_AMOUNT) AS AVG_AMOUNT
FROM PURCHASE_LOG
GROUP BY DT
ORDER BY DT
;
```

	dt character varying (255)	purchase_count bigint	total_amount bigint	avg_amount numeric
1	2014-01-01	2	24516	12258.000000
2	2014-01-02	2	36049	18024.50
3	2014-01-03	3	53029	17676.33
4	2014-01-04	3	29299	9766.333333
5	2014-01-05	3	48256	16085.33
6	2014-01-06	3	29440	9813.333333

- DT로 그룹화하여 카운트, 합계, 평균

이동평균을 사용한 날짜별 추이 보기



매출액만 확인하면, 주말과 같이 주기적으로 높아지는 날이 있어 전체적으로 매출 상승, 하락을 파악하기 어려움 ⇒ 7일 이동평균으로 표현하기

```
SELECT
  DT
```

```

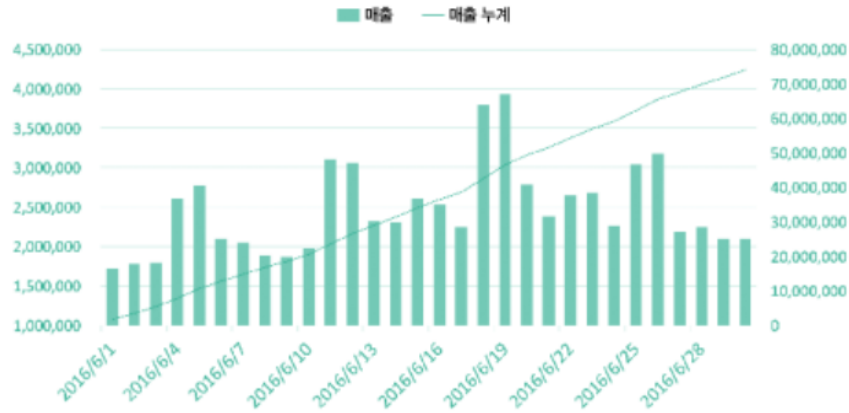
, SUM(PURCHASE_AMOUNT) AS TOTAL_AMOUNT
, AVG(SUM(PURCHASE_AMOUNT))
OVER(ORDER BY DT ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
AS SEVEN_DAY_AVG
-- 정확한 이동평균
, CASE
  WHEN
    7=COUNT(*)
    OVER(ORDER BY DT ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
  THEN
    AVG(SUM(PURCHASE_AMOUNT))
    OVER(ORDER BY DT ROWS BETWEEN 6 PRECEDING AND CURRENT ROW)
  END AS SEVEN_DAY_AVG_STRICT
FROM PURCHASE_LOG
GROUP BY DT
ORDER BY DT
;

```

	dt character varying (255)	total_amount bigint	seven_day_avg numeric	seven_day_avg_strict numeric
1	2014-01-01	24516	24516.000000000000	
2	2014-01-02	36049	30282.500000000000	
3	2014-01-03	53029	37864.666666666667	
4	2014-01-04	29299	35723.250000000000	
5	2014-01-05	48256	38229.800000000000	
6	2014-01-06	29440	36764.833333333333	
7	2014-01-07	47679	38324.000000000000	38324.00
8	2014-01-08	19760	37644.571428571429	37644.57

- 7일 이동평균 구하기
- DT로 그룹화 한다음, 정렬하고, 앞선 6개부터 현재 행까지의 평균을 구함
- 6일 이전의 데이터가 있어야 정확한 이동평균 계산이 가능

당월 매출 누계 구하기



매출액과 매출 누계 그래프

```
-- 날짜별 매출과 당원 누계 매출을 집계하는 쿼리
SELECT
  DT
  -- SUBSTR도 가능
  , SUBSTRING(DT, 1, 7) AS YEAR_MONTH
  , SUM(PURCHASE_AMOUNT) AS TOTAL_AMOUNT
  , SUM(SUM(PURCHASE_AMOUNT))
    OVER(PARTITION BY SUBSTRING(DT, 1, 7)
         ORDER BY DT ROWS UNBOUNDED PRECEDING)
    AS AGG_AMOUNT
FROM PURCHASE_LOG
GROUP BY DT
ORDER BY DT
;
```

	dt character varying (255)	year_month text	total_amount bigint	agg_amount numeric
1	2014-01-01	2014-01	24516	24516
2	2014-01-02	2014-01	36049	60565
3	2014-01-03	2014-01	53029	113594
4	2014-01-04	2014-01	29299	142893
5	2014-01-05	2014-01	48256	191149
6	2014-01-06	2014-01	29440	220589
7	2014-01-07	2014-01	47679	268268
8	2014-01-08	2014-01	19760	288028
9	2014-01-09	2014-01	22944	310972
10	2014-01-10	2014-01	27923	338895

- 월별 누계는 월별 그룹화(PARTITION BY)하고 이전까지의 행들을 모두 합함

<가독성 높이기>

```
-- 날짜별 매출을 임시 테이블로 만드는 쿼리
WITH
DAILY_PURCHASE AS (
  SELECT
    DT
    , SUBSTRING(DT,1,4) AS YEAR
    , SUBSTRING(DT,6,2) AS MONTH
    , SUBSTRING(DT,9,2) AS DATE
    , SUM(PURCHASE_AMOUNT) AS PURCHASE_AMOUNT
    , COUNT(ORDER_ID) AS ORDERS
  FROM PURCHASE_LOG
  GROUP BY DT
)
SELECT * FROM DAILY_PURCHASE
ORDER BY DT;
```

	dt character varying (255)	year text	month text	date text	purchase_amount bigint	orders bigint
1	2014-01-01	2014	01	01	24516	2
2	2014-01-02	2014	01	02	36049	2
3	2014-01-03	2014	01	03	53029	3
4	2014-01-04	2014	01	04	29299	3
5	2014-01-05	2014	01	05	48256	3
6	2014-01-06	2014	01	06	29440	3
7	2014-01-07	2014	01	07	47679	3
8	2014-01-08	2014	01	08	19760	3

```
-- DAILY_PURCHASE 테이블에 당월 누계 매출을 집계하는 쿼리
WITH
DAILY_PURCHASE AS (
  SELECT
    DT
    , SUBSTRING(DT,1,4) AS YEAR
    , SUBSTRING(DT,6,2) AS MONTH
    , SUBSTRING(DT,9,2) AS DATE
    , SUM(PURCHASE_AMOUNT) AS PURCHASE_AMOUNT
    , COUNT(ORDER_ID) AS ORDERS
  FROM PURCHASE_LOG
  GROUP BY DT
)
SELECT
  DT
  , CONCAT(YEAR, '-', MONTH) AS YEAR_MONTH
  , PURCHASE_AMOUNT
  , SUM(PURCHASE_AMOUNT)
```

```

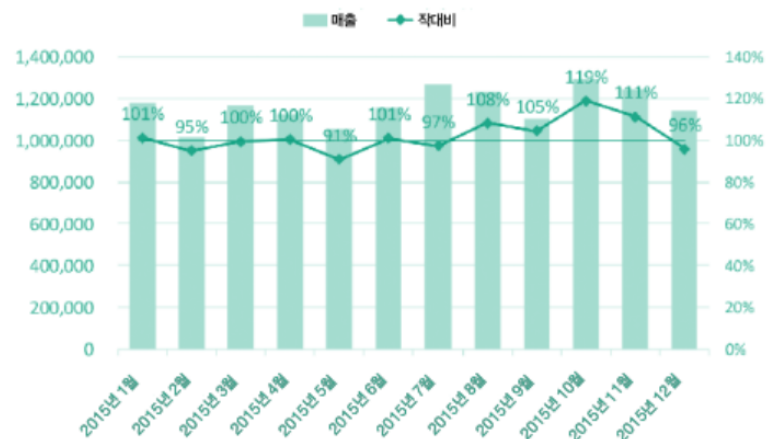
OVER(PARTITION BY YEAR, MONTH
      ORDER BY DT ROWS UNBOUNDED PRECEDING)
AS AGG_AMOUNT
FROM DAILY_PURCHASE
ORDER BY DT;

```

	dt character varying (255)	year_month text	purchase_amount bigint	agg_amount numeric
1	2014-01-01	2014-01	24516	24516
2	2014-01-02	2014-01	36049	60565
3	2014-01-03	2014-01	53029	113594
4	2014-01-04	2014-01	29299	142893
5	2014-01-05	2014-01	48256	191149
6	2014-01-06	2014-01	29440	220589
7	2014-01-07	2014-01	47679	268268
8	2014-01-08	2014-01	19760	288028

- SUBSTRING을 WITH 임시테이블에서 년, 월, 일을 미리 나누어 해당 정보를 사용

월별 매출의 작대비 구하기



매출액과 작대비(작년대비비율)을 동시에 나타냄

```

WITH
DAILY_PURCHASE AS (
  SELECT
    DT
    , SUBSTRING(DT, 1, 4) AS YEAR
    , SUBSTRING(DT, 6, 2) AS MONTH

```

```

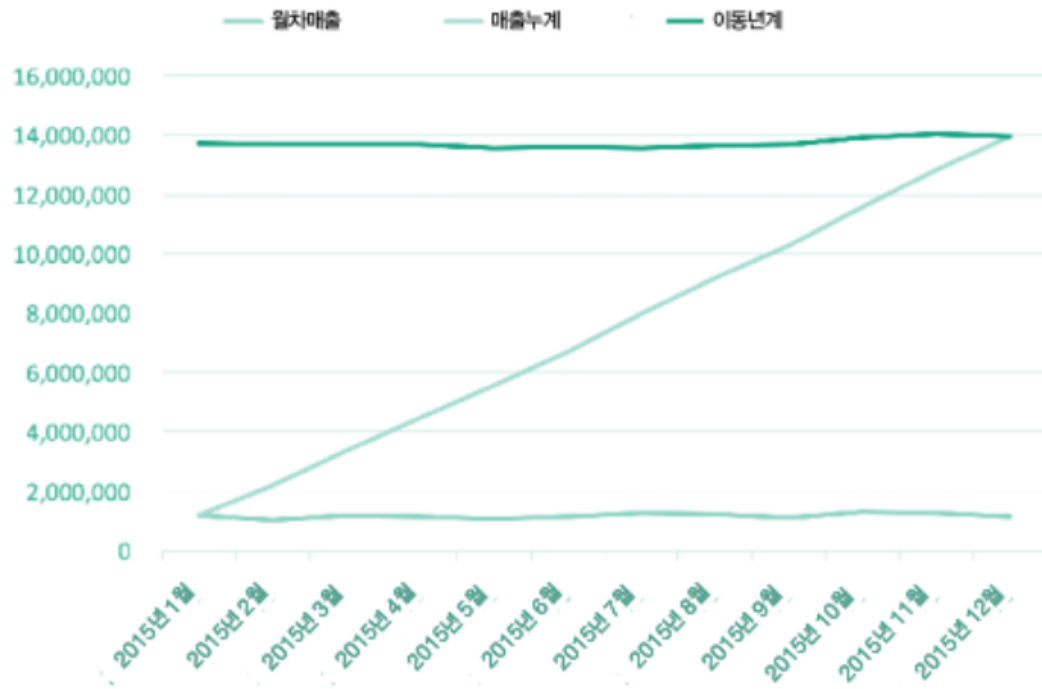
, SUBSTRING(DT, 9, 2) AS DATE
, SUM(PURCHASE_AMOUNT) AS PURCHASE_AMOUNT
, COUNT(ORDER_ID) AS ORDERS
FROM PURCHASE_LOG
GROUP BY DT
)
SELECT
    MONTH
    , SUM(CASE YEAR WHEN '2014' THEN PURCHASE_AMOUNT END) AS AMOUNT_2014
    , SUM(CASE YEAR WHEN '2015' THEN PURCHASE_AMOUNT END) AS AMOUNT_2015
    , 100.0
    * SUM(CASE YEAR WHEN '2015' THEN PURCHASE_AMOUNT END)
    / SUM(CASE YEAR WHEN '2014' THEN PURCHASE_AMOUNT END)
    AS RATE
FROM DAILY_PURCHASE
GROUP BY MONTH
ORDER BY MONTH;

```

	month text	amount_2014 numeric	amount_2015 numeric	rate numeric
1	01	13900	22111	159.0719424460431655
2	02	28469	11965	42.0281709930099406
3	03	18899	20215	106.9633313931954072
4	04	12394	11792	95.1428110375988381
5	05	2282	18087	792.5942156003505697
6	06	10180	18859	185.2554027504911591

- 2014년도 매출, 2015년도 매출, 2014대비 2015매출 비율
- CASE WHEN

Z 차트로 업적의 추이 확인하기



- 계절 변동의 영향을 배제하고 트렌드를 분석하는 방법.
- 월차매출 : 매출 합계를 월별로 집계

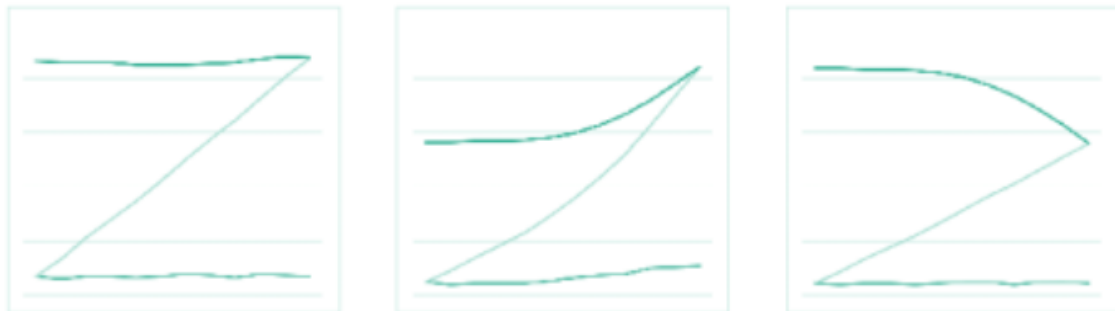
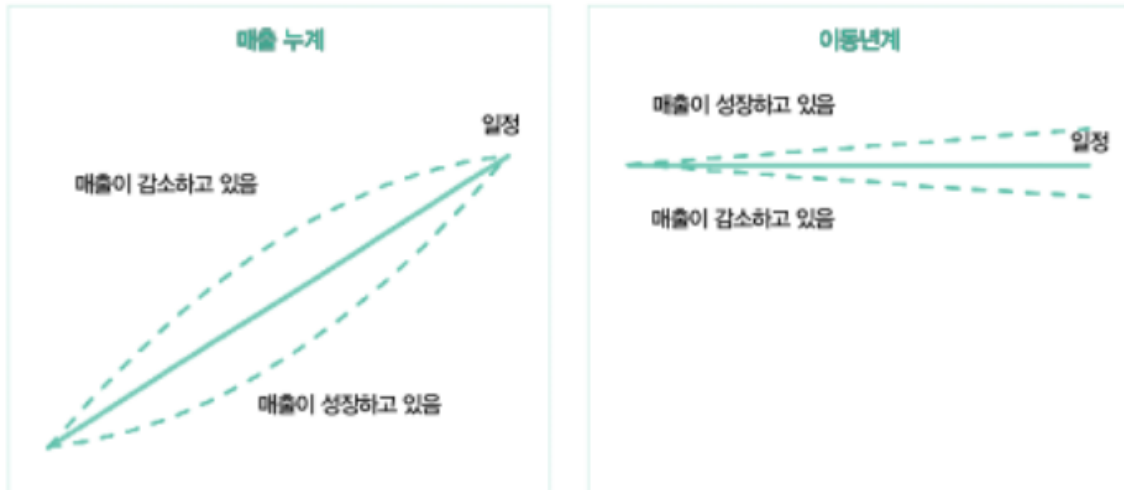
2016년												2017년		
1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월	1월	2월	3월
합계														
	합계													
		합계												

- 매출누계 : 해당 월의 매출에 이전월까지의 매출 누계 합한 것

2016년												2017년		
1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월	1월	2월	3월
합계														
	합계													
		합계												

- 이동년계 : 해당 월의 매출에 과거 11개월의 매출을 합한 것

2015년												2016년		
1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월	1월	2월	3월
	2016년 1월의 이동년계													
		2016년 2월의 이동년계												
			2016년 3월의 이동년계											



- 왼쪽: 매출 일정
- 가운데: 모두 증가하는 방향 ⇒ 매출 성장
- 오른쪽: 이동년계만 내려감 ⇒ 현재는 매출이 일정하지만, 작년에는 매출이 성장하여, 작년 대비 낮아짐

```
WITH
--일별 구매
DAILY_PURCHASE AS (
    SELECT
        DT
```

```

        ,SUBSTRING(DT,1,4) AS YEAR
        ,SUBSTRING(DT,6,2) AS MONTH
        ,SUBSTRING(DT,9,2) AS DATE
        ,SUM(PURCHASE_AMOUNT) AS PURCHASE_AMOUNT
        ,COUNT(ORDER_ID) AS ORDERS
    FROM PURCHASE_LOG
    GROUP BY DT
)
-- 월별 매출액
,MONTHLY_AMOUNT AS (
SELECT
    YEAR
    ,MONTH
    ,SUM(PURCHASE_AMOUNT) AS AMOUNT
    FROM DAILY_PURCHASE
    GROUP BY YEAR, MONTH
)
, CALC_INDEX AS (
SELECT
    YEAR
    ,MONTH
    -- 월 매출
    ,AMOUNT
    -- 누계매출
    ,SUM(CASE WHEN YEAR='2015' THEN AMOUNT END)
      OVER(ORDER BY YEAR, MONTH ROWS UNBOUNDED PRECEDING)
      AS AGG_AMOUNT
    -- 이동년계
    ,SUM(AMOUNT)
      OVER(ORDER BY YEAR, MONTH ROWS BETWEEN 11 PRECEDING AND CURRENT ROW)
      AS YEAR_AVG_AMOUNT
    FROM
        MONTHLY_AMOUNT //수정
    ORDER BY YEAR,MONTH
)
SELECT
    CONCAT(YEAR, '-', MONTH) AS YEAR_MONTH
    ,AMOUNT
    ,AGG_AMOUNT
    ,YEAR_AVG_AMOUNT
FROM CALC_INDEX
WHERE YEAR='2015'
ORDER BY YEAR_MONTH;

```

	year_month text	amount numeric	agg_amount numeric	year_avg_amount numeric
1	2015-01	22111	22111	160796
2	2015-02	11965	34076	144292
3	2015-03	20215	54291	145608
4	2015-04	11792	66083	145006
5	2015-05	18087	84170	160811
6	2015-06	18859	103029	169490
7	2015-07	14919	117948	180382
8	2015-08	12906	130854	187045
9	2015-09	5696	136550	188909
10	2015-10	13398	149948	195591
11	2015-11	6213	156161	185360
12	2015-12	26024	182185	182185

매출을 파악할 때 중요 포인트

매출 상승/하락 원인을 파악하는 것이 중요.

EX) 판매 횟수의 변화 → 방문 횟수, 상품 수, 회원 등록 수 확인

```

WITH
--일별 구매
DAILY_PURCHASE AS (
    SELECT
        DT
        , SUBSTRING(DT,1,4) AS YEAR
        , SUBSTRING(DT,6,2) AS MONTH
        , SUBSTRING(DT,9,2) AS DATE
        , SUM(PURCHASE_AMOUNT) AS PURCHASE_AMOUNT
        , COUNT(ORDER_ID) AS ORDERS
    FROM PURCHASE_LOG
    GROUP BY DT
)
--월별 구매
,MONTHLY_PURCHASE AS (
    SELECT
        YEAR
        , MONTH
        , SUM(ORDERS) AS ORDERS
        , AVG(PURCHASE_AMOUNT) AS AVG_AMOUNT
        , SUM(PURCHASE_AMOUNT) AS MONTHLY
    FROM DAILY_PURCHASE
    GROUP BY YEAR, MONTH
)

SELECT
    CONCAT(YEAR, '-', MONTH) AS YEAR_MONTH
    , ORDERS
    , AVG_AMOUNT

```

```

-- 월별 매출액
, MONTHLY
-- 매출 누계
, SUM(MONTHLY)
  OVER(PARTITION BY YEAR
        ORDER BY MONTH ROWS UNBOUNDED PRECEDING)
  AS AGG_AMOUNT
-- 12개월 전 매출액
, LAG(MONTHLY, 12)
  OVER(ORDER BY YEAR, MONTH)
  AS LAST_YEAR
-- 12개월 전의 매출과 비교하여 비율
, 100.0
  *MONTHLY
  /LAG(MONTHLY, 12)
  OVER(ORDER BY YEAR, MONTH)
  AS RATE
FROM MONTHLY_PURCHASE
ORDER BY YEAR_MONTH;

```

	year_month text	orders numeric	avg_amount numeric	monthly numeric	agg_amount numeric	last_year numeric	rate numeric
1	2014-01	1	13900.0000000000000000	13900	13900	[null]	[null]
2	2014-02	1	28469.0000000000000000	28469	42369	[null]	[null]
3	2014-03	1	18899.0000000000000000	18899	61268	[null]	[null]
4	2014-04	1	12394.0000000000000000	12394	73662	[null]	[null]
5	2014-05	1	2282.0000000000000000	2282	75944	[null]	[null]
6	2014-06	1	10180.0000000000000000	10180	86124	[null]	[null]
7	2014-07	1	4027.0000000000000000	4027	90151	[null]	[null]
8	2014-08	1	6243.0000000000000000	6243	96394	[null]	[null]
9	2014-09	1	3832.0000000000000000	3832	100226	[null]	[null]
10	2014-10	1	6716.0000000000000000	6716	106942	[null]	[null]
11	2014-11	1	16444.0000000000000000	16444	123386	[null]	[null]
12	2014-12	1	29199.0000000000000000	29199	152585	[null]	[null]
13	2015-01	1	22111.0000000000000000	22111	22111	13900	159.0719424460431655
14	2015-02	1	11965.0000000000000000	11965	34076	28469	42.0281709930099406
15	2015-03	1	20215.0000000000000000	20215	54291	18899	106.9633313931954072

- LAG(MONTHLY,12) OVER(ORDER BY YEAR, MONTH)
 - MONTHLY열의 12번째 이전 행의 값을 가져옴
 - 순서가 정의된 열에서 특정 행의 이전 행에 있는 값을 가져올 때 유용