

# 롯데마트 CRM 데이터 분석 교육

고객분석  
USING SQL

~ 20141227

2014년 12월

> RE::VISION

전용준 리비전컨설팅 대표  
010.3095.1451  
[xyxonxyxon@empal.com](mailto:xyxonxyxon@empal.com)  
<http://cafe.daum.net/revisioncrm>  
<http://www.revisioncon.co.kr>

1

# DAY 1.1

**1B ::**

## SAS 분석도구 소개 및 데이터 처리 기초

# SAS : Statistical Analysis System

- SAS는 압도적인 시장 점유율 1위의 분석 소프트웨어
- 대규모 데이터 통계 처리에 강점
- 데이터 집계, 가공, 연결, 예측, 시각화, 통계치 산출 등 다양한 기능제공
- SAS 고유의 Script 언어를 사용
- RDBMS와는 다른 고유한 데이터 저장 방식 사용
- 실제 데이터 분석에서 데이터 처리에는 ANSI SQL의 변형인 Proc SQL 주로 사용

# SAS 활용한 데이터 처리 기초

- 데이터 로딩
- 간단한 데이터 처리 (조회, Sort, 집계, 선택)
- 데이터 타입 변환

# SAS 활용의 첫단계

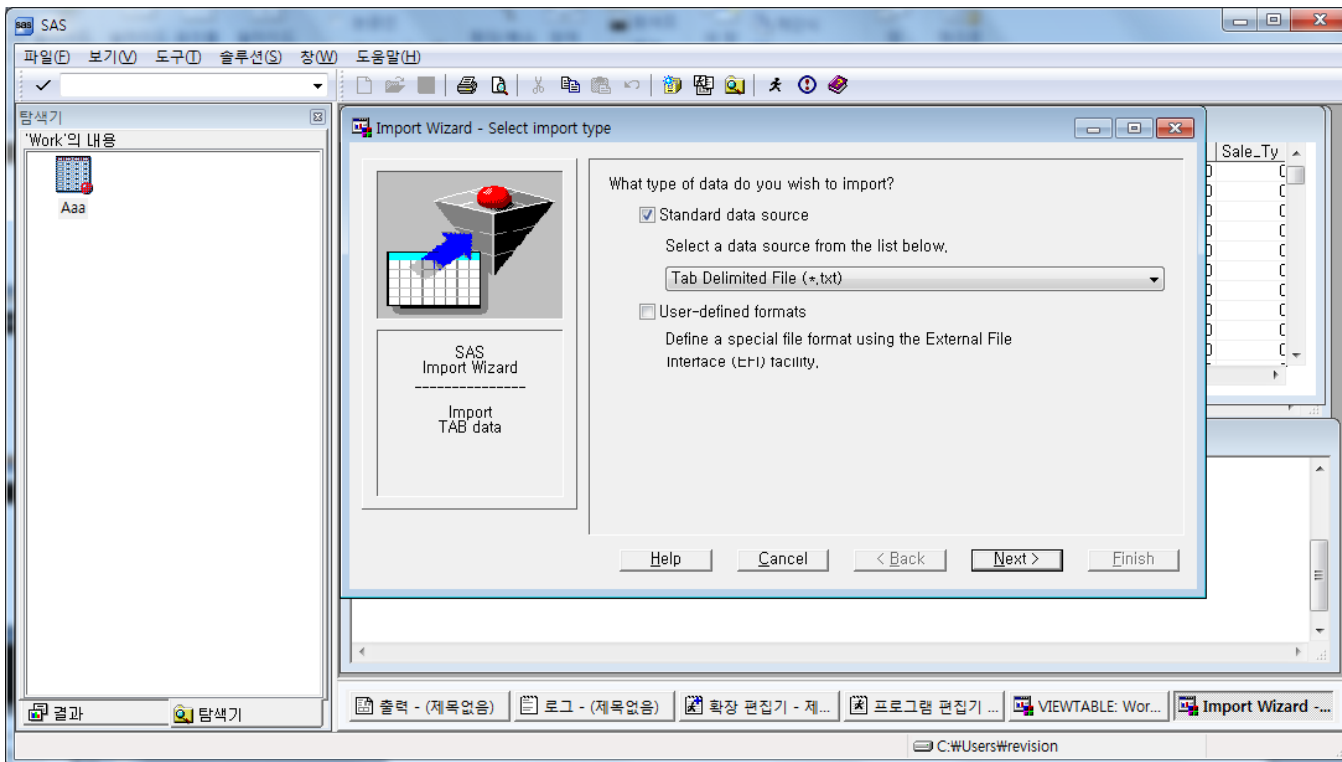
- 라이브러리 생성/지정
- 로컬의 텍스트 파일 데이터 불러오기
- 서버의 데이터 가져오기
- 데이터 내보내기
  
- 변수 정의
- 간단한 계산

## 데이터 테이블 생성시 Tip

- 변수명은 32자 이내
- 숫자로 시작 불가!
- 특수기호 사용 불가! (except "\_")
- 자료의 입력은 되도록 숫자변수로
- 주민등록번호나 전화번호 등은 문자변수로 입력
- 날짜변수는 mm dd yy로 나누어서 입력
  - 한 셀에 입력했을 경우에는 SAS에서 구분 가능
- 문자변수는 되도록 자릿수를 맞춰서 입력

# 로컬의 텍스트 (ASCII) 파일 데이터 불러오기

- 메뉴에서 [ 파일 > 데이터 가져오기 > ] 실행시 위저드가 실행됨
- 파일형식(주로 CSV 또는 TDF)을 지정해주고, 파일 위치를 지정해서 선택
  - <141205\_SAS교육샘플 데이터> 라는 이름의 텍스트 파일





## 스크립트로 데이터 불러오기

- 라이브러리명과 파일이름을 지정하고, 데이터 위치와 데이터파일 형식을 지정
- 주로 CSV 형식의 텍스트 파일을 불러오는 방식을 사용 (엑셀 버전이나 연동관련된 이슈가 없음)
- GETNAMES 명령을 지정해 데이터 파일로부터 필드명을 바로 가져다 사용

```
PROC IMPORT OUT= WORK.aaa02  
DATAFILE=  
    "C:\Users\wrevision\Desktop\wvc_kb\02_EnGageMent\01_13_롯데마  
    트분석교육\01_test.csv"  
REPLACE;  
GETNAMES=YES;  
RUN;
```

# 스크립트로 데이터 불러오기 1

- 라이브러리명과 파일이름을 지정하고, 데이터 위치와 데이터파일 형식을 지정
- 주로 CSV 형식의 텍스트 파일을 불러오는 방식을 사용 (엑셀 버전이나 연동관련된 이슈가 없음)
- GETNAMES 명령을 지정해 데이터 파일로부터 필드명을 바로 가져다 사용

```
PROC IMPORT OUT= LM_POS01.aaa02  
DATAFILE= "C:\WSASTEST_201412\Wa01_test.csv"  
REPLACE;  
GETNAMES=YES;  
RUN;
```

## CARDS 명령문을 사용해 직접 입력

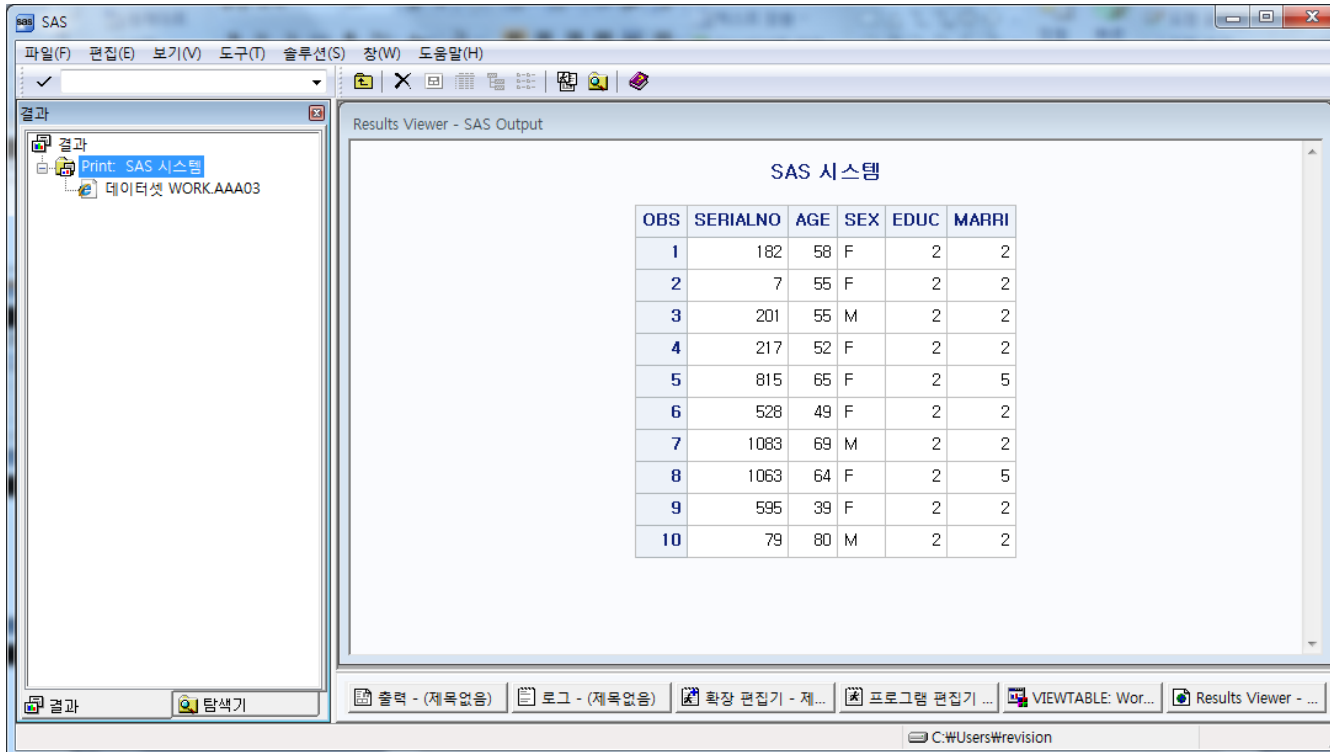
- 간단한 데이터의 경우 구조를 지정하고, CARDS 문을 사용하여 직접 입력 가능

```
DATA WORK.AAA03;  
INPUT SERIALNO AGE SEX $ EDUC MARRI;  
LABEL SEX='성별';  
CARDS;  
182 58 F 2 2  
7 55 F 2 2  
201 55 M 2 2  
217 52 F 2 2  
815 65 F 2 5  
528 49 F 2 2  
1083 69 M 2 2  
1063 64 F 2 5  
595 39 F 2 2  
79 80 M 2 2  
; RUN;
```

# PRINT 문으로 데이터 확인

- 탐색기상의 아이콘을 클릭하여 테이블보기로 보거나,
- 생성된 데이터(테이블)의 내용을 확인 - 화면에 프린트/출력

```
PROC PRINT DATA=AAA03; RUN;
```

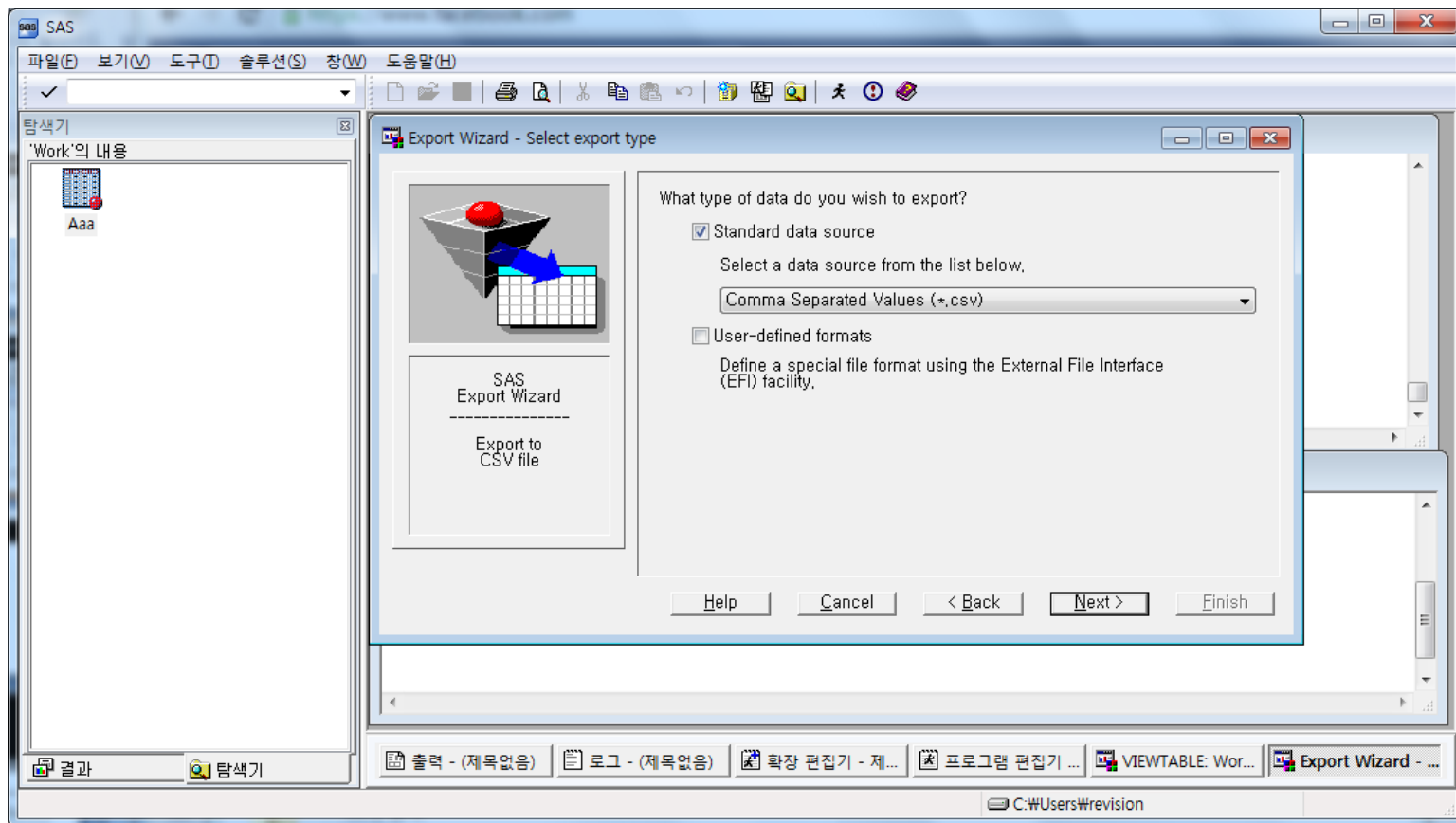


The screenshot shows the SAS Results Viewer window. The left pane displays the project structure with 'Print: SAS 시스템' and '데이터셋 WORK.AAA03'. The main pane shows the output of the PROC PRINT statement, which is a table titled 'SAS 시스템'.

OBS	SERIALNO	AGE	SEX	EDUC	MARRI
1	182	58	F	2	2
2	7	55	F	2	2
3	201	55	M	2	2
4	217	52	F	2	2
5	815	65	F	2	5
6	528	49	F	2	2
7	1083	69	M	2	2
8	1063	64	F	2	5
9	595	39	F	2	2
10	79	80	M	2	2

# 로컬PC로 내보내기 - CSV 파일

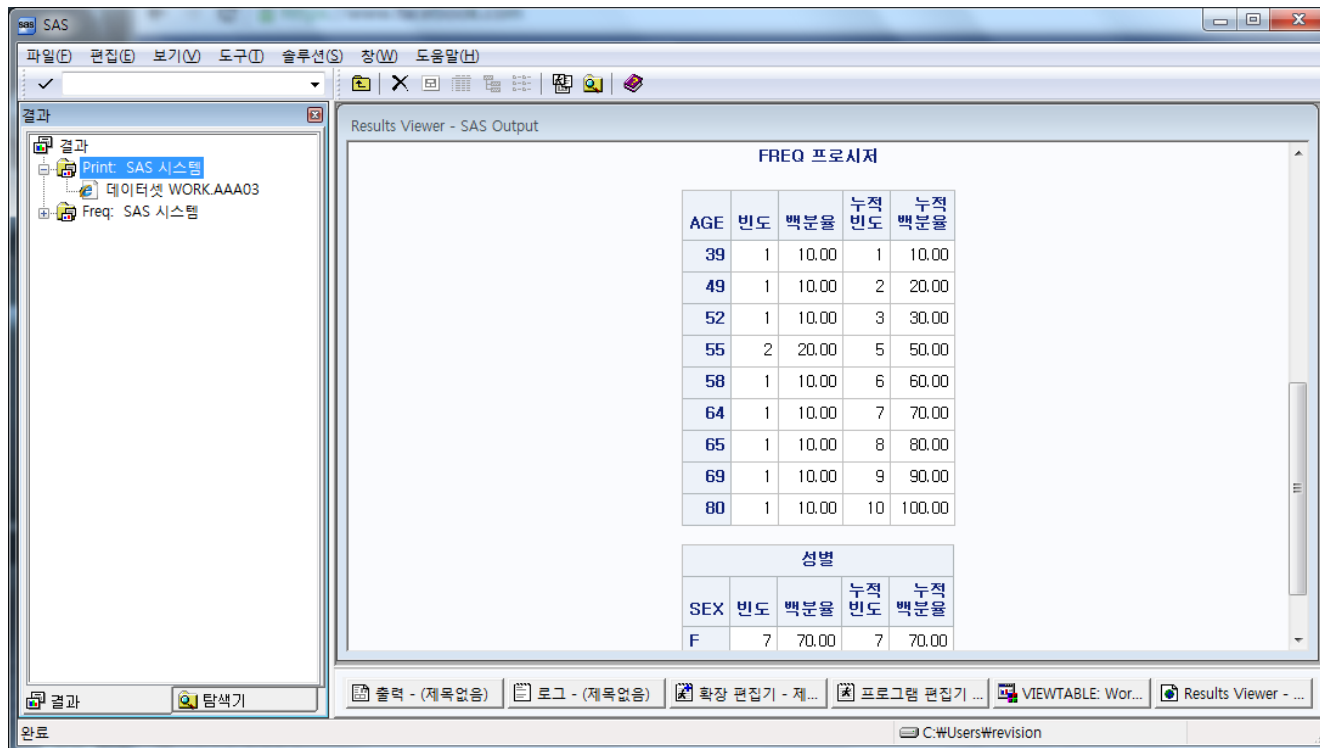
- 메뉴에서 [ 파일 > 데이터 내보내기 > ] 실행시 위저드가 실행됨
- 파일형식을 지정해준 후 (주로 엑셀에서 사용하므로 CSV로 설정 ) 파일 위치를 지정



# 데이터 특성에 대한 기초적 확인

- PROC FREQ 를 사용하여 필드별 값별 빈도수를 확인

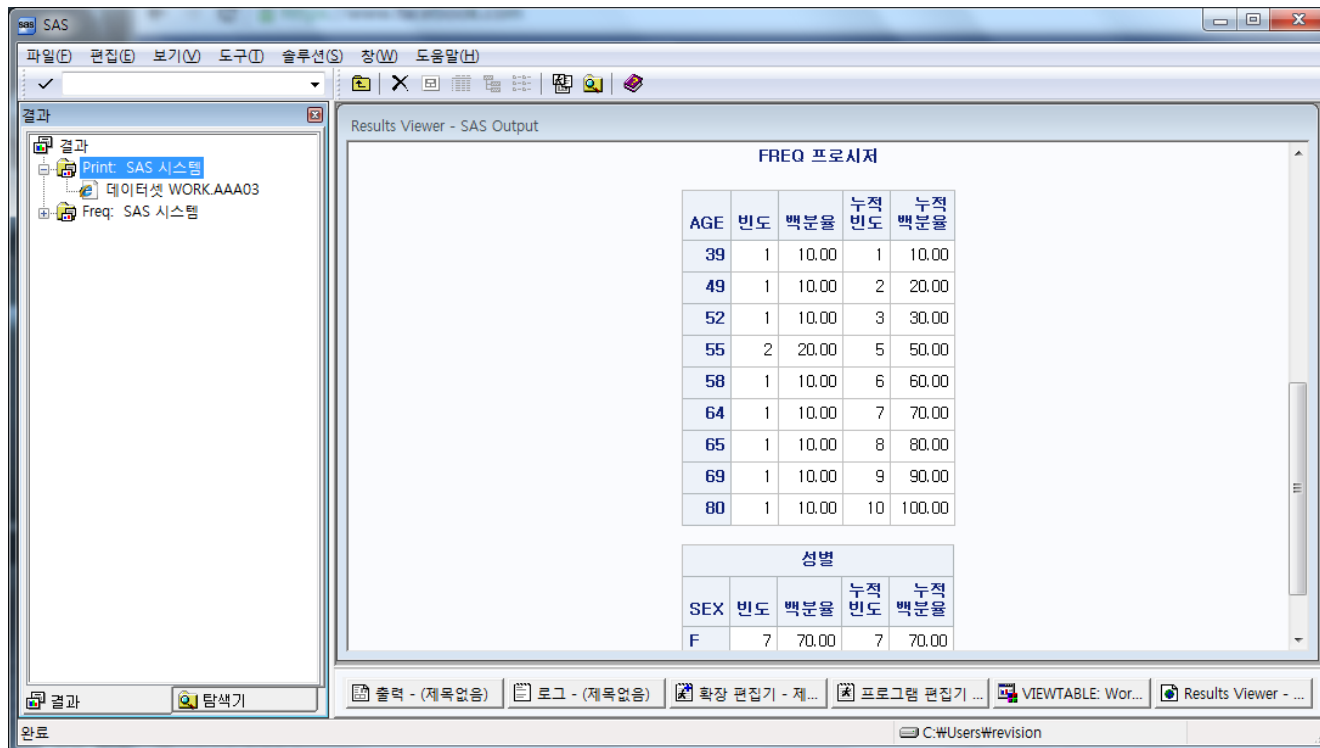
```
PROC FREQ DATA=AAA03;  
TABLES age sex; RUN;
```



# 데이터 특성에 대한 기초적 확인

- PROC FREQ 를 사용하여 필드별 값별 빈도수를 확인

```
PROC FREQ DATA=AAA03;  
TABLES age sex; RUN;
```



# 데이터 특성에 대한 기초적 확인

- PROC FREQ 를 사용하여 실습용 실제 POS 데이터 필드별 빈도수를 확인

```
PROC FREQ DATA=LM_POS01.AAA02;
TABLES str_cd pos_sys_dy ; RUN;
```

The screenshot shows the SAS Results Viewer window with two tables of data. The first table shows frequencies for str\_cd and pos\_sys\_dy. The second table shows frequencies for pos\_sys\_dy and bin counts.

str_cd	pos_sys_dy	빈도	백분율	누적 빈도	누적 백분율
607	115	11.50	66.50		
608	78	7.80	74.30		
702	73	7.30	81.60		
703	26	2.60	84.20		
946	1	0.10	84.30		
948	54	5.40	89.70		
949	21	2.10	91.80		
981	82	8.20	100.00		

Pos_Sys_Dy	빈도	백분율	누적 빈도	누적 백분율
20141124	252	25.20	252	25.20
20141125	184	18.40	436	43.60
20141126	381	38.10	817	81.70
20141127	146	14.60	963	96.30
20141128	37	3.70	1000	100.00

- Str\_cd (점포코드), Pos\_Sys\_dy (포스 기준 일자) 두 필드에 대한 값 분포를 확인
- 20141124~20141128 (총5일간)로부터 추출된 데이터임을 확인
- 필드의 특성(값 분포)을 확인



## 라이브러리의 지정

- 라이브러리란? ➔ SAS dataset이 저장되는 (논리적인) 장소
- Dataset 이름 사용 형식
  - <library.dataset>
  - library : 영문 8자 이내
  - dataset : 영문 32자 이내
  - 대소문자 구분 없음, 특수문자 사용 불가(예외:\_)
  - 맨 첫 글자에 숫자 사용 불가
- Library name 지정
  - LIBNAME 라이브러리 이름 '위치';

```
LIBNAME LM_POS01 'C:\SASTEST_201412';
```

## 샘플 데이터 셋의 구조

- 1000건의 거래 내역을 포함한 POS 이력

• STR_CD	점포코드
• SALE_DY	판매일자
• POS_NO	포스번호
• TRD_NO	거래번호
• CUST_NO	고객번호
• CANCEL_FG	취소여부
• CUST_PNT	고객포인트
• SALE_SALE_AMT	판매금액
• SALE_PROFIT_AMT	이익금액
• SALE_DC_AMT	할인금액
• SALE_EMP_NO	판매직원번호

# DAY 2

2A ::

## SAS PROC SQL

참조 :: [마트SAS] SQL 레벨001 -- 다음블로그  
( 검색창에서 [ 마트 sas sql ] 로 검색 )

<http://blog.daum.net/revisioncrm/270>

# SQL 데이터 처리의 표준언어

- SQL : Standard Query Language
- 데이터의 조회, 정의, 제어의 세가지 기능을 포함
  - 데이터 정의 언어 (DDL : Data Definition Language)
  - 데이터 조작 언어 (DML : Data Manipulation Language)
  - 데이터 제어 언어 (DCL : Data Control Language)



```
PROC SQL;  
SELECT * FROM LM_POS01.AAA02  
WHERE STR_CD=402 AND POS_NO=6 ;  
QUIT;
```

- 단순히 기존의 테이블로부터 특정 조건에 맞는 행을 추출

```
PROC SQL;  
CREATE TABLE LM_POS01.AAA021 AS  
SELECT STR_CD, SALE_DY FROM LM_POS01.AAA02  
WHERE STR_CD=402 AND POS_NO<10 ;  
QUIT;
```

- 일부 필드와 조건에 맞는 행만으로 새로운 테이블 생성

## 연습 :: 취소건만 가지고 오려면?

- SQL : Standard Query Language
- 데이터의 조회, 정의, 제어의 세가지 기능을 포함
  - 데이터 정의 언어 (DDL : Data Definition Language)
  - 데이터 조작 언어 (DML : Data Manipulation Language)
  - 데이터 제어 언어 (DCL : Data Control Language)

```
PROC SQL;  
CREATE TABLE LM_POS01.AAA022 AS  
SELECT STR_CD, SALE_DY, SALE_SALE_AMT  
FROM LM_POS01.AAA02  
WHERE CANCEL_FG = 2 AND STR_CD=402 ;  
QUIT;
```

- CANCEL\_FG 값중  
취소건을 확인해  
서 조건을 지정

## 테이블의 생성

- CREATE TABLE 문은 라이브러리와 테이블명을 지정해주면 해당 테이블을 생성하는 명령
- 주의 : 모든 필드를 선택하여 새로운 테이블을 반복해서 생성하게 되면 기하급수적으로 데이터 물리적 사이즈 증가

```
PROC SQL;  
CREATE TABLE LM_POS01.STRPOS AS  
SELECT STR_CD, POS_NO  
FROM LM_POS01.AAA02;  
QUIT;
```

## SAS 언어와 SQL 혼합 지원

- 표준 SQL(ANSI SQL)에 SAS 고유의 언어를 혼용 가능
- SAS keyword expressions (예: DROP, KEEP, RENAME, WHERE)를 SQL에서 사용 가능함

```
PROC SQL;  
CREATE TABLE LM_POS01.AAA022 AS  
SELECT *  
FROM AAA02(DROP=STR_CD );  
QUIT;
```

- AAA02 테이블에서 STR\_CD 필드는 제외하고 나머지 필드들만 가져옴

# Duplicates 중복 제거

- DISTINCT 문은 같은 필드에 반복해서 나오는 항목을 제거하고 종류만 골라내는 기능

```
PROC SQL;  
CREATE TABLE LM_POS01.TEMP001 AS  
SELECT STR_CD  
FROM LM_POS01.AAA02;  
QUIT;
```

```
PROC SQL;  
CREATE TABLE LM_POS01.STR_CD_TB AS  
SELECT DISTINCT STR_CD  
FROM LM_POS01.AAA02;  
QUIT;
```

```
PROC SQL;  
CREATE TABLE LM_POS01.STR_CD_TB003 AS  
SELECT DISTINCT STR_CD, SALE_DY  
FROM LM_POS01.AAA02;  
QUIT;
```

- 점포코드만 리스트를 추출하여 테이블로 생성



## Sorting 정렬

- ORDER BY 문을 사용하여 데이터 정렬
- Default는 오름차순, DESC를 주면 역순 (내림차순)

```
PROC SQL;  
CREATE TABLE LM_POS01.STR_CD_TB AS  
SELECT DISTINCT STR_CD, POS_NO  
FROM LM_POS01.AAA02  
ORDER BY STR_CD, POS_NO DESC;  
QUIT;
```

- STR\_CD는 오름차순, POS\_NO는 내림차순으로 정렬한 테이블 생성

## 선택 및 연산 Sub-setting and Calculating

- +, -, \*, / 등의 사칙 연산을 사용 가능
- () 를 사용해서 반복적인 표현 사용 가능
- WHERE 절에 IN() 함수를 사용하여 복수의 값 선택 가능

```
PROC SQL;
CREATE TABLE LM_POS01.STR_CD_TB01 AS
SELECT STR_CD, SALE_DY,
(SALE_SALE_AMT-100)/1000 AS TEMP_SALEAMT
FROM LM_POS01.AAA02
WHERE STR_CD IN (302 322)
ORDER BY STR_CD, SALE_DY;
QUIT;
```

- 통상적인 사칙연산을 SELECT 절에 적용하여 새로운 필드 생성 가능
- WHERE 절을 이용하여 일부 값에 맞는 레코드들만 선별 가능

# Aggregation

- GROUP BY 는 분류별 (예: 지역, 점포, 연도, 고객등급 등) 진계에 주로 사용되는 구문
- COUNT(), SUM(), AVG() 등이 집계(즉, “대표값” 계산)를 위해 많이 사용됨

```
PROC SQL;  
CREATE TABLE LM_POS01.STR_CD_TB AS  
SELECT STR_CD, AVG(SALE_SALE_AMT) AS SSAMT_AVG  
FROM LM_POS01.AAA02  
GROUP BY STR_CD;  
QUIT;
```

## GROUP BY 분류별 집계 활용

- 집계기준으로 복수의 키를 사용할 수 있음 (예: 점포별 > 주별 > ... )
- 집계 키에서는 사용되는 순서대로 순서가 적용됨 (오름차, 내림차 정렬에 따라 결과가 달라짐에 유의 필요)

```
PROC SQL;  
CREATE TABLE LM_POS01.STR_DY_SSAMT001 AS  
SELECT STR_CD, SALE_DY  
SUM(SALE_SALE_AMT)/1000 AS STR_SALEAMT  
FROM LM_POS01.AAA02  
GROUP BY STR_CD, SALE_DY  
ORDER BY SALE_DY DESC, STR_CD;  
QUIT;
```

## 연습 :: Aggregation

- 평균, 합계, 건수 기초적으로 사용되는 "대표값" 산출방법 예: 건단가 10만원 초과 고객 머릿수 산출

### PROC SQL;

```
CREATE TABLE LM_POS01.CUST_TMP001 AS  
SELECT CUST_NO, AVG(SALE_SALE_AMT) AS SSAMT_AVG  
FROM LM_POS01.AAA02  
GROUP BY CUST_NO;
```

### QUIT;

### PROC SQL;

```
CREATE TABLE LM_POS01.CUST_TMP002 AS  
SELECT CUST_NO, SSAMT_AVG  
FROM LM_POS01.CUST_TMP001  
WHERE SSAMT_AVG > 100000  
ORDER BY SSAMT_AVG DESC ;
```

### QUIT;

## 연습 :: Aggregation 2

- 평균, 합계, 건수 기초적으로 사용되는 "대표값" 산출방법 예: 건단가 10만원 초과 고객 머릿수 산출

### PROC SQL;

```
CREATE TABLE LM_POS01.CUST_TMP001 AS
SELECT CUST_NO, AVG(SALE_SALE_AMT) AS SSAMT_AVG
FROM LM_POS01.AAA02
GROUP BY CUST_NO;
```

QUIT;

- 고객별 건별금액 평균값 산출

### PROC SQL;

```
CREATE TABLE LM_POS01.CUST_TMP003 AS
SELECT COUNT(*) AS CNT
FROM LM_POS01.CUST_TMP001
WHERE SSAMT_AVG > 100000 ;
```

QUIT;

- 일정 금액 이상 고객의 수 산출

- 일부고객의 매출  
합계 산출

### PROC SQL;

```
CREATE TABLE LM_POS01.SALE_SUM01
AS
SELECT SUM(SALE_SALE_AMT) AS
SSA_SUM
FROM LM_POS01.AAA02
WHERE SSAMT_AVG > 100000 ;
QUIT;
```

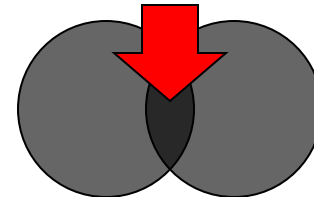
# Joining Tables

- JOIN 은 두개 이상의 테이블이 가진 데이터를 연결시켜 하나의 테이블을 생성하는 명령
  - INNER, OUTER(LEFT), FULL 세가지 존재

## INNER JOIN

```
PROC SQL;  
CREATE TABLE LM_POS01.STR_CD_TB03 AS  
SELECT A.*, B.*  
FROM LM_POS01.STR_CD_TB01 A INNER JOIN LM_POS01.STR_CD_TB02 B  
ON (A.STR_CD=B.STR_CD)  
WHERE A.STR_CD IN (302 322);  
QUIT;
```

- 순서 무관하게 연결하는 두 테이블 모두에 존재하는 레코드만을 결과로 반환

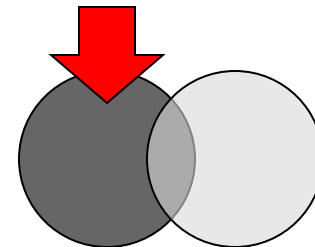


# LEFT JOIN

- LEFT JOIN은 한 쪽을 기준으로 다른 테이블의 레코드 중 일치하는 부분만 결합
- 예: 매장포스에 거래내역 있는 온라인 회원의 목록을 포스 거래 내역과 결합 (기준은 매장 포스 거래내역)

## LEFT JOIN

```
PROC SQL;  
CREATE TABLE LM_POS01.STR_CD_TB04 AS  
SELECT A.*, B.*  
FROM LM_POS01.STR_CD_TB01 A LEFT JOIN LM_POS01.STR_CD_TB02 B  
ON (A.STR_CD=B.STR_CD)  
WHERE A.STR_CD IN (302 322);  
QUIT;
```





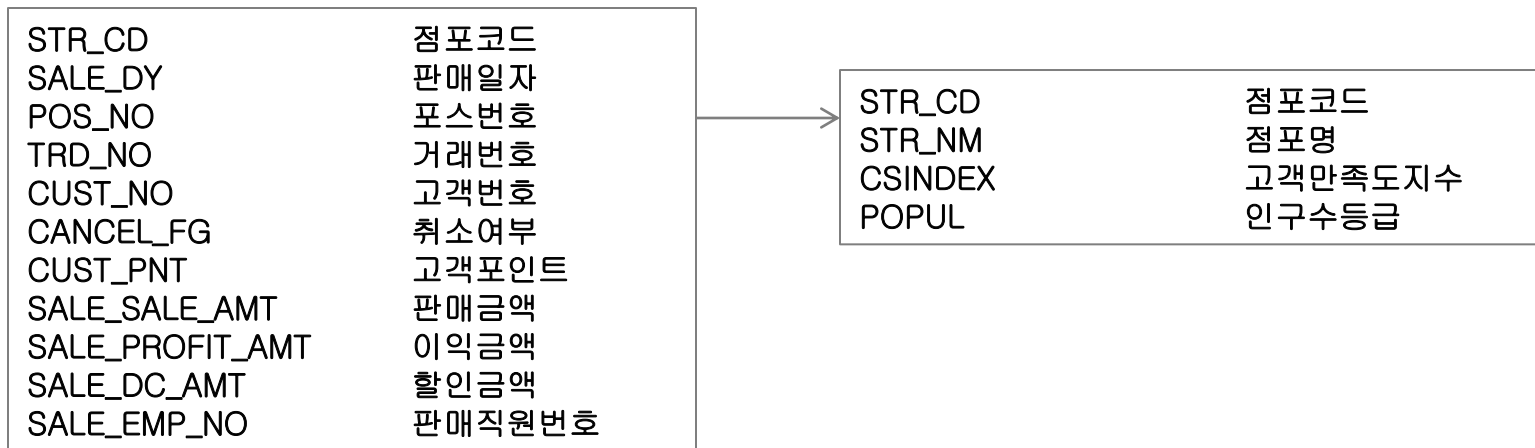
## WEEK 2 연습문제 ::

**[문제 1]** 점포별 취소건의 고객포인트 최대값과 최소값을 산출하라

**[문제 2]** 고객만족도 지수 50점 초과 점포에 한하여 할인금액의 일자별 합계를 산출하여 점포명을 함께 표시한 테이블을 생성하라

**[문제 3]** 점포별 일자별 취소율(취소건의 비율)을 산출하라

**[문제 4]** 매출합계는 점포중 5위 이내이면서, 인구수 등급은 중간에 해당하는 점포를 찾아라

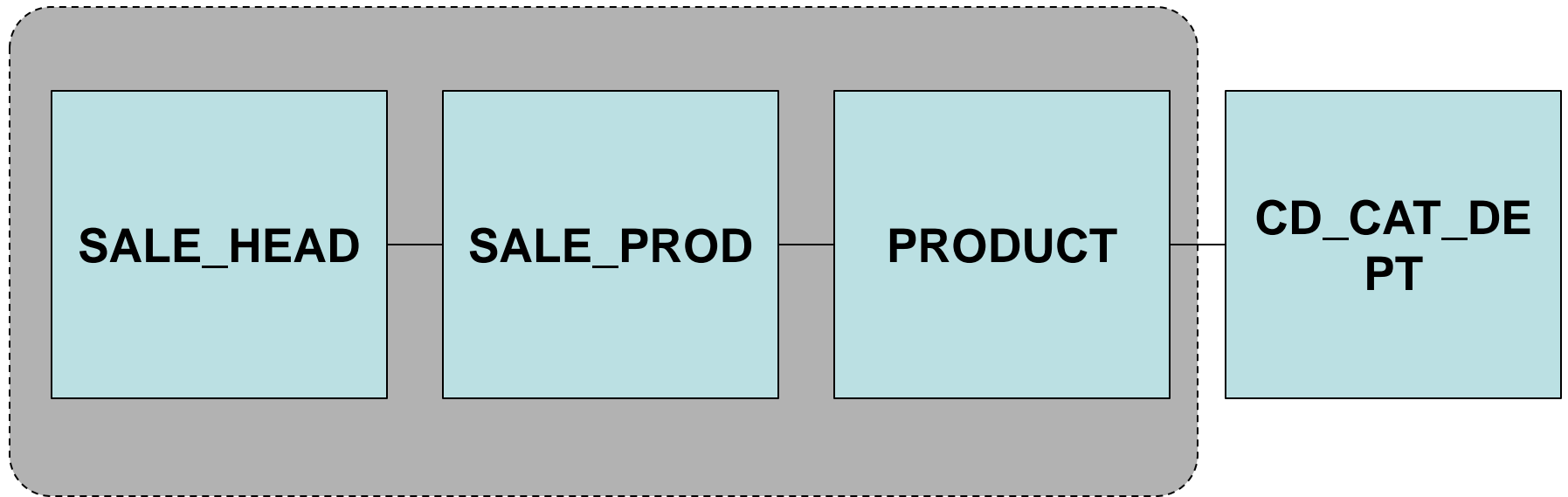


# DAY 2+

2A+ ::

SQL

# THE NEW POS DATA SET



POS 영수증

# IMPORTING THE NEW POS DATA SET

```
LIBNAME LM_POS01 'C:\SASTEST_201412';
```

```
PROC IMPORT OUT= LM_POS01.SALE_HEAD  
DATAFILE= "C:\SASTEST_201412\sale_head_201412.csv"  
REPLACE;  
GETNAMES=YES; RUN;
```

```
PROC IMPORT OUT= LM_POS01.SALE_PROD  
DATAFILE= "C:\SASTEST_201412\sale_prod_201412.csv"  
REPLACE;  
GETNAMES=YES; RUN;
```

```
PROC IMPORT OUT= LM_POS01.PRODUCT  
DATAFILE= "C:\SASTEST_201412\product.csv"  
REPLACE;  
GETNAMES=YES; RUN;
```

```
PROC IMPORT OUT= LM_POS01.CD_CAT_DEPT  
DATAFILE= "C:\SASTEST_201412\cd_cat_dept.csv"  
REPLACE;  
GETNAMES=YES; RUN;
```

## 반복적인 JOIN 응용 연습

- 고객별로 몇 개의 상품을 구매했는가?
- 가장 많은 개수의 상품을 구매한 점포는?
- 상품군별로 고객수는?
- 취소 고객수가 가장 많은 상품군명은?

## SUBSTR() 함수 활용

- SUBSTR(필드명, 시작자리, 문자열길이)

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP008 AS  
SELECT PROD_NM, SUBSTR(PROD_NM,3,6) AS SPROD_NM  
FROM LM_POS01.PRODUCT  
;  
QUIT;
```

## CASE 문의 활용

- CASE문은 조건부로 값을 계산하기 위해 사용
- CASE WHEN ... THEN ... WHEN ... THEN .... ELSE ... END의 형식

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP009 AS  
SELECT TRD_NO, CUST_NO, SALE_SALE_AMT,  
CASE WHEN SALE_SALE_AMT > 50000 THEN 1  
ELSE 0 END  
AS IS_BIG  
FROM LM_POS01.SALE_HEAD;  
QUIT;
```

## UPDATE 문을 활용한 NULL 처리

- UPDATE문은 값을 변경시키기 위해 사용 (예: 일괄적으로 NULL을 채우기)
- UPDATE 문은 수정이므로 반복해서 일부분을 수정해서 계층적으로 만드는 것도 가능

### **PROC SQL;**

```
UPDATE LM_POS01.SPTMP009  
SET IS_BIG = 3  
WHERE SALE_SALE_AMT > 30000 ;  
QUIT;
```

### **PROC SQL;**

```
UPDATE LM_POS01.SPTMP009  
SET IS_BIG = 8  
WHERE SALE_SALE_AMT > 80000 ;  
QUIT;
```



## 연습문제 :: 20141220

- **[연습1]** 구매금액의 합계는 5위이내이나 구매한 고객수는 비교적 작은 상품군은 무엇인가? 세가지 부문의 이름을 파악하라
- **[연습2]** '신선'으로 시작되는 명칭을 가진 DEPT의 구매고객수와 이익률은 얼마인가?
- **[연습3]** 일자별로 구매단품수를 기준으로 상위 10%에 해당하는 고객의 절대 인원수와 해당 고객의 리스트 (CUST\_NO)
- **[연습4]** 구매건수(영수증 기준)는 많지 않으나, 한 영수증건당 구매한 고객수는 전체 점포 중 상위 30%이내에 해당하는 많은 구매 고객수를 가진 점포의 명칭 명칭을 조회하라
- **[연습5]** 고객별로 총이용액과 포인트 적립한 금액을 합산해서 산출하고, 총이용액 대비 포인트 금액의 비율이 가장 높은 5명의 고객리스트를 추출하라
- **[연습6]** 5명의 고객을 구분하기 위한 필드를 BC001 이라 하고, 7% 이상의 총이용액 대비 포인트 금액이 되는 고객을 모두 추출하라
- **[연습7]** 가장 고객수가 많은 점포에서 평균적인 고객별 구매 일자(날짜) 수가 가장 높은(자주구매하는) 상품군 DEPT의 명칭을 찾아라
- **[연습8]** 전체 구매한 고객중 2일 이상 구매한 고객의 수 비율은 몇%인가?

# SAS 남은 항목들

- Data Step 중요 몇 가지 사용방법
- Data type 변환
  - Sting, Numeric, Datetime
  - Update 문 활용
- SQL 추가 활용방법
  - Nested Query 작성
  - 복수의 테이블 Left Join
  - 함수 활용 : MONOTONIC, UNIQUE, RANUNI, NMISS
- 응용
  - 기간 구분한 예측모델 데이터 셋 준비
  - 실전적인 탐색적 데이터 분석 연습

## 주요 집계 함수 : MAX, AVG, MIN, CV, COUNT

- 대표적인 집계 함수는 최대, 최소, 평균, 레코드 수, 변동성 정도
- $CV = \text{Coefficient of Variance} = \text{표준편차} / \text{평균}$

```
PROC SQL;
CREATE TABLE LM_POS01.SPTMP011 AS
SELECT DISTINCT IS_BIG, MAX(SALE_SALE_AMT) AS MAX_SSAMT,
    AVG(SALE_SALE_AMT) AS AVG_SSAMT,
    MIN(SALE_SALE_AMT) AS MIN_SSAMT,
    CV(SALE_SALE_AMT) AS CV_SSAMT,
    COUNT(*) AS CNT_SSAMT
FROM LM_POS01.SPTMP009
GROUP BY IS_BIG;
QUIT;
```

## 문자 – 숫자 변환 함수 활용

- 문자 ➔ 숫자 :: INPUT ; 숫자 ➔ 문자 :: PUT
  - 자릿수의 적절한 지정 필요

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP012 AS  
SELECT IS_BIG, PUT(IS_BIG*100, 3.) AS IS_BIG01  
FROM LM_POS01.SPTMP011;  
QUIT;
```

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP013 AS  
SELECT IS_BIG, IS_BIG01, SUBSTR(IS_BIG01, 2,2) AS IS_BIG01A,  
      INPUT(IS_BIG01, 6.) AS IS_BIG02  
FROM LM_POS01.SPTMP012;  
QUIT;
```

## DATE Type Handling

- 날짜는 SAS에서 숫자 형식으로 관리하고 표시형식을 지정하여 나타냄

```
PROC SQL;
CREATE TABLE LM_POS01.SPTMP014 AS
SELECT *, INPUT(SUBSTR(STRIP('12/28/2014'),1,10),MMDDYY10.) AS
    CHECK_DAY,
    INPUT(SUBSTR(STRIP('12/28/2014'),1,10),MMDDYY10.) AS
    CHECK_DAY01  FORMAT MMDDYY10.
FROM LM_POS01.SPTMP013; QUIT;
```

```
PROC SQL;
CREATE TABLE LM_POS01.SPTMP015 AS
SELECT *, CHECK_DAY-20085 AS CHECKDAY00 ,
    CHECK_DAY-20085 AS CHECKDAY001  FORMAT MMDDYY10.
FROM LM_POS01.SPTMP014 ; QUIT;
```

## DATE 표시형식 변경

- MMDDYY, YYMMDD 등 표시 형식을 출력시 지정, 변경 가능
  - 10자리가 되는 이유는 구분자도 계산하기 때문

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP016 AS  
SELECT *, CHECK_DAY AS CHECK_DAY02   FORMAT YYMMDD10.  
FROM LM_POS01.SPTMP014;  
QUIT;
```

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP016 AS  
SELECT *, CHECK_DAY AS CHECK_DAY02   FORMAT YYMMDD10. ,  
        CHECK_DAY AS CHECK_DAY03   FORMAT YYMMDD8.  
FROM LM_POS01.SPTMP014;  
QUIT;
```

## DATETIME과 DATE의 구분 활용

- DATETIME 포맷을 적용하고 있으나 사실상 활용은 하지 않고 있음
- 불필요한 시간 부분을 제외하기 위해 DATEPART() 함수 적용

### PROC SQL;

```
CREATE TABLE LM_POS01.SPTMP042 AS  
SELECT DISTINCT STR_CD, DATEPART(SALE_DY) FORMAT YYMMDD10.  
    AS SALE_DYA,  
    SUM(SALE_SALE_AMT) AS SSAMT  
FROM LM_POS01.SALE_HEAD  
GROUP BY STR_CD, SALE_DY;  
QUIT;
```

## ROUND와 숫자 자리수 포맷 표시

- 숫자의 경우, 반올림, 올림, 내림 등 소수점 표현 처리 필요
  - 반올림 등의 자리수 기준 지정 및 표시 소수점 자릿수 지정에 옵션 명시 필요

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP033 AS  
SELECT DISTINCT PROD_NM, STR_NM, ROUND(AVG(SALE_SALE_AMT) ,  
    0.001) FORMAT=7.2 AS AVG_SSAMTR  
FROM LM_POS01.SPTMP031  
WHERE STR_NM IN ('서현점' '부평역점' )  
GROUP BY STR_NM, PROD_NM  
ORDER BY PROD_NM;  
QUIT;
```



## NESTED / SUB QUERY 활용

- 하나의 SQL 질의 안에 일부로 다른 SQL 질의가 포함되는 형태
- JOIN을 명시적으로 사용하지 않고도 필요한 결과를 한번에 얻을 수 있음

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP021 AS  
SELECT DEPT_CD, DEPT_NM, SSAMT  
FROM (SELECT * FROM LM_POS01.SPTMP007  
      WHERE CCNT > 30) ; QUIT;
```

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP022 AS  
SELECT DEPT_CD, DEPT_NM, SSAMT, CCNT  
FROM LM_POS01.SPTMP007  
WHERE DEPT_CD IN (SELECT DEPT_CD FROM LM_POS01.SPTMP007  
                  WHERE CCNT > 50 AND SSAMT>2000000) ; QUIT;
```

## 여러 테이블을 한번에 JOIN

- 기준이 되는 테이블을 지정하고 그와 결합될 두 개 이상 테이블을 JOIN

```
PROC SQL;
CREATE TABLE LM_POS01.SPTMP031 AS
SELECT A.STR_CD, B.STR_NM, A.SALE_DY, A.PROD_CD, C.PROD_NM,
       A.SALE_SALE_AMT
FROM LM_POS01.SALE_PROD AS A
     LEFT JOIN LM_POS01.FULL_STRCD AS B ON A.STR_CD=B.STR_CD
     LEFT JOIN LM_POS01.PRODUCT AS C ON A.PROD_CD=C.PROD_CD ;
QUIT;
```

```
PROC SQL;
CREATE TABLE LM_POS01.SPTMP032 AS
SELECT DISTINCT STR_NM, PROD_NM, SUM(SALE_SALE_AMT) AS SSAMT
FROM LM_POS01.SPTMP031 WHERE STR_NM LIKE '%VIC%'
GROUP BY STR_NM, PROD_NM; QUIT;
```

문자중 일부만 알고 있을 때 LIKE 문을 사용해 검색 - 단, 속도문제!

## SQL 내에서 사용 가능한 함수 활용

- 기준이 되는 테이블을 지정하고 그와 결합될 두 개 이상 테이블을 JOIN

- RANUNI
- MONOTONIC
- UNIQUE
- NMISS

## RANUNI 단순 랜덤 샘플링

- 0~1 사이의 난수를 발생시키는 함수
- 난수를 이용한 무작위 샘플링에 활용 가능 (예: 전체 고객중 무작위 5%)

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP051 AS  
SELECT DISTINCT CUST_NO, RANUNI(1234)*100 AS RNDM_ID  
FROM LM_POS01.SALE_HEAD;  
QUIT;
```

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP052 AS  
SELECT MAX(RNDM_ID) AS MAX_RNDM_ID , AVG(RNDM_ID) AS  
      AVG_RNDM_ID ,  
      MIN(RNDM_ID) AS MIN_RNDM_ID  
FROM LM_POS01.SPTMP051; QUIT;
```

# RANUNI SHUFFLING

- RANUNI 문은 ORDER BY 절에서 정렬순서를 무작위로 바꾸는데도 활용 가능 (SHUFFLING)
  - 괄호안의 숫자는 Seed
- OUTOBS와 함께 사용하면 랜덤 샘플링의 기능을 사용 가능

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP053 AS  
SELECT STR_CD, SUM(SALE_SALE_AMT) AS SSAMT  
FROM LM_POS01.SALE_HEAD  
GROUP BY STR_CD  
ORDER BY RANUNI(1234);  
QUIT;
```

## MONOTONIC 지정한 순번의 레코드 추출

- MONOTONIC 함수는 특정한 번호에 해당하는 순번의 레코드 조회
- 정렬이 되어 있는 테이블에서 앞에서부터(위에서부터) 10~20% 구간내의 행을 가져오는 식의 요건 처리에 활용 가능

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP055 AS  
SELECT  CUST_NO  
FROM LM_POS01.SALE_HEAD  
WHERE MONOTONIC() BETWEEN 1 AND 10 ;  
QUIT;
```

## UNIQUE 와 DISTINCT 차이

- DISTINCT와 동일한 기능을 하는 UNIQUE
  - 단, 표준 SQL의 DISTINCT와는 달리 UNIQUE는 함수 형태
- SAS PROC SQL에서는 함수로도 DISTINCT() 사용 가능

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP061 AS  
SELECT  UNIQUE(PROD_NM) AS PROD_NM  
FROM LM_POS01.PRODUCT ; QUIT;
```

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP062 AS  
SELECT  DISTINCT(PROD_NM) AS PROD_NM  
FROM LM_POS01.PRODUCT; QUIT;
```

## UNIQUE 와 DISTINCT 활용

- COUNT() 함수를 활용하면 한번에 테이블내의 복수 필드에 대한 값의 카테고리 수 통계 산출 가능

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP063 AS  
SELECT  COUNT(UNIQUE(PROD_NM)) AS CNT_PROD_NM,  
        COUNT(UNIQUE(PROD_CD)) AS CNT_PROD_CD,  
        COUNT(UNIQUE(DEPT_CD)) AS CNT_DEPT_CD  
FROM LM_POS01.PRODUCT ; QUIT;
```

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP064 AS  
SELECT  COUNT(DISTINCT PROD_NM) AS CNT_PROD_NM,  
        COUNT(DISTINCT PROD_CD) AS CNT_PROD_CD,  
        COUNT(DISTINCT DEPT_CD) AS CNT_DEPT_CD  
FROM LM_POS01.PRODUCT ; QUIT;
```



## NMISS – MISSING RECORD 추출

- COUNT는 값이 있는 레코드 건수를 반환
  - \* 또는 특정 필드를 지정
- NMISS는 해당 필드가 MISSING인가를 반환
  - MISSING인 레코드의 숫자를 세기 위해 사용

```
PROC SQL;  
CREATE TABLE LM_POS01.SPTMP071 AS  
SELECT COUNT(*) AS CNT_REC,  
       COUNT(CUST_NO) AS CNT_CUST_NO,  
       NMISS(CUST_NO) AS CNT_CUST_NO_MSSNG,  
       COUNT(STR_CD) AS CNT_STR_CD,  
       NMISS(STR_CD) AS CNT_STR_CD_MSSNG  
FROM LM_POS01.SALE_HEAD ;  
QUIT;
```

# PROC SQL and SAS Macro language

- SAS Macro를 SQL 구문내에서도 사용할 수 있도록 허용
  - 반복된 처리를 프로그램화 하는 용도로 사용

```
PROC SQL NOPRINT;  
SELECT DISTINCT PATIENT INTO :PAT1- :PAT999  
FROM VITALS  
ORDER BY PATIENT;  
QUIT;
```

```
PROC SQL NOPRINT;  
SELECT DISTINCT PATIENT INTO :PATLIST  
SEPARATED BY ','  
FROM VITALS  
ORDER BY PATIENT;  
QUIT;
```

# Data Table Management: INSERT와 DELETE

```
PROC SQL NOPRINT;  
INSERT INTO VITALS  
VALUES(102 '20AUG2001'd 75 98.4 122 90);  
INSERT INTO VITALS  
SET PATIENT=102, DATE='27AUG2001'd,  
PULSE=77, TEMP=98.8, BPS=129, BPD=88;  
QUIT;
```

```
PROC SQL NOPRINT;  
DELETE FROM VITALS  
WHERE PATIENT = 101;  
QUIT;
```

# Data Table Management:

## ALTER TABLE & DROP TABLE

- DROP TABLE : 테이블의 삭제
- ALTER TABLE : 테이블의 구조 변경

```
PROC SQL NOPRINT;  
ALTER TABLE VITALS  
MODIFY DATE FORMAT=MMDDYY8.  
DROP TEMP;  
QUIT;
```

```
PROC SQL NOPRINT;  
DROP TABLE BP;  
QUIT;
```

## PROC SQL advantages over DATA step

- 인덱스 관리가 개선되면서 오히려 DATA STEP 보다 빠른 처리 가능해짐
- 테이블간의 JOIN 연결이 간편함 (간단한 스크립트로 처리)
- SQL안에 SQL을 넣는 Nest 방식 활용 가능
- 외부 데이터베이스와의 연결이 용이
- **DATA STEP은 한 테이블에 행이 많은 경우에 적합**

## [연습문제답] CUST\_SEG EXAMPLE

(연습용 가상 서점 고객 데이터베이스 사용)

**PROC SQL;**

```
CREATE TABLE LM_POS01.CUST_SEG03 AS  
SELECT SEX,  
       CASE WHEN AGE < 30 THEN 1  
            WHEN AGE >=30 AND AGE <45 THEN 2  
            ELSE 3 END  
       AS AGE_GRP,  
       AVG(TOT_AMT) AS AVG_TOT_AMT,  
       COUNT(*) AS CNT_CUST  
FROM LM_POS01.CUST_SEG01  
GROUP BY SEX, AGE_GRP ;  
QUIT;
```

## 연습문제 :: ADDITIONAL

- **[연습1]** 취소건수가 가장 많은 상위 50명의 고객 중 7명의 고객을 무작위로 추출하여 고객번호를 하나의 테이블로 생성하라
- **[연습2]** 신선부문의 상품을 구매한 영수증에서 고객번호가 MISSING인 영수증 건수를 구하라
- **[연습3]** 판매액 합계가 가장 큰 두 개의 점포의 점포명과 해당 점포의 고객들의 구매금액 최대, 최소 값과 포인트 합계를 하나의 테이블에 표시하라
- **[연습4]** 이용고객수가 가장 많은 점포 3개의 취소율(총영수증 건수 중 취소가 포함된 영수증 건수 비율)을 구하라
- **[연습5]** 구매가 가장 많은 5개 상품군에 대하여 각각 별로 가장 취소율이 낮은 단품의 목록을 각각 2개씩 추출하여, 상품군명과 단품명과 함께 표시하라
- **[연습6]** (AAA02 data set 을 사용하여 ) 일자별, 점포별 최대 구매액을 보였던 고객의 리스트를 추출하라
- **[연습7]** 일자별로 가장 많은 고객이 이용한 점포명과 이용 고객수, 그 점포의 다음날 이용고객수를 산출하라

# DAY 2

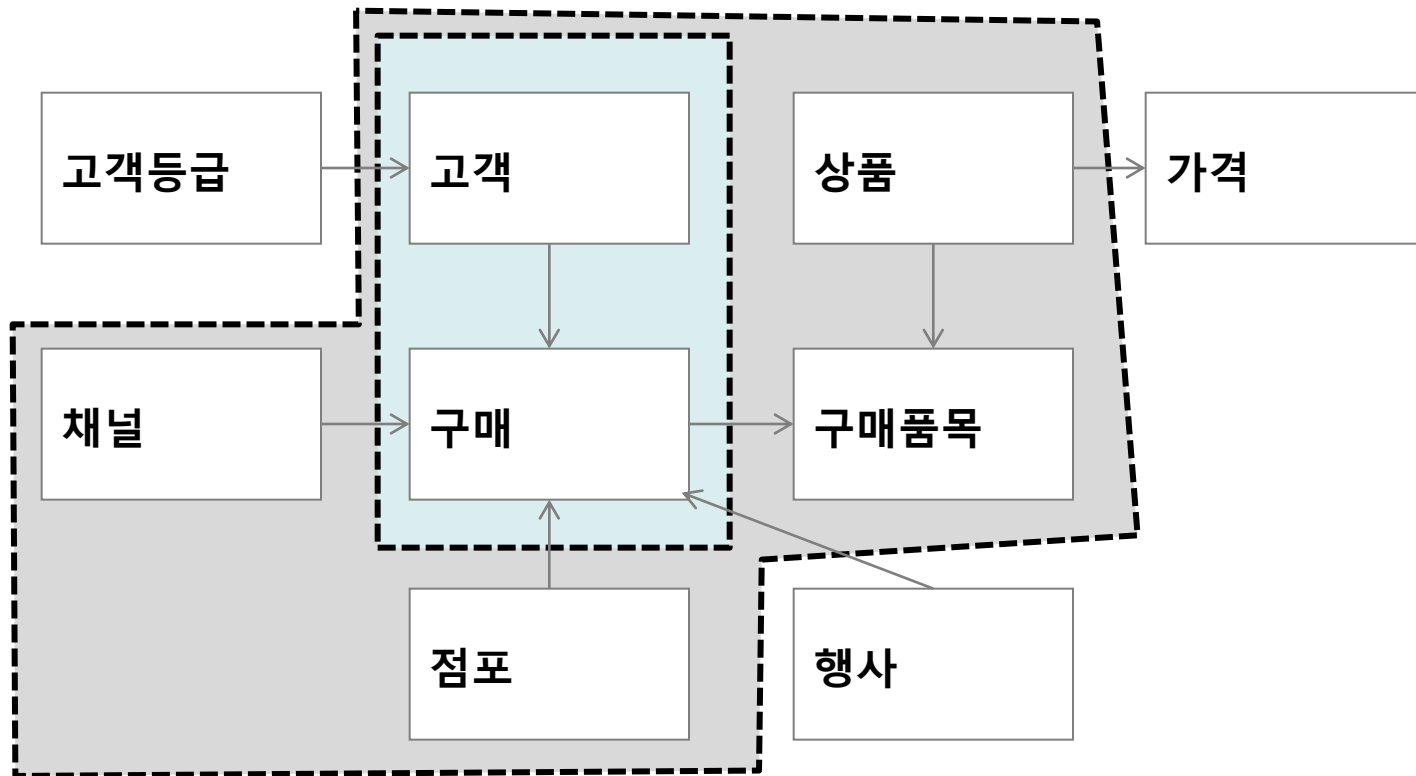
2B ::

## 유통업에서의 고객데이터 분석 유형



# 유통업 고객 데이터 분석 기본 패턴

- 1차 :: 고객별 구매액, 구매건수 집계
- 2차 :: 고객별 구매패턴 전반을 영수증 전체 활용해서 분석
- 3차 :: 고객등급, 채널, 점포, 상품, 행사 등 각각을 중심으로 분석



## 유통업 고객 데이터 분석의 예

- **1차 :: 고객별 구매액, 구매건수 집계**
  - 월별 고객별 구매액
  - 연간 전체 고객 평균 구매일수
- **2차 :: 고객별 구매패턴 전반을 영수증 전체 활용해서 분석**
  - 최근 1년간 고객별 점포별 구매일수
  - 최근 3개월간 행사상품 구매 구매건수 및 행사상품 구매금액 비중
  - 최근 3개월 십분위별(Decile) 상품군별 객단가와 구매일수
- **3차 :: 고객등급, 채널, 점포, 상품, 행사 등 각각을 중심을 기준으로 분석**
  - 고객등급별 구매일수 평균
  - 점포별 우수고객 라면류 구매금액 비중
  - 캠페인 반응 고객 행사기간 중 인당구매금액 평균
  - DM수신거부 고객 최종구매후 경과기간 평균

# 상품, 점포, 고객등급, 채널별 등 기본 분석 패턴

- 상품별 분석
  - 대분류 상품군별 구매금액 합계
  - 중분류 상품군별 구매고객수
- 점포별 분석
  - 최근 3개월 월별 수도권 점포 매출비중
  - 점포별 고객 구매주기
- 고객등급 분석
  - 전년대비 우수고객 등급 이탈 비율
  - 고객등급별 주구매 상품군
- 채널 분석
  - 복수채널 이용고객 비율
  - 온라인 회원 중 오프라인 점포 이용 고객 비율



# Thank You!

## contact:

> RE::VISION



전 용 준

대표/컨설턴트 | 리비전컨설팅

[xyxonyxon@empal.com](mailto:xyxonyxon@empal.com) 010.3095.1451

Keyword: 예측모델링 | 데이터 마이닝 | 빅 데이터

<http://www.revisioncon.co.kr>