آشنایی با نرم‌افزار SAS

حسن رنجی

پاییز ۱۳۹۱

مقدمه

SAS یک بسته‌ی نرم‌افزاری برای انجام تحلیل‌های آماری است. SAS در اوایل دهه‌ی ۱۹۷۰ ساخته شد، هدف آغازین آن مدیریت و تحلیل داده‌های کشاورزی بوده است ولی در حال حاضر یکی از پر استفاده‌ترین نرم‌افزارهای آماری است. واژه‌ی SAS در ابتدا سرنام کلمات Statistical Analysis System بود ولی دیگر چنین نیست. سامانه‌ی SAS دارای مجموعه‌ای از شیوه‌ها برای تحلیل داده‌ها و توانایی کار با داده‌های ورودی/خروجی است. این سامانه به کاربر امکان می‌دهد تا برای تحلیل‌های آماری عبارت‌های برنامه‌ای خود را بنویسد، همچنین روال‌های SAS را که شیوه‌ها نام دارند، فراخوانی کند. عبارت‌های برنامه‌ای معمولاً برای انجام تغییر در داده‌ها مثل تبدیل مقدارهای متغیرهای موجود، ایجاد متغیرهای جدید از متغیرهای موجود یا ساختن زیرمجموعه‌ای از مشاهده‌ها یا متغیرها استفاده می‌شود. داده‌های آماده‌سازی شده به عنوان ورودی برای تحلیل‌های آماری مورد نظر کاربر به کار می‌رود.

SAS مجموعه‌ای از چند نرم‌افزار است که متداول‌ترین آن‌ها عبارت‌اند از:

- SAS/BASE: مدیریت داده‌ها، و شیوه‌های پایه‌ای
- SAS/STAT: تحلیل‌های آماری
- SAS/GRAPH: گرافیک‌های با کیفیت بالا
- SAS/OR: تحقیق در عملیات
- SAS/ETS: اقتصادسنجی و سری‌های زمانی
- SAS/IML: زبان ماتریسی محاوره‌ای
- SAS/QC: کنترل کیفیت

قاعده‌ها و شکل دستوری پایه‌ای SAS

مقدار داده‌ای

مقدارهای داده‌ای به نویسه‌ای و عددی رده‌بندی می‌شوند. مقدار نویسه‌ای حداکثر می‌تواند ۳۲۷۶۷ نویسه داشته باشد و می‌تواند شامل حرف انگلیسی، عدد، علائم خاص و فاصله باشد.

چند مثال از مقدارهای نویسه‌ای عبارت‌اند از:

y5, 789, southwest, do, a

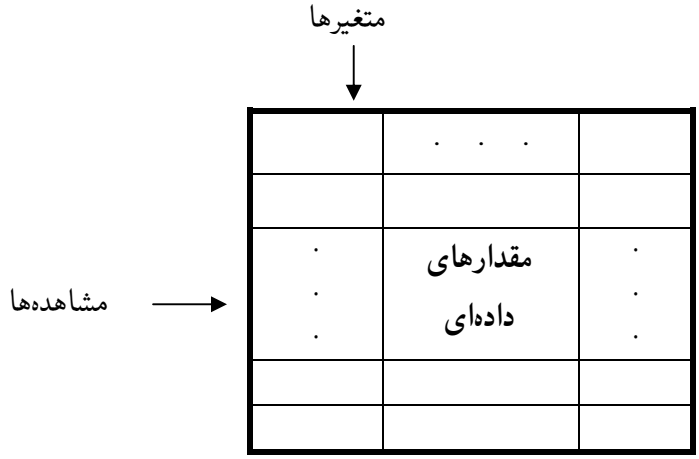
مقدار عددی، یک عدد با و یا بدون نقطه‌ی اعشاری است که قبل از آن می‌تواند علامت به‌علاوه یا منها قرار گیرد ولی نمی‌تواند شامل واوک (,) باشد. مثال‌هایی برای مقدار عددی عبارت‌اند از:

45, 0.038, -81.,241.25, 2.345E-4

داده‌هایی که در یکی از این دو رده قرار نمی‌گیرند (مثل تاریخ با خط کج (/) یا عدد شامل واوک) با استفاده از اینفورمت به صورت عددی یا نویسه‌ای در مجموعه‌ی داده‌های SAS دخیره می‌شوند. در صورتی که نوع مقدار داده‌ای مشخص نشوند، پیش فرض SAS عددی بودن آن‌هاست.

مجموعه‌ی داده‌های SAS

مجموعه‌ی داده‌های SAS به شکل ماتریسی هستند که سطرهای آن مشاهده‌ها و ستون‌ها، مقدار متغیرها هستند. در شکل ۱-۲ مقدارهای داده‌ای در ستون‌ها نمایانگر متغیرها و در سطرها نمایشگر مشاهده‌ها هستند.



نمایشی از ماتریس داده‌ها

متغیرها

هر ستون ماتریس داده‌ها نمایانگر مقدارهای یک متغیر SAS است. متغیرهای SAS بر دو نوع‌اند:عددی و نویسه‌ای. مقدار متغیر نویسه‌ای می‌تواند شامل عدد باشد ولی با آن‌ها مثل نویسه رفتار می‌شود.

متغیرهای SAS علاوه بر ویژگی‌های نام و نوع دارای ویژگی‌های طول (بر حسب بایت) و موقعیت در مجموعه‌ی داده‌ها، اینفورمت، فورمت و برچسب هستند. ویژگی‌های متغیرهای SAS مثل مقدار آن‌ها در مجموعه‌ی داده‌های SAS ذخیره می‌شوند.

مشاهده‌ها

یک مشاهده گروهی از مقدارهای داده‌ای است که نمایانگر اندازه‌های مختلف مربوط به یک فرد (عنصر) است. منظور از فرد یک واحد آماری است که می‌تواند شخص، حیوان آزمایشگاهی، کرت و ... باشد. در مجموعه‌ی داده‌های SAS هر سطر نمایانگر یک مشاهده است.

نام‌ها

برای تحلیل آماری داده‌ها باید به متغیرها و مجموعه‌ی داده‌ها نامی اختصاص داده شود. نام‌ها باید با یک حرف یا خط زیرین (_) شروع شوند و می‌توانند شامل حروف، اعداد یا خط زیرین (_) باشند. طول نام‌ها نمی‌تواند بیش از ۳۲ نویسه باشد، نام‌ها نمی‌توانند شامل فاصله یا نویسه‌های ویژه مثل واوک، نقطه واوک (;) و علامت‌های $ و # باشند. نام‌های زیر مثال‌هایی از نام‌ها در سامانه‌ی SAS هستند:

gender, exam1, rep_nu, y123,_test

نکته: برای این‌که بتوانید از فاصله و نویسه‌های ویژه در نام‌ها استفاده کنید عبارت زیر را در برنامه‌ی SAS به کار ببرید:

```
Options validvarname=any;
```

سپس برای مشخص‌کردن این‌گونه نام‌ها از قالب زیر استفاده کنید:

'variable name'N

هر عبارت SAS با یک واژه‌ی کلیدی شروع و به علامت ; ختم می‌شود. برنامه‌ها را می‌توان با حروف بزرگ یا کوچک نوشت. هر عبارت را می‌توان در چند خط نوشت، در یک خط می‌توان بیش از یک عبارت نوشت. عبارت‌ها را می‌توان از هر ستونی در هر خط نوشت. علامت ; به تنهایی یک عبارت است که به آن عبارت پوچ گفته می‌شود. عبارت توضیح، عبارتی است که با * شروع می‌شود، علاوه بر این، برای نوشتن توضیح در برنامه می‌توان توضیح مورد نظر را بین دو علامت */ و /* نوشت.

فهرست متغیرهای SAS

یک فهرست متغیرهای SAS شامل نام متغیرها است که حدّ اقل با یک فاصله از هم جدا شده باشد. مثلاً fer rep_no yield، فهرستی ساده از نام متغیرها است. به هر زیر مجموعه از متغیرها که در مجموعه‌ی داده‌های SAS وجود دارند فارغ از اینکه عددی هستند یا نویسه‌ای، می‌توان با اتصال نام اول و نام آخر آن‌ها با دو خط (--) ارجاع داد. برای مثال می‌توان متغیرهای W، X، Y، و Z به‌صورت W--Z ارجاع داد. ارجاع به متغیرهای دنباله‌ای مثل X1، X2، X3، X4، و X5 با یک خط و به صورت X1-X5 انجام می‌شود.

برای مشخص‌کردن متغیرهایی که با یک واژه‌ی خاص شروع می‌شوند از شکل :word استفاده کنید. برای مثال، :zar مشخص کننده‌ی فهرست متغیرهایی است که نام آن‌ها با zar شروع می‌شود.
به‌علاوه برای ارجاع به همه‌ی متغیرهای یک مجموعه‌ی داده‌ها از عبارت _ALL_، برای متغیرهای عددی از _NUMERIC_ و برای متغیرهای نویسه‌ای از _CHAR_ استفاده کنید.

گام‌های برنامه‌ای SAS

گام DATA

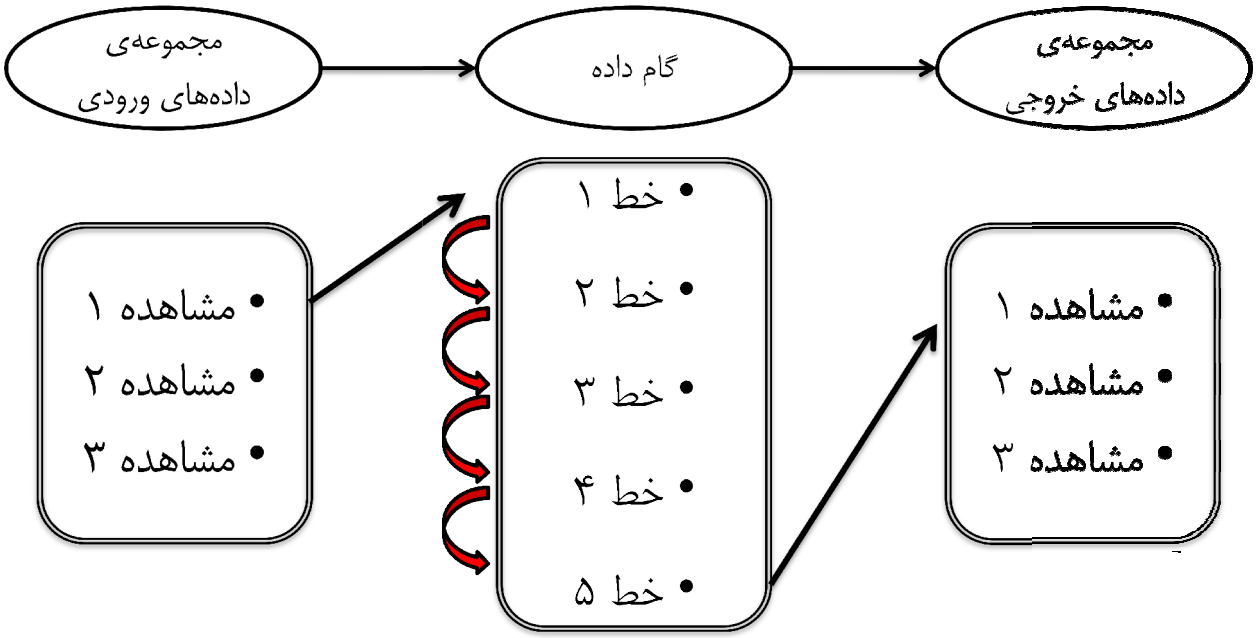
در گام DATA مجموعه‌ی داده‌های SASی که مناسب تحلیل‌های آماری است و گام PROC از آن استفاده می‌کند، به صورت زیر ایجاد می‌شود.

۱- برای شروع گام DATA از عبارت DATA و نامی برای آن استفاده می‌شود. نام‌گذاری مجموعه‌ی داده‌ها اختیاری است.
۲- از یکی از عبارت‌های INPUT، SET، MERGE، یا UPDATE برای مشخص‌سازی مکان و اطلاعاتی که در مجموعه‌ی داده‌ها است، استفاده می‌شود.

گام DATA شامل گروهی از عبارت‌ها است که با عبارت زیر شروع می‌شوند.

```
DATA <نام‌ها>;
```

گام DATA دارای یک حلقه‌ی داخلی است و به صورت خط به خط و مشاهده به مشاهده اجرا می‌شود:



گام PROC

تحلیل آماری مجموعه‌ی داده‌های SAS با شیوه‌های موجود در SAS صورت می‌گیرد. این شیوه‌ها با عبارت PROC که کوته‌نوشته‌ی Procedure است فراخوانی و اجرا می‌شود. گام PROC شامل گروهی از عبارت‌ها است که با عبارت زیر شروع می‌شوند:

```
PROC نام شیوه;
```

استفاده از گزینه‌های مجموعه‌ی داده‌ها

از گزینه‌های مجموعه‌ی داده‌ها می‌توانید برای پردازش داده‌ها استفاده کنید. این گزینه‌ها بین دو علامت پرانتز، بلافاصله بعد از نام مجموعه‌ی داده‌ها قرار می‌گیرند. این گزینه‌ها را در گام DATA و گام PROC می‌توان مورد استفاده قرار داد. برخی از این گزینه‌ها در جدول زیر فهرست شده است. شما می‌توانید یک یا چند گزینه را برای رسیدن به اثرهای ترکیبی مورد استفاده قرار دهید.

گزینه‌ی مجموعه‌ی داده	مثال	شرح
(DROP =)	(DROP =A B)	متغیرهای A و B را از مجموعه‌ی داده‌ها حذف می‌کند.
(FIRSTOBS =)	(FIRSTOBS = 5)	خواندن داده‌ها را از مشاهده‌ی مشخص‌شده (۵) شروع می‌کند.
(IN =)	(IN = DS1)	متغیر موقت DS1 را برای مشخص کردن مجموعه‌ی داده‌ی منبع ایجاد می‌کند. مقدار این متغیر در صورت وجود مشاهده در مجموعه‌ی داده‌ی متناظر، ۱ و در غیر این صورت ۰ است.
(KEEP =)	(KEEP = A B)	متغیرهای A و B را حفظ می‌کند.
(OBS =)	(OBS = 10)	خواندن داده‌ها را بعد از رسیدن به مشاهده‌ی مشخص‌شده (۱۰) متوقف می‌کند.
(RENAME = ())	(RENAME = (A = D))	نام متغیر A را به D تغییر می‌دهد. توجه داشته باشید که سایر گزینه‌ها در صورت استفاده به همراه این گزینه، باید از نام‌های اولیه‌ی متغیرها، و نه نام‌های جدید، استفاده کنند.
(WHERE = ())	(WHERE = (A >= 10))	فقط مشاهداتی را در نظر می‌گیرد که مقدار متغیر A در آن‌ها بزرگ‌تر یا مساوی ۲۰ باشد.

نکته: توجه داشته باشید که این گزینه‌ها مجموعه‌ی داده‌های اصلی را تغییر نمی‌دهند.

انتخاب زیرمجموعه‌ای از داده‌ها

از عبارت‌های IF و WHERE می‌توانید برای ساختن زیرمجموعه‌ای از مجموعه‌ی داده‌های موجود استفاده کنید. تفاوت‌هایی بین این دو عبارت بر حسب این که چه‌موقع و با چه گزینه‌هایی می‌توانند مورد استفاده قرار گیرند، وجود دارد. شرط‌های WHERE پیش از این که داده‌ها وارد بافر ورودی شوند اِعمال می‌شوند در حالی که شرط‌های IF پس از ورود داده‌ها به بردار داده‌های برنامه (PDV) اِعمال می‌شوند. برای مجموعه‌های داده‌های بزرگ، عبارت WHERE کاراتر است، زیرا پیش از آن که داده‌ها وارد PDV شوند، اجرا می‌شود. در دو حالت نمی‌توان از WHERE استفاده کرد. عبارت WHERE تنها با متغیرهای موجود در مجموعه‌ی داده‌های ورودی می‌توان مورد استفاده قرار داد. اگر شرط‌های WHERE شامل متغیرهایی که در گام DATA ایجاد شده‌اند یا متغیرهای موقتی مانند _N_ باشند، با یک پیغام خطا مواجه خواهید شد. به‌علاوه، WHERE را نمی‌توان با عبارت INPUT یا گزینه‌های =POINT و =FIRSTOBS به کار برد. برای استفاده از عبارت WHERE، یک عبارت SET، MERGE یا UPDATE باید در گام DATA موجود باشد. همچنین، توجه داشته باشید که هنگام ترکیب داده‌ها، شرط WHERE پیش از ترکیب مجموعه‌های داده‌ها اِعمال می‌شود بر خلاف شرط IF که پس از ترکیب مجموعه‌های داده‌ها اِعمال می‌شود.

هیچ تفاوتی از نظر کارایی بین گزینه‌ی WHERE و عبارت WHERE وجود ندارد. عمل‌گرهای بسیاری در SAS وجود دارد که می‌توانند با هر دو عبارت IF و WHERE استفاده شوند. چند عمل‌گر وجود دارد که فقط می‌توانند با عبارت WHERE مورد استفاده قرار گیرند. این عمل‌گرها عبارت‌اند از: BETWEEN-AND، اصلاح‌گر دونقطه (:)، CONTAINS یا ?، IS NULL یا IS MISSING، LIKE ‘pattern’، علامت %، علامت *=، و زیرخط (_).

مثال

```
DATA test;
   INPUT name $ class $ score;
   CARDS;
Tim   Math     9
Tim   History  8
Tim   Science  7
Sally Math    10
Sally Science  7
Sally History 10
John  Math     8
John  History  8
John  Science  9
RUN;
DATA grade;
   SET test;
   WHERE name = 'Tim' or name = 'Sally';
   IF class = 'Math' THEN classnum = 1;
   ELSE IF class = 'Science' THEN classnum = 2;
   ELSE IF class = 'History' THEN classnum = 3;
   IF classnum = 2;
```

```
RUN;
PROC PRINT DATA = grade;
RUN;
```

خروجی این برنامه را در زیر مشاهده می کنید:

Obs	name	class	score	classnum
1	Tim	Science	7	2
2	Sally	Science	7	2

۳-۱-۵- اجراهای شرطی

عبارت‌های IF-THEN و ELSE را می‌توان برای اجرای عبارت‌های برنامه‌ای SAS بسته به مقدار عبارت، به کار برد. شکل دستوری این عبارت‌ها به صورت زیر است:

```
IF جمله THEN عبارت;
<ELSE عبارت>;
```

در شکل دستوری بالا، نتیجه‌ی یک عبارت مقایسه‌ای مقدار ۱ یا صفر می‌گیرد. مقدار ۱ نشانگردرست بودن مقایسه است و مقدار صفر نشانگر نادرست بودن آن. یک عبارت مقایسه‌ای شامل مقایسه‌های عددی و نویسه‌ای با عمل‌گرهای مقایسه‌ای مثل AND (&) یا OR(|) است. در شکل دستوری بالا، می‌تواند هر عبارت اجرایی SASباشد.

مثال ۱-۴

```
IF exp<.8 THEN PUT id "Below Poverty Line";
ELSE PUT id "Above Poverty Line";
```

در مثال بالا، نتیجه‌ی جمله‌ی exp < .8 یک است اگر مقدار متغیر expی مشاهده‌ی جاری کمتر از ۰/۸۰ باشد. در این‌صورت عبارت "Below Poverty Line" در مقابل idی جاری نوشته می‌شود، در غیر این‌صورت مقدار جمله صفر است و عبارت "Above Poverty Line"در مقابل متغیر id نوشته می‌شود.

مثال

```
IF ostan='TEH' | ostan='QUM' THEN region='MARKAZI';
```

عبارت بالا یک عبارت منطقی است که مقدار آن ۱ است اگر مقدار متغیر ostan برای مشاهده‌ی جاری "TEH" یا "QUM" باشد، و در غیر این‌صورت صفر است. بنا بر این مقدار متغیر region مشاهده‌ی جاری رشته نویسه‌ای"MARKAZI"را می‌گیرد اگر مقدار متغیر ostan مشاهده‌ی جاری "TEH" یا "QUM" باشد. در غیر این‌صورت مقدار region یا خالی می‌ماند و یا با عبارت IF-THEN دیگر مشخص می‌شود.

مثال: گاهی اوقات لازم است مشروط به مقدار جمله‌ای چند عبارت برنامه‌ای SAS اجرا نشود. عبارت‌های زیر نحوه‌ی جلوگیری از اجرای برخی عبارت‌های برنامه‌های SAS را نشان می‌دهد.

```
IF 0.0 <= exp <=0.8 THEN GO TO poverty;
```

عبارت‌های برنامه‌ای SAS

```
poverty: inc=(person*min)+2;
```

عبارت‌های برنامه‌ای SAS

در عبارت‌های بالا کلمه‌ی poverty یک برچسب است. در صورتی‌که عبارت 0.8=> exp =>0.0 درست باشد، SAS از اجرای "عبارت‌های برنامه‌ای SAS" اول جلوگیری کرده، عبارت ;inc=(person*min)+2 را اجرا می‌کند.

از گروه DO-END نیز برای اجرای چند عبارت SAS می‌توان استفاده کرد. مثال زیر نحوه‌ی استفاده از DO-END را نشان می‌دهد.

مثال

```
IF inc<0.8 THEN DO;
   n = n+1;
   pexp = exp/n;
   weight = .4;
END;
ELSE weight = .2;
```

مثال: این مثال نحوه‌ی ساختن یک متغیر جدید با استفاده از عبارت‌های IF-THEN/ELSE را نشان می‌دهد:

```
DATA group1;
INPUT id score @@;
DATALINES;
01 18 02 14 03 09 04 13 05 17 06 12 07 19 08 11 09 10 10 10
;
RUN;

DATA group2;
SET group1;
IF  0<= score <10 THEN scoregroup ='F';
IF 10<= score <12 THEN scoregroup ='D';
IF 12<= score <14 THEN scoregroup ='C';
IF 14<= score <17 THEN scoregroup ='B';
IF      score >=17 THEN scoregroup ='A';
RUN;

PROC PRINT DATA=group2;
RUN;
```

در مثال بالا نماد "@@" اجازه می‌دهد که در هر خط داده‌ای مقدارهای بیش از یک مشاهده وارد شود. در غیر این‌صورت در هر سطر داده‌ای باید یک مشاهده وارد شود. عبارت SET از عبارت‌هایی است که در گام DATA به کار می‌رود و به SAS می‌گوید که داده‌های مجموعه‌ی داده‌هایی که نامش بعد از SET می‌آید را در مجموعه‌ی داده‌هایی با نامی که بعد از عبارت DATA می‌آید با اعمال عبارت‌های شرطی که به دنبال عبارت SET می‌آیند، بریزد.

خروجی مثال بالا در زیر نمایش داده شده است.

Obs	id	score	scoregroup
1	1	18	A
2	2	14	B
3	3	09F	
4	4	13	C
5	5	17	A
6	6	12	C
7	7	19	A
8	8	11	D
9	9	10	D
10	10	10	D

عمل‌گر IN

برای این‌که مقدار یک متغیر را با فهرستی از مقادیر مقایسه کنید ناچارید از چندین مقایسه به همراه عمل‌گر منطقی OR استفاده کنید، دستور زیر را در نظر بگیرید:

```
IF a = 2 OR a = 5 OR a = 7 OR a = 9 THEN b = 1;
```

زمانی که تعداد مقادیر مورد مقایسه زیاد شود استفاده از شکل دستوری بالا منطقی به نظر نمی‌رسد، در این موارد می‌توانید از عملگر IN به‌صورت زیر استفاده کنید:

```
IF a IN(2, 5, 7, 9) THEN b = 1;
```

مقادیر موجود در فهرست مورد نظر را می‌توانید با یک فاصله‌ی خالی یا یک واوک از هم جدا کنید. برای مثال دستور زیر را ببینید:

```
IF Province IN('Tehran' 'Isfahan' 'Shiraz' 'Alborz') THEN ...
```

محاسبه‌های مکرر

انجام محاسبه‌های تکراری علاوه بر استفاده از عبارت DO-END می‌تواند با استفاده از عبارت‌های DOنیز انجام شوند. عبارت‌های DOی مکرر، DO WHILE و DO UNTIL به دلیل کاربردهای متنوع و این‌که می‌توان آن‌ها را ادغام کرد، انعطاف‌پذیری بیشتری دارند.

مثال: این مثال نمایشی از کاربست DOی تکراری و عبارت ARRAY را ارائه می‌دهد:

```
DATA exam;
   INPUT hw1-hw6 test1-test3;
   ARRAY exam(9) hw1-hw6 test1-test3;
   DO i=1 to 8;
```

```
      IF exam{i}=. THEN exam{i}=0;
   END;
   DATALINES;
خط‌های داده‌ای
;
RUN;
```

به طور کلی Doی مکرر برای انجام عملیات یکسان روی دنباله‌ای از متغیرها به کار می‌رود. معرفی دنباله‌ای از متغیرها با استفاده از عبارت ARRAY صورت می‌گیرد. این عبارت در گام DATA معمولاً بعد از معرفی متغیرها آورده می‌شود. عبارت ARRAY به کاربر اجازه می‌دهد که به متغیرهایی که متناظر با عنصرهای ARRAY هستند، ارجاع دهد.

در مثال بالا متغیرهای hw1-hw6 و test1-test3 به عنوان عنصرهای ARRAY با نام exam معرفی شده‌اند. هنگامی‌که شاخص از ۱ تا ۸ است، به این متغیرها در حلقه‌ی DO به ترتیب با exam{1} تا exam{8} ارجاع داده می‌شود. داخل حلقه‌ی DO از شاخصi به عنوان متغیر شمارش‌گر استفاده شده است.

در این مثال از DOی مکرر برای تبدیل داده‌های گم‌شده به صفر برای متغیرهای hw1-hw6 و test1-test3 در خط‌های داده‌ای و از عبارت ARRAY برای ارجاع به این متغیرها استفاده شده است.

شکل کلی عبارت ARRAY به صورت زیر است:

```
ARRAY نام آرایه {n} نام عنصرهای آرایه مقدارهای آغازین;
```

که n تعداد عنصرهای آرایه، «نام عنصرهای آرایه» فهرستی از نام متغیرها است، و «مقدارهای آغازین» مقدارهای اولیه‌ی عددی و یا نویسه‌ای مربوط به عنصرهای آرایه است.

مثال ۱-۱۰

```
DATA set1;
   INPUT d1-d7;
   ARRAY day{7} d1-d7;
   ARRAY hour{7} h1-h7;
   DO i=1 to 7;
      IF day{i}=999 THEN day{i}=. ;
hour{i}=day{i}*12;
   END;
DATALINES;
خط‌های داده‌ای
;
RUN;
```

در این مثال آرایه‌ی day مرتبط با متغیرهای d1 تا d7 و آرایه‌ی hour مرتبط با متغیرهای h1 تا h7 تعریف شده‌اند. در حلقه‌ی DO مقدار هر متغیر d1 تا d7 اگر مقدار ۹۹۹ داشته باشند به مقدار گم‌شده تبدیل می‌شوند. سپس مقدار جاری هر متغیر h1 تا h7 با مقدار d1 تا d7 ضرب در ۱۲ تنظیم می‌شود.

حلقه‌های تکرار DO WHILE و DO UNTIL

```
DO WHILE(condition);
   SAS-statements
END;
```

```
DO UNTIL(condition);
   SAS-statements
END;
```

حلقه‌ی DO WHILE عبارت‌های مربوط را تا وقتی که شرط مورد نظر برقرار است اجرا می‌کند. حلقه‌ی DO UNTIL عبارت‌های مربوط را تا برقرار شدن شرط مورد نظر اجرا می‌کند. به عبارت دیگر، شرط مورد نظر در DO WHILE شرط اجرای حلقه است، در حالی که این شرط در DO UNTIL شرط توقف حلقه است. در DO WHILE، شرط در ابتدای حلقه بررسی می‌شود، در حالی که در DO UNTIL، شرط در انتهای حلقه بررسی می‌شود. به عبارت دیگر، حلقه‌ی DO WHILE ممکن است هیچ‌گاه اجرا نشود، ولی حلقه‌ی DO UNTIL حدّ اقل یک بار اجرا می‌شود.

مثال: دو مجموعه‌ی دستورات زیر، خروجی یکسانی تولید می‌کنند:

```
DATA _NULL_;
   n = 1;
   DO WHILE(n < 5);
      PUT n=;
      n + 1;
   END;
RUN;
```

```
DATA _NULL_;
   n = 1;
   DO UNTIL(n >= 5);
      PUT n=;
      n + 1;
   END;
RUN;
```

۱-۵-۲ تابع‌های SAS

تابع‌های SAS محاسبات یا تبدیل‌هایی را روی مقادیری که به‌عنوان شناسه دریافت می‌کنند انجام می‌دهند. هر یک از این شناسه‌ها می‌تواند مقدار ثابت، نام متغیر یا یک عبارت محاسباتی باشد. برخی تابع‌های SAS در جدول زیر فهرست شده است:

تابع	شرح	مثال	نتیجه
تابع‌های نویسه‌ای			
CAT(arg1, arg2, ...)	دو یا چند رشته‌ی نویسه‌ای را به هم می‌چسباند.	a = ‘ cat ‘; b = ‘dog’; c = CAT(a, b);	c = ‘ cat dog’
CATS(arg1, arg2, ...)	دو یا چند رشته‌ی نویسه‌ای را، پس از حذف فاصله‌های خالی ابتدا و انتهای آن‌ها، به هم می‌چسباند.	a = ‘ cat ‘; b = ‘dog’; c = CATS(a, b);	c = ‘catdog’

نتیجه	مثال	شرح	تابع
c = ‘cat dog’	a = ‘ cat ‘; b = ‘dog’; c = CATX(‘ ’, a, b);	دو یا چند رشته‌ی نویسه‌ای را، پس از حذف فاصله‌های خالی ابتدا و انتهای آن‌ها و قرار دان جداگر بین آن‌ها، به هم می‌چسباند.	CATX(‘separator’, arg1, arg2, ...)
b = ‘cat&dog’ c = ‘cat dog)’	a = ‘cat & dog’; b = compress(a); c = compress(a, ‘&’);	فاصله‌ی خالی یا نویسه‌های مشخص‌شده را از یک رشته‌ی نویسه‌ای حذف می‌کند.	COMPRESS(arg, ‘chars’)
b = 8	a = ‘ My cat ’; b = LENGTH(a);	طول یک رشته‌ی نویسه‌ای را برمی‌گرداند. این تابع فاصله‌های خالی انتهای رشته را نمی‌شمارد و طول یک رشته‌ی پوچ را ۱ برمی‌گرداند.	LENGTH(arg)
b = ‘8863’	a = ‘(9821) 8863 0442)’; b = SUBSTR(a, 8, 4);	زیررشته‌ای از یک رشته را از مکان p و به طول n برمی‌گردند.	SUBSTR(arg, p, n)
b = ’19-05-85’	a = ‘19/05/85’; b = TRANSLATE(a, ‘-’, ‘/’);	نویسه‌های to را جایگزین نویسه‌های from موجود در arg می‌کند. طول to و from باید یکسان باشد.	TRANSLATE(arg, to1, from1, ...)
b = ‘Ms. Smith’	a = ‘Mrs. Smith’; b = TRANWRD(a, ‘Mrs.’, ‘Ms.’);	replacement را جایگزین target موجود در arg می‌کند.	TRANWRD(arg, target, replacement)
b = ‘MY CAT’	a = ‘My cat’; b = UPCASE(a);	تمام حروف لاتین موجود در arg را به شکل بزرگ تبدیل می‌کند.	UPCASE(arg)
تابع‌های عددی			
a = 2 b = -2	a = INT(2.3); b = INT(-2.3)	بخش صحیح arg را برمی‌گرداند.	INT(arg)
a = 3 b = -2	a = CEIL(2.3); b = CEIL(-2.3);	کوچک‌ترین عدد صحیح بزرگ‌تر یا مساوی arg را برمی‌گرداند.	CEIL(arg)
a = 2; b = -3;	a = FLOOR(2.3); b = FLOOR(-2.3);	بزرگ‌ترین عدد صحیح کوچک‌تر یا مساوی arg را برمی‌گرداند.	FLOOR(arg)

نتیجه	مثال	شرح	تابع
a = 4 B = 3.28	a = ROUND(3.7); b = ROUND(3.275, .01);	arg را گرد می‌کند	ROUND(arg, rounding-unit)
a = 1	a = MIN(2, 7, ., 1);	مینیمم مقادیر	MIN(arg1, arg2, ...)
a = 7	a = MAX(2, 7, ., 1);	ماکسیمم مقادیر	MAX(arg1, arg2, ...)
a = 3	a = N(2, 7, ., 1);	تعداد مقادیر غیر گم‌شده	N(arg1, arg2, ...)
a = 1	a = NMISS(2, 7, ., 1);	تعداد مقادیر گم‌شده	NMISS(arg1, arg2, ...)
a = 10	a = SUM(2, 7, ., 1);	مجموع مقادیر غیر گم‌شده	SUM(arg1, arg2, ...)
a = 3.33	a = MEAN(2, 7, ., 1);	میانگین مقادیر غیر گم‌شده	MEAN(arg1, arg2, ...)
تبدیل داده‌های نویسه‌ای به عددی و برعکس			
b = 32275	a = '$32,275'; b = INPUT(a, comma7.);	رشته‌ی arg را با توجه به اینفورمت مشخص‌شده به یک مقدار عددی تبدیل می‌کند.	INPUT(arg, informat)
b = '00275';	a = 275; b = PUT(a, z5.);	عدد arg را با توجه به فورمت مشخص‌شده به یک مقدار نویسه‌ای تبدیل می‌کند.	PUT(arg, format)

نکته: برای استفاده از این توابع روی فهرستی از متغیرها به‌صورت زیر عمل کنید:

```
a = SUM(of x1-x10);
b = MIN(of zar:);
```

عبارت RETAIN

به‌صورت پیش‌فرض در هر تکرار گام DATA، مقدار متغیرهای جدید ابتدا به مقدار گم‌شده تغییر می‌یابد، سپس محاسبات مربوط انجام می‌شود، این عمل باعث می‌شود که ما به مقدار پیشین این متغیرها (در تکرار قبلی گام DATA) دسترسی نداشته باشیم. برای حفظ مقدار این‌گونه متغیرها برای استفاده در تکرار بعدی گام DATA می‌توانید از عبارت RETAIN استفاده کنید:

```
RETAIN نام متغیر<مقدار آغازین>;
```

مثال: دستورات زیر میانگین مقادیر متغیر Score را محاسبه می‌کند:

```
DATA scores;
   INPUT Score @@;
   CARDS;
12 15 17 25 19
RUN;
DATA mscore;
```

```
   RETAIN n Sum (2 * 0);
   SET scores end = last;
   n = n + 1;
   Sum = Sum + score;
   IF last THEN DO;
      Average = Sum / n;
      OUTPUT;
   END;
   DROP score;
RUN;
PROC PRINT DATA = mscore;
RUN;
```

نکته: گزینه‌ی END = last، متغیری موقتی به نام last می‌سازد که مقدار آن برای آخرین مشاهده ۱ و برای سایر مشاهده‌ها ۰ است، از این گزینه می‌توان برای تشخص آخرین مشاهده استفاده کرد.

مرتب کردن داده‌ها

مرتب کردن مجموعه‌ی داده‌ها بر حسب مقدارهای یک یا چند متغیر و قبل از پردازش داده‌ها با شیوه‌ی SORT انجام می‌شود. عبارت BY عبارت اطلاعاتی شیوه‌ای است که با PROC SORT به کار می‌رود. این عبارت نحوه‌ی مرتب کردن مجموعه‌ی داده‌ها را به شیوه‌ی SORT می‌گوید. در استفاده از شیوه‌ی SORT، مشاهده‌ها ابتدا به صورت افزایشی بر حسب مقدارهای اولین متغیر در عبارت BY مرتب می‌شوند. سپس درون هر گروه (زیرمجموعه) مشاهده‌ها بر حسب مقدارهای دومین متغیر در عبارت BY به صورت افزایشی مرتب می‌شوند. همین قاعده برای متغیرهای بعدی عبارت BY اِعمال می‌شود.

برای متغیرهای عددی، مقدار متغیر برای مرتب کردن مجموعه‌ی داده‌ها استفاده می‌شود و مقدار گم‌شده کم‌ترین رتبه را می‌گیرد. متغیرهای نویسه‌ای به ترتیب حرف‌های الفبای انگلیسی- A کوچکترین و Z بزرگ‌ترین- مرتب می‌شوند. فاصله‌ی خالی کوچک‌ترین نویسه است، بنا بر این، 'South east' بزرگ‌تر از 'North' ولی کوچک‌تر از 'Southern' است.

مثال:

```
PROC SORT DATA= score;
BY gender inc;
RUN;

PROC PRINT;
BY gender inc;
RUN;
```

این مثال مجموعه‌ی داده‌های score را ابتدا بر حسب مقدارهای متغیر gender و سپس درون هر گروه بر حسب مقدار متغیر inc مرتب می‌کند. اگر مقدار متغیر gender مقدارهای ۱ و ۲ باشند، ابتدا مجموعه‌ی داده‌ها به دو گروه با مقدارهای ۱ و ۲ی متغیر gender تقسیم شده و سپس داخل هر گروه داده‌ها بر حسب مقدار متغیر inc مرتب شوند.

حذف مشاهده‌های تکراری

شیوه‌ی SORT دارای دو گزینه‌ی NODUPKEY و = DUPOUT است. گزینه‌ی اول مشاهده‌های تکراری را از مجموعه‌ی داده‌ی خروجی حذف می‌کند. گزینه‌ی دوم، به همراه گزینه‌ی اول، مشاهده‌های تکراری حذف‌شده را در یک مجموعه‌ی داده‌های دیگر ذخیره می‌کند.

استفاده از SET به همراه BY

دو مجموعه‌ی داده‌های set1 و set2 را در نظر گیرید. اگر هر دوی این دو مجموعه‌ی داده‌ها بر حسب یک یا چند متغیر مرتب شده باشند (مثلاً بر حسب متغیر id)، استفاده از عبارت BY در دستورات زیر باعث می‌شود که مجموعه‌ی داده‌های حاصل، یعنی set3، نیز بر حسب متغیر id مرتب شود:

```
DATA set3;
   SET set1 set2;
   BY id;
RUN;
```

متغیرهای موقتی FIRST.var و LAST.var

اگر در گام DATA از عبارت BY var استفاده کنید، دو متغیر موقتی FIRST.var و LAST.var ساخته می‌شوند. مقدار متغیر FIRST.var برای اولین رخداد هر یک از مقادیر یکتای متغیر var برابر ۱ و در سایر جاها برابر ۰ است. مقدار متغیر LAST.var برای آخرین رخداد هر یک از مقادیر یکتای متغیر var برابر ۱ و در سایر جاها برابر ۰ است. برای روشن شدن مطلب، دستورات زیر و خروجی مربوط را ببینید:

```
DATA scores;
   INPUT name $ course $ score @@;
   CARDS;
Ali Math 15 Ali Science 12 Ali History 15
Reza Math 11 Reza Science 15
Navid Math 17 Navid Science 15
Taghi Science 15
RUN;
PROC SORT DATA = scores;
   BY name;
RUN;
DATA scores1;
   SET scores;
   BY name;
   first_name = FIRST.name;
   last_name  = LAST.name;
RUN;
PROC PRINT DATA = scores1;
RUN;
```

Obs	name	course	score	first_ name	last_ name
1	Ali	Math	15	1	0
2	Ali	Science	12	0	0
3	Ali	History	15	0	1
4	Navid	Math	17	1	0
5	Navid	Science	15	0	1
6	Reza	Math	11	1	0
7	Reza	Science	15	0	1
8	Taghi	Science	15	1	1

مثال: دستورات زیر ماکسیمم نمره‌ی هر دانشجو را نمایش می‌دهد:

```
PROC SORT DATA = scores;
   BY name score;
RUN;
DATA scores1 (DROP = score RENAME = (score = MaxScore));
   SET scores;
   BY name;
   IF LAST.name;
RUN;
PROC PRINT DATA = scores1;
RUN;
```

Obs	name	MaxScore
1	Ali	15
2	Navid	17
3	Reza	15
4	Taghi	15

ترکیب دو یا چند مجموعه‌ی داده‌ها با استفاده از MERGE

وقتی می‌خواهید مشاهده‌های یک مجموعه‌ی داده‌ها را با مشاهده‌های یک مجموعه‌ی دیگر جور کنید می‌توانید از عبارت MERGE در گام DATA استفاده کنید. اگر می‌دانید که ترتیب مشاهده‌ها در در مجموعه‌ی داده‌ها دقیقاً یکی است، نیازی به متغیر مشترک در دو مجموعه‌ی داده‌ها نیست. هرچند در عمل، معمولاً این‌گونه نیست و شما نیاز به متغیر(های) مشترک برای جورسازی صحیح مشاهده‌ها دارید. فرایند ترکیب مجموعه‌های داده‌ها به این صورت است که ابتدا باید با استفاده از شیوه‌ی SORT، مجموعه‌های داده‌ها را بر حسب متغیر(های) مشترک مرتب کنید، سپس در یک گام DATA، فهرست مجموعه‌های داده‌های مورد نظر را در عبارت MERGE، و فهرست متغیر(های) مشترک را در عبارت BY مشخص کنید:

```
DATA نام مجموعه‌ی داده‌ی جدید;
   MERGE فهرست مجموعه‌های داده‌های مورد نظر;
   BY فهرست متغیرهای مشترک;
RUN;
```

اگر مجموعه‌های داده‌ها علاوه بر متغیرهای استفاده‌شده در عبارت BY دارای متغیر(های) مشترک دیگری باشند، مقدار این متغیر(ها) از آخرین مجموعه‌ی داده‌های فهرست مشخص‌شده در عبارت MERGE خوانده می‌شود.

مثال: برنامه‌ی زیر نمره‌های دو کلاس ریاضی و علوم را با هم ترکیب می‌کند:

```
DATA math;
   INPUT id score @@;
   CARDS;
101 15107 14105 11103 17
RUN;
DATA science;
   INPUT id score;
   CARDS;
103 15107 17109 11
RUN;
PROC SORT DATA = math (rename = (score = Math));
   BY id;
RUN;
PROC SORT DATA = science (rename = (score = Science));
   BY id;
RUN;
DATA scores;
   MERGE math science;
   BY id;
RUN;
PROC PRINT DATA = scores;
RUN;
```

Obs	id	Math	Science
1	101	15	.
2	103	17	15
3	105	11	.
4	107	14	17
5	109	.	11

همان‌طور که در خروجی می‌بینید مجموعه‌ی داده‌های حاصل شامل همه‌ی مشاهده‌های موجود در دو مجموعه‌ی داده‌ها (مشاهده‌های مشترک و غیرمشترک) است، برای این‌که فقط مشاهده‌های مشترک را ذخیره کنید، می‌توانید از گزینه‌ی IN = به‌صورت زیر استفاده کنید:

```
DATA scores;
   MERGE math (IN = a) science (IN = b);
   BY id;
   IF a = 1 and b = 1 THEN OUTPUT;
RUN;
```

عبارت IF بالا را به‌صورت زیر نیز می‌توانستید بنویسید:

```
IF a and b;
```

بهنگام کردن مجموعه‌ی داده‌ها با استفاده از UPDATE

شما می‌توانید مقادیر متغیر(های) یک مجموعه‌ی داده‌ها را با استفاده از مقادیر یک مجموعه‌ی داده‌های دیگر بهنگام کنید. برای این کار دو مجموعه‌ی داده‌ها را باید بر اساس متغیر(های) مشترک جور سازید. علاوه بر این، مشاهده‌های مجموعه‌ی داده‌ی اصلی باید بر حسب متغیر(های) مشترک یکتا باشند. مقادیر گم‌شده‌ی موجود در مجموعه‌ی داده‌های بهنگام‌کننده جایگزین مقادیر موجود در مجموعه‌ی داده‌های اصلی نمی‌شود. اگر مجموعه‌ی داده‌های بهنگام‌کننده بر حسب متغیر(های) مشترک، یکتا نباشند مقدار آخرین مشاهده‌ی تکراری مورد نظر، برای بهنگام کردن مورد استفاده قرار می‌گیرد.

```
DATA مجموعه‌ی-داده‌های-اصلی;
   UPDATE مجموعه‌ی-داده‌های-بهنگام‌کننده مجموعه‌ی-داده‌های-اصلی;
   BY متغیر(های)-مشترک;
RUN;
```

شیوه‌ی SQL

۱ مقدمه

زبان پرس‌وجوی ساختار یافته (SQL)[۱] یک زبان استانداردشده و پرکاربرد برای بازیابی و بهنگام‌سازی داده‌ها در جداول و پایگاه‌های داده‌ای رابطه‌ای است. شیوه‌ی SQL بخشی از سامانه‌ی SAS/Base است و می‌تواند با هر مجموعه داده (جدول) SAS مورد استفاده قرار گیرد. PROC SQL اغلب می‌تواند بدیلی برای شیوه‌های دیگر SAS یا گام داده‌ای باشد. شما می‌توانید عناصر برنامه‌ای SAS مانند عبارت‌های سراسری، گزینه‌های مجموعه داده، توابع، و فرمت‌ها را با PROC SQL مورد استفاده قرار دهید. PROC SQL می‌تواند:

- گزارش تولید کند
- آماره‌های خلاصه تولید کند
- داده‌ها را از جداول یا دیدها[۲] بازیابی کند
- داده‌های حاصل از جداول یا دیدها را ترکیب کند
- جداول، دیدها، و ایندکس‌ها را ایجاد کند
- مقادیر داده‌ها را در جداول بهنگام کند
- داده‌های موجود در سامانه‌ی مدیریت پایگاه‌های داده‌ای (DBMS)[۳]
- یک جدول را با اضافه کردن، اصلاح یا حذف ستون‌ها، اصلاح کند.

زبان‌شناسی: جدول زیر جملات معادل در SQL، SAS، و پردازش داده‌ها را فهرست کرده است:

SQL	SAS	پردازش داده‌ها
جدول	فایل داده‌های SAS	فایل
سطر	مشاهده	رکورد
ستون	متغیر	فیلد

پرس‌وجوها: پرس‌وجوها داده‌ها را از یک جدول، دید، یا DBMS بازیابی می‌کنند. شما در PROC SQL، از یک عبارت SELECT و زیرعبارت‌های مرتبط با آن برای ایجاد یک پرس‌وجو استفاده می‌کنید.

دیدها: دیدها هیچ داده‌ای را در بر ندارند، بلکه حاوی یک عبارت SELECT یا یک پرس‌وجوی ذخیره‌شده هستند. زمانی که یک دید را در یک گام شیوه‌ای یا یک گام داده‌ای مورد استفاده قرار می‌دهید، پرس‌وجوی مرتبط با آن اجرا شده و داده‌ها از جدول(های) مربوط بازیابی می‌شوند.

مقادیر پوچ (Null): مقادیر پوچ در SQL معادل مقادیر گم‌شده در SAS هستند.

---

[۱] Structured Query Language

[۲] Views

[۳] Database Management System

۲ بازیابی داده‌ها از یک جدول واحد

برای بازیابی داده‌ها باید از یک عبارت SELECT به‌همراه بندهای[۱] آن استفاده کنید. ترتیب این بندها به‌صورت زیر است:

1. SELECT
2. FROM
3. WHERE
4. GROUP BY
5. HAVING
6. ORDER BY

تنها بندهای SELECT و FROM اجباری بوده و سایر بندها اختیاری هستند.

در عبارت SELECT نام ستون‌هایی را که می‌خواهید در خروجی ظاهر شوند مشخص کنید. نام ستون‌ها را با ویرگول (,) از هم جدا کنید. برای انتخاب همه‌ی ستون‌های یک جدول از ستاره (*) استفاده کنید.

```
proc sql;
      title 'U.S. Cities with Their Coordinates';
      select *
      from sql.uscitycoords;
quit;
```

انتخاب ستون‌ها مورد نظر:

```
proc sql;
      select City, State
      from sql.uscitycoords;
quit;
```

حذف سطرهای تکراری از نتایج پرس‌وجو: در برخی موارد، ممکن است بخواهید فقط مقادیر یکتای موجود در یک ستون را نمایش دهید. در این حالت از کلیدواژه‌ی DISTINCT در عبارت SELECT استفاده کنید.

```
proc sql;
      select distinct Continent
      from sql.unitedstates;
quit;
```

نکته: زمانی که در عبارت SELECT تمام ستون‌های یک جدول را به‌همراه کلیدواژه‌ی DISTINCT مشخص می‌کنید، PROC SQL سطرهای تکراری را از نتایج حذف می‌کند.

تعیین ساختار یک جدول: برای دستیابی به فهرست تمام ستون‌های یک جدول و خصوصیات آن‌ها می‌توانید از عبارت DESCRIBE TABLE استفاده کنید.

```
proc sql;
      describe table sql.unitedstates;
quit;
```

ایجاد ستون‌های جدید: علاوه بر انتخاب ستون‌هایی که در یک جدول وجود دارند، می‌توانید ستون‌های جدیدی را ایجاد کنید که صرفاً در طی یک پرس‌وجو موجود هستند. این ستون‌ها می‌توانند شامل مقادیر ثابت متنی یا عددی، یا محاسباتی بر اساس ستون‌های موجود باشند.

```
proc sql outobs=12;
      select 'Postal code for', Name label = '#', 'is',
             Code label='#'
      from sql.postalcodes;
quit;
```

---

[۱] clause

در مثال بالا، عبارت label = '#' باعث می‌شود نام ستون‌ها در خروجی نمایش داده نشود.

```
proc sql outobs=12;
      select City, (AvgLow - 32) * 5/9 format = 4.1
      from sql.worldtemps;
quit;
```

انتساب یک نام مستعار به یک ستون: با این کار شما می‌توانید نام جدیدی به همر ستون موجود در یک پرس‌وجو منتسب کنید. این نام مستعار باید از قواعد نام‌گذاری SAS تبعیت کند. برای مشخص کردن نام مستعار بعد از نام ستون یا عبارت محاسباتی یا مقدار ثابت، از AS newname استفاده کنید. شما بعداً در داخل این پرس‌وجو می‌توانید به این نام مستعار ارجاع دهید.

```
proc sql outobs=12;
      select City, (AvgLow - 32) * 5/9 as LowCelsius
      from sql.worldtemps;
quit;
```

ارجاع دادن به یک ستون محاسبه‌شده به‌وسیله‌ی نام مستعار: برای این کار باید قبل از نام مستعار از کلیدواژه‌ی CALCULATED استفاده کنید.

```
proc sql;
      select City, (AvgHigh - 32) * 5/9 as HighC,
                   (AvgLow  - 32) * 5/9 as LowC,
                  (calculated HighC - calculated LowC)
                  as Range
      from sql.worldtemps;
quit;
```

تخصیص شرطی مقادیر: عبارت CASE این امکان را به شما می‌دهد تا از یک منطق شرطی برای تخصیص مقادیر استفاده کنید. شما هر جا که بتوانید از نام ستون استفاده کنید از عبارت CASE نیز می‌توانید استفاده کنید.

```
proc sql;
    select City, Country, Latitude,
    case
       when        Latitude >   67 then 'North Frigid'
       when  23 <= Latitude <=  67 then 'North Temperate'
       when -23 <  Latitude <   23 then 'Torrid'
       when -67 <= Latitude <= -23 then 'South Temperate'
       else 'South Frigid'
    end as ClimateZone
    from sql.worldcitycoords
    order by City;
quit;
```

استفاده از شکل عملوند-CASE: شما می‌توانید از این شکل عبارت CASE همانند مثال زیر استفاده کنید:

```
proc sql;
      select Name, Continent,
      case Continent
           when 'North America' then 'Continental U.S.'
           when 'Oceania' then 'Pacific Islands'
           else 'None'
      end as Region
      from sql.unitedstates;
quit;
```

نکته: هنگامی که از CASE به‌صورت بالا استفاده کنید فقط می‌توانید از شروط برابری استفاده کنید، یعنی نمی‌توانید از عملگرهای مقایسه‌ای یا سایر عملگرها همانند آنچه در مثال پیش‌تر بود استفاده کنید.

جایگزینی مقادیر گم‌شده: تابع COALESCE این امکان را به شما می‌دهد که یک مقدار مشخص را جایگزین مقادیر گم‌شده‌ی یک ستون کنید. در واقع، این تابع تمام آرگومان‌هایش تا رسیدن به اولین مقدار غیرگم‌شده بررسی می‌کند و سپس آن مقدار را برمی‌گردند. اگر تمام آرگومان‌ها مقادیر گم‌شده باشند، این تابع یک مقدار گم‌شده برمی‌گرداند.

```
proc sql;
      select Name, coalesce(LowPoint, 'Not Available')
                   as LowPoint
      from sql.continents;
quit;
```

مشخص کردن خصوصیات یک ستون: شما می‌توانید خصوصیات زیر را برای ستون‌ها مشخص کنید، این خصوصیات چگونگی نمایش داده‌ها توسط SAS را مشخص می‌کند:

- FORMAT =
- INFORMAT =
- LABEL =
- LENGTH =

اگر این خصوصیات را مشخص نکنید، PROC SQL از خصوصیاتی که در حال حاضر در جدول ذخیره شده استفاده می‌کند، یا اگر خصوصیاتی ذخیره نشده باشد، از خصوصیات پیش‌فرض استفاده می‌کند.

```
proc sql;
      select Name label='State', Area format=comma10.
      from sql.unitedstates;
quit;
```

نکته: استفاده از کلیدواژه‌ی LABEL = اختیاری است. برای مثال دو عبارت زیر معادل هستند:

```
select Name label='State', Area format=comma10.

select Name 'State', Area format=comma10.
```

مرتب‌سازی نتایج: از بند ORDER BY می‌توانید برای مرتب کردن نتایج یک پرس‌وجو استفاده کنید. فهرست ستون‌هایی را که می‌خواهید مرتب‌سازی بر اساس آن‌ها صورت گیرد در جلوی این بند مشخص کنید. نام ستون‌ها را با ویرگول (,) از هم جدا کنید. این فهرست می‌تواند شامل ستون‌هایی باشد که در عبارت SELECT انتخاب نشده باشند. برای مرتب‌سازی به‌صورت نزولی از عبارت DESC پس از نام ستون مورد نظر استفاده کنید.

```
proc sql;
      select Name, Continent
      from sql.countries
      order by Continent, Name desc;
quit;
```

در مثال بالا، نتایج ابتدا به‌صورت صعودی بر اساس نام قاره، سپس به‌صورت نزولی بر اساس نام کشور مرتب می‌شود.

بازیابی سطرهایی که در شرط خاصی صدق می کنند: برای این کار می‌توانید از بند WHERE استفاده کنید. شرط مورد نظر را در جلوی این بند قرار دهید.

```
proc sql;
      select Name, Population
```

```
      from sql.countries
      where Continent = 'Africa'
            and Population > 20000000
      order by Population desc;
quit;
```

استفاده از عملگر LIKE: از این عملگر برای انتخاب سطرهایی که در یک الگوی خاص صدق می‌کنند استفاده کنید. برای مشخص کردن الگو می‌توانید از نویسه‌های جایگزین[1] % و _ استفاده کنید. برای مثال، پرس‌وجوی زیر فهرست کشورهایی را ارائه می‌کند که نام آن‌ها با حرف Z شروع می‌شود و دارای طول دلخواهی هستند، یا نام آن‌ها دارای پنج حرف بوده و به حرف a ختم می‌شود.

```
proc sql;
      select Name
      from sql.countries
      where Name like 'Z%' or Name like '____a';
quit;
```

تلخیص داده‌ها

شما می‌توانید از توابع تجمیعی (یا توابع تلخیصی) برای تلخیص آماری داده‌های یک جدول استفاده کنید. اگر یک ستون را به‌عنوان آرگومان تابع تجمیعی معرفی کنید، محاسبات روی مقادیر آن ستون صورت می گیرد. اگر چند ستون به‌عنوان آرگومان تابع تجمیعی مورد استفاده قرار گیرد، آن تابع به‌عنوان تابع تجمیعی SQL در نظر گرفته نمی‌شود. در این صورت اگر تابعی با همان نام در فهرست توابع معمولی SAS وجود داشته باشد، SQL آن را به‌عنوان تابع معمولی SAS در نظر گرفته و محاسبات برای هر سطر بر اساس مقادیر ستون‌های مشخص‌شده صورت می‌گیرد. در صورتی که تابعی معمولی با آن نام وجود نداشته باشد، یک خطا رخ می‌دهد. برای مثال، SUM(x) مجموع مقادیر ستون x را می‌دهد، در حالی‌که SUM(x, y, z) برای هر سطر داده‌ها جمع سه ستون x و y و z را نتیجه می‌دهد، استفاده از عبارت AVG(x, y, z) هم یک خطا را به دنبال خواهد داشت.

فهرست برخی از توابع تجمیعی در جدول زیر آمده است:

تابع	شرح	تابع	شرح
AVG، MEAN	میانگین مقادیر	COUNT، FREQ، N	تعداد مقادیر غیرگم‌شده
CV	ضریب تغییرات (درصد)	MAX	بزرگ‌ترین مقدار
MIN	کوچک‌ترین مقدار	NMISS	تعداد مقادیر گم‌شده
STD	انحراف استاندارد	STDERR	خطای استاندارد میانگین
SUM	مجموع مقادیر	VAR	واریانس مقادیر

برای مثال، دستورات زیر مجموع ذخایر نفتی تمام کشورها را ارائه می‌کنید:

```
proc sql;
      select sum(Barrels) format=comma18. as TotalBarrels
      from sql.oilrsrvs;
quit;
```

[1] wildcards

ترکیب آماره‌های خلاصه با داده‌ها: توابع تجمیعی، می‌توانند باعث تکرار نتیجه‌ی محاسبات برای هر سطر داده‌ها شوند. این اتفاق زمانی رخ می‌دهد که داده‌ها را مجدداً ترکیب شوند. ترکیب مجدد در هر یک از شرایط زیر رخ می‌دهد:

- بند SELECT به ستونی ارجاع دهد که شامل یک تابع تجمیعی است ولی در بند GROUP BY نیامده است.
- بند SELECT به ستونی ارجاع دهد که شامل یک تابع تجمیعی است و به ستون‌های دیگری ارجاع دهد که در بند GROUP BY نیامده‌اند.
- یک یا چند ستون یا عبارت‌های ستونی در بند HAVING فهرست شده‌اند ولی در یک زیرپرس‌وجو یا بند GROUP BY نیامده‌اند.

مثال زیر جمعیت چین را که بیش‌ترین جمعیت را دارد برای تمام سطرها تکرار می‌کند:

```
proc sql;
      select Name, Population format=comma20.,
            max(Population) as MaxPopulation format=comma20.
      from sql.countries
      order by Population desc;
quit;
```

در برخی حالات، ممکن است بخواهید از نتیجه‌ی محاسبات یک تابع تجمیعی در محاسبات دیگری استفاده کنید. برای این کار، کافی است یک پرس‌وجو برای PROC SQL بسازید تا هر دو محاسبه به‌صورت خودکار انجام شود. این عمل باعث می‌شود تا ترکیب مجدد داده‌ها صورت گیرد.

برای مثال، اگر بخواهید درصد جمعیت هر کشور از جمعیت کل جهان را محاسبه کنید، باید پرس‌وجویی بسازید که:

- با استفاده از تابع SUM جمعیت کل جهان را به‌دست آورد
- جمعیت هر کشور را بر جمعیت کل جهان تقسیم کند.

PROC SQL یک پرس‌وجوی داخلی برای پیدا کردن مجموع اجرا می‌کند، سپس پرس‌وجوی داخلی دیگری را برای تقسیم جمعیت هر کشور بر مجموع اجرا می‌کند.

```
proc sql;
      select Name, Population format=comma14.,
            Population / sum(Population) * 100 as Percentage
            format=comma8.2
      from sql.countries
      order by Percentage desc;
quit;
```

استفاده از توابع تجمیعی با مقادیر یکتا: می‌توان از DISTINCT با یک تابع تجمیعی استفاده کرد تا تابع فقط از مقادیر یکتای یک ستون استفاده کند.

برای مثال، پرس‌وجوی زیر تعداد قاره‌های مجزا و غیرگم‌شده‌ی موجود در جدول مربوط را برمی‌گرداند:

```
proc sql;
      select count(distinct Continent) as Count
      from sql.countries;
run;
```

نکته: برای شمارش تمام سطرهای یک جدول از عبارت COUNT(*) استفاده کنید.

پرس‌وجوی زیر تعداد کشورهای موجود در جدول مربوط را ارائه می‌کند:

```
proc sql;
```

```
      select count(*) as Number
      from sql.countries;
quit;
```

گروه‌بندی داده‌ها

بند GROUP BY داده‌ها را بر اساس یک یا چند ستون گروه‌بندی می‌کند. زمانی که از بند GROUP BY استفاده می‌کنید، می‌توانید از یک تابع تجمیعی نیز در بند SELECT یا در یک بند HAVING استفاده کنید تا PROC SQL را رهنمون کنید که چگونه داده‌ها را برای هر گروه تلخیص کند. PROC SQL تابع تجمیعی را برای هر گروه به‌صورت جداگانه محاسبه می‌کند.

مثال زیر جمعیت کل کشورهای هر قاره را محاسبه می‌کند:

```
proc sql;
      select Continent,
             sum(Population) format=comma14. as TotalPopulation
      from sql.countries
      where Continent is not missing
      group by Continent;
quit;
```

گروه‌بندی بدون تلخیص: زمانی که از یک بند GROUP BY بدون یک تابع تجمیعی استفاده می‌کنید، PROC SQL با GROUP BY همانند یک بند ORDER BY رفتار می‌کند.

شما می‌توانید گروه‌بندی را بر اساس بیش از یک ستون انجام دهید. برای این کار فهرست ستون‌ها را، که با ویرگول از هم جدا شده‌اند، جلوی بند GROUP BY قرار دهید. مثال زیر، مجموع مساحت تمام صحراها و دریاچه‌ها را برای هر موقعیت جغرافیایی محاسبه می‌کند.

```
proc sql;
      select Location, Type,
             sum(Area) as TotalArea format=comma16.
      from sql.features
      where type in ('Desert', 'Lake')
      group by Location, Type
      order by Location desc;
quit;
```

پالایش داده‌های گروه‌بندی‌شده

می‌توان از بند HAVING به‌همراه بند GROUP BY برای پالایش[1] داده‌های گروه‌بندی‌شده استفاده کنید. استفاده از بند HAVING برای گروه‌ها مشابه استفاده از WHERE برای سطرهای فردی است. وقتی از یک بند HAVING استفاده می‌کنید، تنها گروه‌هایی نمایش می‌یابند که در شرایط عبارت HAVING صدق می‌کنند.

مثال زیر ویژگی‌ها را بر اساس Type گروه بندی می‌کند و فقط تعداد جزایر، اقیانوس‌ها و دریاها را نمایش می‌دهد:

```
proc sql;
      select Type, count(*) as Number
      from sql.features
      group by Type
      having Type in ('Island', 'Ocean', 'Sea')
      order by Type;
quit;
```

[1] filter

انتخاب بین HAVING و WHERE: تفاوت‌های موجود بین بندهای HAVING و WHERE در جدول زیر آمده است. از آن‌جایی که بند HAVING هنگام کار با گروه‌ها مورد استفاده قرار می‌گیرد، پرس‌وجوهای شامل HAVING اغلب شامل موارد زیر نیز هستند:

- یک بند GROUP BY
- یک تابع تجمیعی

بند HAVING	بند WHERE
عموماً برای تعیین شرایطی برای شامل بودن یا نبودن گروه‌های حاصل از سطرهای یک جدول به‌کار می‌رود.	برای تعیین شرایطی برای شامل بودن یا نبودن سطرهای فردی یک جدول به‌کار می‌رود.
در صورت وجود بند GROUP BY در پرس‌وجو، باید بعد از بند GROUP BY بیاید.	در صورت وجود بند GROUP BY در پرس‌وجو، باید قبل از بند GROUP BY بیاید.
متأثر از بند GROUP BY است؛ وقتی از بند GROUP BY استفاده نشود، بند HAVING همانند WHERE عمل می‌کند.	متأثر از بند GROUP BY نیست.
بعد از بند GROUP BY و توابع تجمیعی پردازش می‌شود.	قبل از بند GROUP BY (در صورت وجود) و قبل از توابع تجمیعی پردازش می‌شود.

استفاده از HAVING به‌همراه توابع تجمیعی: پرس‌وجوی زیر جمعیت تمام قاره‌هایی را که دارای بیش از ۱۵ کشور هستند برمی‌گرداند:

```
proc sql;
      select Continent,
             sum(Population) as TotalPopulation
             format=comma16.,
      count(*) as Count
      from sql.countries
      group by Continent
      having count(*) > 15
      order by Continent;
quit;
```

اعتبارسنجی یک پرس‌وجو

عبارت VALIDATE این امکان را به شما می‌دهد که بدون اجرای یک پرس‌وجو، درستی گرامر آن را بررسی کنید. در این حالت، پیغامی در پنجره‌ی log مبنی بر درست بودن یا نبودن گرامر آن پرس‌وجو نمایش می‌یابد.

```
proc sql;
      validate
      select Name, Statehood
      from sql.unitedstates
      where Statehood < '01Jan1800'd;
quit;
```

۳ بازیابی داده‌ها از چند جدول

داده‌های مورد نیاز شما برای تهیه‌ی گزارش ممکن است در بیش از یک جدول قرار داشته باشند. برای انتخاب داده‌ها از این جدول‌ها، آن‌ها را در یک پرس‌وجو به هم متصل کنید. اتصال[1] جدول‌ها شما را قادر می‌سازد تا داده‌ها را از چندین جدول انتخاب کنید، گویی که در یک جدول قرار دارند. اتصال‌ها جدول‌های اصلی را تغییر نمی‌دهند.

ابتدایی‌ترین اتصال، فهرست کردن نام دو جدول در بند FROM یک عبارت SELECT است. مثال زیر و خروجی مربوط را در نظر بگیرید:

```
data One;
   input x y;
   cards;
1 2
2 3
run;
data Two;
   input x z;
   cards;
2 5
3 6
4 9
run;
proc sql;
      select *
      from One, Two;
quit;
```

حاصل‌ضرب دکارتی جدول One و Two

x	y	x	z
1	2	2	5
1	2	3	6
1	2	4	9
2	3	2	5
2	3	3	6
2	3	4	9

اتصال جدول‌ها به این شکل، حاصل‌ضرب دکارتی جدول‌ها را برمی‌گرداند. هر سطر جدول اول با هر سطر جدول دوم ترکیب می‌شود. اگر این پرس‌وجو را اجرا کنید، پیغام زیر در پنجره‌ی log ظاهر می‌شود:

```
NOTE: The execution of this query involves performing one or more Cartesian product
      joins that can not be optimized.
```

حاصل‌ضرب دکارتی جداول بزرگ ممکن است بسیار بزرگ باشد. شما عموماً، زیرمجموعه‌ای از حاصل‌ضرب دکارتی را می‌خواهید. شما این زیرمجموعه را با تعیین نوع اتصال مشخص می‌کنید.

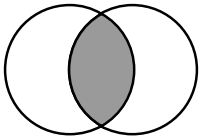
دو نوع اتصال وجود دارد:

- اتصال درونی: حاصل این اتصال تمام سطرهای موجود در یک جدول است که یک یا چند سطر جور در جدول یا جداول دیگری که در عبارت FROM لیست شده‌اند، دارد.

[1] join

- اتصال بیرونی: حاصل این اتصال، یک اتصال درونی به‌اضافه‌ی سطرهایی است که با هیچ سطری در جداول دگر اتصال جور نیستند. سه نوع اتصال بیرونی وجود دارد: چپ، راست، و کامل.

اتصال درونی:



یک اتصال درونی فقط زیرمجموعه‌ای از سطرهای جدول اول را برمی‌گرداند که با سطرهایی از جدول دوم جور هستند. شما می‌توانید ستون‌هایی را که می‌خواهید برای جور بودن مقایسه شوند در بند WHERE مشخص کنید.

کد زیر یک بند WHERE به پرس‌وجوی قبلی اضافه می‌کند. بند WHERE تعیین می‌کند که فقط سطرهایی از جدول One که مقادیر ستون x آن‌ها با مقادیر ستون x جدول Two جور است، باید در خروجی ظاهر شوند. خروجی این پرس‌وجو را با خروجی قبلی مقایسه کنید.

```
proc sql;
      select * from one, two
      where one.x = two.x;
quit;
```

توجه داشته باشید که نام ستون‌ها در بند WHERE بعد از نام جدول مربوط آمده است. این کار توصیف نام ستون‌ها نام دارد، و استفاده از آن هنگامی که ستون‌هایی را که دارای نام یکسان در دو یا چند جدول هستند مشخص می‌کنید، ضروری است. توصیف نام ستون مانع از ایجاد ارجاع به ستون نامعلوم می‌شود.

استفاده از نام‌های مستعار جداول: نام مستعار جدول، یک نام بدیل موقت برای یک جدول است. نام‌های مستعار را می‌توان در بند FROM تعیین کرد. نام‌های مستعار جداول در اتصال‌ها برای توصیف نام جداول مورد استفاده قرار می‌گیرد و می‌تواند با کوتاه کردن نام جداول، خواندن پرس‌وجو را ساده‌تر کند.

نام مستعار را به این صورت می‌توانید مشخص کنید:

نام جدول AS نام مستعار

نکته: استفاده از کلیدواژه‌ی AS اختیاری است.

مثال:

```
proc sql;
      select * from sql.oilprod as p, sql.oilrsrvs as r
      where p.country = r.country;
quit;
```

تعیین ترتیب خروجی اتصال: شما می‌توانید خروجی جداول اتصال‌یافته را بر حسب یک یا چند ستون از هر یک از دو جدول مرتب کنید. مثال زیر را ببینید:

```
proc sql outobs=6;
      select p.country, barrelsperday 'Production',
            barrels 'Reserves'
      from sql.oilprod p, sql.oilrsrvs r
      where p.country = r.country
      order by barrelsperday desc;
run;
```

ایجاد اتصال درونی با استفاده از کلیدواژه‌ی INNER JOIN: از این‌کلیدواژه می‌توان برای اتصال جداول استفاده کرد. در این حالت، بند ON جانشین بند WHERE برای مشخص کردن ستون‌های اتصال می‌شود. عملکرد این روش دقیقاً معادل روش قبلی است.
کد زیر همان خروجی کد قبلی را تولید می‌کند:

```
proc sql;
      select p.country, barrelsperday 'Production',
             barrels 'Reserves'
      from sql.oilprod p inner join sql.oilrsrvs r
      on p.country = r.country
      order by barrelsperday desc;
quit;
```

ایجاد اتصال‌های چند ستونی: وقتی یک سطر به‌وسیله‌ی ترکیبی از مقادیر موجو در بیش از یک ستون متمایز می‌شود، همه‌ی ستون‌های مورد نیاز را در اتصال به‌کار گیرید. برای مثال، یک نام شهر ممکن است در بیش از یک کشور وجود داشته باشد. برای انتخاب صحیح شهر، باید هم ستون شهر و هم ستون کشور را در اتصال مشخص کنید. مثال زیر طول و عرض جغرافیایی شهرهای بزرگ را نمایش می‌دهد:

```
proc sql;
      select Capital, Name,
             City, Country,
             latitude, longitude
      from sql.countries, sql.worldcitycoords
      where Capital like 'L%' and
            Capital = City and
            Name = Country;
quit;
```

انتخاب داده‌ها از بیش از دو جدول: داده‌های مورد نیاز شما ممکن است در بیش از دو جدول قرار گرفته باشند. برای مثال، اگر بخواهید مختصات مراکز ایالت‌های ایالات متحد را نشان دهید، باید جدول UNITEDSTATES را که حاوی مراکز ایالت‌هاست، با جدول USCITYCOORDS، که حاوی مختصات شهرهای ایالات متحد است، به یکدیگر متصل کنید.
اتصال شهرها از طریق اتصال UINTEDSTATES.Capital به USCITYCOORDS.City ساده است. هر چند، در جدول UNITEDSTATES ستون Name حاوی نام کامل ایالت‌هاست، در حالی که در USCITYCOORDS ایالت‌ها به‌وسیله‌ی کد پستی مشخص شده‌اند. بنا بر این اتصال مستقیم این دو جدول بر اساس ستون‌های ایالت ممکن نیست. برای حل این مسئله، لازم است که از جدول POSTALCODES، که حاوی نام ایالت‌ها و کد پستی آن‌هاست، به‌عنوان یک جدول میانی برای برقراری ارتباط صحیح بین UNITEDSTATES و USCITYCOORDS استفاده کنیم.

```
proc sql;
      select us.Capital, us.Name 'State',
             pc.Code, c.Latitude, c.Longitude
      from sql.unitedstates us, sql.postalcodes pc,
           sql.uscitycoords c
      where us.Capital = c.City and
            us.Name = pc.Name and
            pc.Code = c.State;
quit;
```

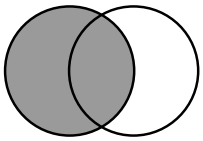
نشان دادن ارتباط درون یک جدول واحد با استفاده از خوداتصالی: وقتی نمایش ارتباط نسبی بین مقادیر یک جدول مورد نیاز باشد، گاهی اوقات لازم است که ستون‌های درون یک جدول را به هم متصل کنیم. اتصال یک جدول به خودش، خوداتصالی یا اتصال انعکاسی نام دارد. شما می‌توانید خوداتصالی را این‌گونه تصور کنید که PROC SQL یک کپی داخلی از یک جدول می‌سازد و آن جدول را به کپی خود متصل می‌کند.
برای مثال، کد زیر از خوداتصالی برای انتخاب شهرهایی که متوسط سالانه‌ی درجه‌ی حرارت بالای آن‌ها با متوسط سالانه‌ی درجه‌ی حرارت پایین سایر شهرهاست، استفاده می‌کند:

```
proc sql;
      select High.City, High.Country,
             High.AvgHigh, ' | ',
             Low.City, Low.Country,
             Low.AvgLow
      from sql.worldtemps High, sql.worldtemps Low
      where High.AvgHigh = Low.AvgLow and
            High.city ^= Low.city and
            High.country ^= Low.country;
quit;
```

اتصال‌های بیرونی

اتصال‌های بیرونی عبارت‌اند از اتصالات درونی به‌اضافه‌ی سطرهایی که با هیچ طری در جدول دیگر جور نیستند. خروجی حاصل شامل سطرهای جور شده و سطرهای جور نشده‌ی حاصل از جداول مرجع اتصال است. سطرهای جور نشده داری مقادیر پوچ (گم‌شده) برای ستون‌های حاصل از جدول جور نشده هستند. از بند ON به جای بند WHERE برای مشخص کردن ستون یا ستون‌هایی که بر اساس آن‌ها اتصال بین جداول را برقرار می‌کنید، استفاده کنید.

اضافه کردن سطرهای جور نشده با اتصال بیرونی چپ:



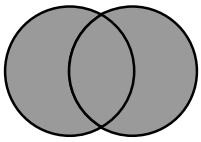
اتصال بیرونی چپ، سطرهای جور شده و سطرهایی از جدول سمت چپی (اولین جدول فهرست شده در بند FROM) که با هیچ سطری در جدول سمت راستی جور نیستند را فهرست می‌کند. اتصال چپ با کلیدواژه‌های LEFT JOIN و ON مشخص می‌شود.
مثال

```
proc sql;
      select Capital, Name 'Country',
             Latitude, Longitude
      from sql.countries a left join sql.worldcitycoords b
      on a.Capital = b.City and
         a.Name = b.Country
      order by Capital;
quit;
```

اضافه کردن سطرها با اتصال بیرونی راست: اتصال راست با کلیدواژه‌های RIGHT JOIN و ON، متضاد اتصال چپ است: ستون‌های جور نشده از جدول سمت راستی (دومین جدولی که در بند FROM فهرست شده است) به‌همراه همه‌ی سطرهای جور شده، در خروجی ظاهر می‌شوند.
مثال

```
proc sql;
      select City, Country, Population
      from sql.countries right join sql.worldcitycoords
      on Capital = City and
         Name = Country
      order by City;
quit;
```

انتخاب همه‌ی سطرها با اتصال بیرونی کامل:



اتصال بیرونی کامل، که با کلیدواژه‌های FULL JOIN و ON مشخص می‌شود، تمام سطرهای جور شده و جور نشده را انتخاب می‌کند.

مثال

```
proc sql;
      select City '#City#(WORLDCITYCOORDS)',
             Capital '#Capital#(COUNTRIES)',
             Population, Latitude, Longitude
      from sql.countries full join sql.worldcitycoords
      on Capital = City and
         Name = Country;
quit;
```

استفاده از تابع COALESCE در اتصال‌ها: همان‌طور که در مثال‌های قبلی دیدید، سطرهای جور نشده در اتصال‌های بیرونی شامل مقادیر گم‌شده هستند. با استفاده از تابع COALESCE، می‌توانید روی ستون‌ها را بپوشانید به‌طوری که فقط سطر حاصل از جدول حاوی داده‌ها فهرست شود. به یاد بیاورید که تابع COALESCE فهرستی از ستون‌ها را به‌عنوان آرگومان می‌پذیرد و نخستین مقدار غیرگم‌شده را که با آن مواجه می‌شود برمی‌گرداند.

این مثال تابع COALESCE را به مثال قبلی می‌افزاید. Countries.Name به‌عنوان آرگومان تابع COALESCE به‌کار برده می‌شود چون برخی جزایر فاقد پایتخت هستند:

```
proc sql outobs=10;
      select coalesce(Capital, City, Name) 'City',
             coalesce(Name, Country) 'Country',
             Population, Latitude, Longitude
      from sql.countries full join sql.worldcitycoords
      on Capital = City and Name = Country;
run;
```

استفاده از زیرپرس‌وجوها برای انتخاب داده‌ها

در حالی‌که یک اتصال جدولی چندین جدول را برای رسیدن به یک جدول جدید ترکیب می‌کند، یک زیرپرس‌وجو (که بین دو پرانتز محصور می‌شود) سطرهایی را از یک جدول بر اساس مقادیر موجود در جدول دیگر انتخاب می‌کند. یک زیرپرس‌وجو، یا پرس‌وجوی درونی، پرس‌وجویی است که به‌عنوان بخشی از پرس‌وجوی دیگر عمل می‌کند. یک زیرپرس‌وجو ممکن است یک مقدار واحد یا چندین مقدار را برگرداند. زیرپرس‌وجوها اغلب در عبارت‌های WHERE و HAVING استفاده می‌شوند.

زیرپرس‌وجوهای تک-مقداری: یک زیرپرس‌وجوی تک-مقداری یک سطر و یک ستون واحد را برمی‌گرداند. این زیرپرس‌وجو را می‌توان در یک بند WHERE یا HAVING با یک عملگر مقایسه‌ای مورد استفاده قرار داد. این زیرپرس‌وجو باید فقط یک مقدار را برگرداند، در غیر این‌صورت پرس‌وجو به شکست می‌انجامد و یک پیغام خطا در log چاپ می‌شود.

پرس‌وجوی زیر از یک زیرپرس‌وجو در بند WHERE خود، برای انتخاب ایالت‌هایی که دارای جمعیتی بیش از جمعیت بلژیک هستند، استفاده می‌کند. ابتدا زیرپرس‌وجو ارزیابی می‌شود، سپس جمعیت بلژیک را به پرس‌وجوی بیرونی برمی‌گرداند.

```
proc sql;
      select Name 'State', population format=comma10.
      from sql.unitedstates
      where population >
            (select population from sql.countries
             where name = "Belgium");
quit;
```

زیرپرس‌وجوهای چند-مقداری: یک زیرپرس‌وجوی چند-مقداری می‌تواند بیش از یک مقدار را از یک ستون برگرداند. این زیرپرس‌وجو در یک عبارت WHERE یا HAVING که حاوی عملگر IN یا یک عملگر مقایسه‌ای که با ANY یا ALL تعدیل می‌شود، مورد استفاده قرار می‌گیرد. مثال زیر جمعیت کشورهای تولیدکننده‌ی نفت را نمایش می‌دهد:

```
proc sql;
      select name 'Country', Population format=comma15.
      from sql.countries
      where Name in
            (select Country from sql.oilprod);
quit;
```

اگر در این پرس‌وجو از عملگر NOT IN استفاده کنید، خروجی، جمعیت کشورهای غیر تولیدکننده‌ی نفت خواهد شد.

زیرپرس‌وجوهای همبسته: پرس‌وجوهای قبلی دارای زیرپرس‌وجوهای ساده‌ای بودند که خودکفا بوده و مستقل از پرس‌وجوی بیرونی اجرا می‌شوند. یک زیرپرس‌وجوی همبسته نیازمند مقدار یا مقادیری از پرس‌وجوی بیرونی است. پس از این‌که زیرپرس‌وجو اجرا شد، نتایج را به پرس‌وجوی بیرونی برمی‌گرداند. زیرپرس‌وجوهای همبسته می‌توانند یک یا چندین مقدار را برگردانند.

مثال زیر تمام ذخایر نفتی کشورهای قاره‌ی آفریقا را انتخاب می‌کند:

```
proc sql;
      select * from sql.oilrsrvs o
      where 'Africa' =
            (select Continent from sql.countries c
             where c.Name = o.Country);
quit;
```

بررسی وجود گروهی از مقادیر: شرط EXISTS وجود مجوعه‌ای از مقادیر را بررسی می‌کند. یک شرط EXISTS درست است اگر سطر یا سطرهایی به‌وسیله‌ی زیرپرس‌وجو تولید شود، و نادرست است اگر هیچ سطری تولید نشود. برعکس، شرط NOT EXIST درست است اگر یک زیرپرس‌وجو یک جدول خال تولید کند.

مثال زیر همان نتایج مثال قبلی را تولید می‌کند. EXISTS وجود کشورهایی در قاره‌ی آفریقا را که دارای ذخایر نفتی هستند بررسی می‌کند. توجه داشته باشید که در اینجا، بند WHERE در زیرپرس‌وجو شامل شرط Continent = ‘Africa’ است که در مثال قبلی در پرس‌وجوی بیرونی بود.

```
proc sql;
      select * from sql.oilrsrvs o
      where exists
            (select Continent from sql.countries c
             where o.Country = c.Name and
                   Continent = 'Africa');
quit;
```

سطوح چندگانه زیرپرس‌وجوهای آشیانی: زیرپرس‌وجوها می‌توانند به‌صورت آشیانی تعریف شوند به‌طوری که درونی‌ترین زیرپرس‌وجو مقدار یا مقادیری را برای استفاده توسط پرس‌وجوی بیرونی بعدی برمی‌گرداند. سپس، مقدار یا مقادیر آن زیرپرس‌وجو توسط پرس‌وجوی بیرونی بعدی استفاده می‌شود و قس علی هذا. ارزیابی همیشه با درونی‌ترین زیرپرس‌وجو شروع می‌شود و به طرف بیرون ادامه می‌یابد.

این مثال شهرهای آفریقایی واقع در کشورهای با ذخایر عمده‌ی نفتی را فهرست می‌کند.

❶ ابتدا درونی‌ترین پرس‌وجو ارزیابی می‌شود. این پرس‌وجو کشورهایی را که در قاره‌ی آفریقا واقع شده‌اند برمی‌گرداند.

❷ زیرپرس‌وجوی بیرونی ارزیابی می‌شود. این زیرپرس‌وجو زیرمجموعه‌ای از کشورهای آفریقایی که دارای ذخایر عمده‌ی نفتی هستند را از طریق مقایسه‌ی فهرست کشورهایی که توسط زیرپرس‌وجوی دروی برگردانده شده بود با کشورهای موجود در جدول Oilrsrvs، برمی‌گرداند.

❸ سرانجام، بند WHERE در پرس‌وجوی بیرونی مختصات شهرهایی را که در جدول Worldcitycoords هستند و کشورشان با نتایج زیرپرس‌وجوی بیرونی جور است، برمی‌گرداند.

```
proc sql;
      select * from sql.worldcitycoords
      ❸ where country in
            ❷ (select Country from sql.oilrsrvs o
              where o.Country in
                    ❶ (select Name from sql.countries c
                       where c.Continent='Africa'));
quit;
```

ترکیب یک اتصال با یک زیرپرس‌وجو: شما می‌توانید اتصال‌ها و زیرپرس‌وجوها را در یک پرس‌وجوی واحد ترکیب کنید. فرض کنید می‌خواهید نزدیک‌ترین شهر به هر یک از شهرهای جدول Uscitycoords را بیابید. پرس‌وجو باید ابتدا شهر A را انتخاب می‌کند، فاصله‌ی شهر A را از سایر شهرها محاسبه کند، و سرانجام شهری را که دارای حدّ اقل فاصله از شهر A است انتخاب کند. این کار را می‌توانید با اتصال جدول Uscitycoords را به خودش (خوداتصالی) و سپس تعیین نزدیک‌ترین فاصله‌ی بین شهرها با استفاده از یک خوداتصالی دیگر در یک زیرپرس‌وجو انجام دهید.

فرمول محاسبه‌ی فاصله‌ی بین دو مختصات عبارت است از:

```
SQRT(((Latitude2-Latitude1)**2) + ((Longitude2-Longitude1)**2))
```

هر چند نتایج این فرمول به دلیل اعوجاج ناشی از انحنای زمین کاملاً دقیق نیست، ولی برای این مثال برای تعیین شهری که از دیگری نزدیک‌تر است، به اندازه‌ی کافی دقیق است.

```
proc sql;
```

```
      select a.City, a.State,
             a.Latitude 'Lat', a.Longitude 'Long',
             b.City, b.State,
             b.Latitude 'Lat', b.Longitude 'Long',
             sqrt(((b.latitude-a.latitude)**2) +
             ((b.longitude-a.longitude)**2)) as dist
      from sql.uscitycoords a, sql.uscitycoords b
      where a.city ^= b.city and
            calculated dist =
               (select min(sqrt(((d.latitude-c.latitude)**2) +
                     ((d.longitude-c.longitude)**2)))
                from sql.uscitycoords c, sql.uscitycoords d
                where c.city = a.city and
                      c.state = a.state and
                      d.city ne c.city)
      order by a.city;
quit;
```

ترکیب پرس‌وجوها با استفاده از عملگرهای مجموعه‌ای

PROC SQL می تواند نتایج دو یا چند پرس‌وجو را به راه‌های متعددی با استفاده از عملگرهای مجموعـه‌ای زیر ترکیب کند:

UNION (اجتماع)	تمام سطرهای یکتای حاصل از هر دو پرس‌وجو را برمی‌گرداند.
EXCEPT (تفاضل)	تمام سطرهایی را که فقط در پرس‌وجوی اول قرار دارند برمی‌گرداند.
INTERSECT (اشتراک)	سطرهای مشترک در نتایج دو پرس‌وجو را برمی‌گرداند.
OUTER UNION (اجتماع بیرونی)	نتایج پروس‌وجوها را به‌دنبال هم می‌آورد.

این عملگرها بین دو پرس‌وجو استفاده می‌شود، برای مثال:

```
select نام ستون‌ها from نام جدول
عملگر مجموعه‌ای
select نام ستون‌ها from نام جدول;
```

علامت «;» را فقط بعد از آخرین عبارت SELECT قرار دهید. عملگرهای مجموعه‌ای ستون‌های حاصـل از دو پرس‌وجو را بر اساس موقعیت آن‌ها در جداول مرجع بدون در نظر گرفتن نام ستون‌ها ترکیب می‌کند. نـوع داده‌ی ستون‌های موجود در موقعیت نسبی یکسـان در دو پـرس‌وجـو بایـد یکسـان باشـد. نـام سـتون‌هـا در پرس‌وجوی اول نام ستون‌ها در جدول خروجی خواهد شد. کلیدواژه‌های اختیاری زیر کنتـرل بـیش‌تـری روی عملگرهای مجموعه‌ای در اختیار شما قرار می‌دهند:

ALL: سطرهای تکراری را کنار نمی‌گـذارد. وقتـی از کلیـدواژه‌ی ALL اسـتفاده مـی‌کنیـد، PROC SQL داده‌ها را، برای حذف سطرهای تکراری، مجدداً بررسی نمی‌کند. بنا بر این، استفاده از ALL کاراتر از اسـتفاده نکردن از آن است.

(CORRESPONDING (CORR: سـتون‌هـایی را کـه دارای نـام یکسـان در هـر دو جـدول هسـتند را می‌پوشاند. وقتی این گزینه را با EXCEPT، INTERSECT، و UNION استفاده کنید، ستون‌هایی کـه در هر دو جدول نیستند نادیده گرفته می‌شوند.

هر عملگر مجموعه‌ای بر اساس دو جدول زیر تشریح می‌شود.

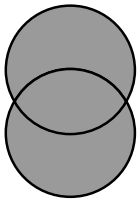
Table A	Table B

x	y
1	one
2	two
2	two
3	Three

x	y
1	one
2	two
4	Four

اتصال‌ها جداول را به‌صورت افقی ترکیب می‌کنند، در حالی که عملگرهای مجموعه‌ای جداول را به‌صورت عمودی ترکیب می‌کنند. بنا بر این، نمودارهای مجموعه‌ای موجود در هر بخش به‌صورت عمودی نشان داده می‌شوند.

تولید سطرهای یکتای حاصل از دو پرس‌وجو (UNION)



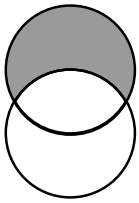
عملگر UNION نتایج دو پرس‌وجو ترکیب می‌کند و تما سطرهای یکتایی که از هر دو پرس‌وجو حاصل می‌شود را برمی‌گرداند؛ یعنی، سطری را که در اولی، یا دومی، یا هر دو موجود است، برمی‌گرداند. UNION سطرهای تکراری را برنمی‌گرداند. اگر یک سطر بیش از یک بار ظاهر شود، فقط یک بار برگردانده می‌شود.

```
proc sql;
      select * from a
      union
      select * from b;
quit;
```

شما می‌توانید از کلیدواژه‌ی ALL استفاده کنید تا سطرهای تکراری در خروجی باقی بمانند.

```
proc sql;
      select * from a
      union all
      select * from b;
quit;
```

تولید سطرهایی که فقط در نتایج پرس‌وجوی اول هستند (EXCEPT)



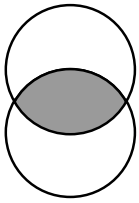
عملگر EXCEPT سطرهای حاصل از پرس‌وجوی اول که در پرس‌وجوی دوم نیستند را برمی‌گرداند.

```
proc sql;
      select * from a
      except
      select * from b;
quit
```

توجه داشته باشید که سطر تکراری موجود در جدول A در خروجی ظاهر نمی‌شود. EXCEPT سطرهای تکراری که با سطرهای موجود در پرس‌وجوی دوم جور نیستند را برنمی‌گرداند. اضافه کردن ALL هر سطر تکراری که در پرس‌وجوی دوم وجود ندارد را نگه می‌دارد.

```
proc sql;
      select * from a
      except all
      select * from b;
quit;
```

تولید سطرهای که به نتایج هر دو پرس‌وجو تعلق دارد (INTERSECT)

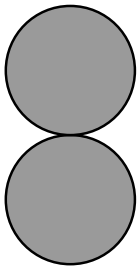


عملگر INTERSECT سطرهای حاصل از پرس‌وجوی اول را که در پرس‌وجوی دوم نیز وجود دارند برمی‌گرداند.

```
proc sql;
      select * from a
      intersect
      select * from b;
quit;
```

خروجی عملگر INTERSECT ALL حاوی سطرهای تولید شده در پرس‌وجوی اول است که با سطرهای تولید شده توسط پرس‌وجوی دوم به‌صورت یک به یک جور هستند. در این مثال، خروجی INTERSECT ALL همان خروجی INTERSECT است.

به دنبال هم آوردن نتایج پرس‌وجوها (OUTER UNION)



عملگر OUTER UNION نتایج پرس‌وجوها را به دنبال هم قرار می‌دهد.

```
proc sql;
      select * from a
      outer union
      select * from b;
quit;
```

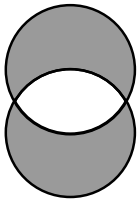
x	y	x	y
1	one	.	
2	two	.	
2	two	.	
3	Three	.	
.		1	one
.		2	two
.		4	four

توجه کنید که OUTER UNION ستون‌های حاصل از دو پرس‌وجو را نمی‌پوشاند. برای پوشاندن ستون‌های موجود در موقعیت یکسان، از کلیدواژه‌ی CORRESPONDING استفاده کنید.

```
proc sql;
      select * from a
      outer union corr
      select * from b;
quit;
```

x	y
1	one
2	two
2	two
3	Three
1	one
2	two
4	four

تولید سطرهای حاصل از پرس‌وجوی اول یا پرس‌وجوی دوم (اجتماع انحصاری):



هیچ کلیدواژه‌ای وجود ندارد که سطرهای یکتایی را برگرداند که در جدول اول یا دوم هستند ولی در هر دو نیستند. مثال زیر رهای برای انجام این عملیات است:

```
proc sql;
      (select * from a
       except
       select * from b)
      union
      (select * from b
       except
       select * from a);
quit;
```

اجتماع انحصاری

x	y
3	Three
4	four

٤ ایجاد و بهنگام کردن جداول و دیدها

ایجاد جداول

عبارت CREATE TABLE این امکان را می‌دهد که جداول بدون سطر داده‌ها از طریق تعریف ستون‌ها یا ایجاد کنید، یا جداولی را با استفاده از نتایج یک پرس‌وجو ایجاد کنید.

ایجاد جداول از طریق تعریف ستون‌ها: شما می‌توانید یک جدول جدید بدون سطر را با استفاده از عبارت CREATE TABLE از طریق تعریف ستون‌ها و ویژگی‌های آن‌ها، ایجاد کنید. شما می‌توانید نام ، نوع، طول، اینفرمت، فرمت، و برچسب ستون را تعیین کنید.

مثال زیر جدولی با نام newstates، در کتابخانه‌ی sql، ایجاد می‌کند که دارای یک ستون به نام state، از نوع نویسه‌ای، به طول ۲؛ یک ستون به نام date، از نوع عددی، با فرمت و اینفرمت از نوع تاریخ ۹ حرفی؛ و یک ستون به نام population ، از نوع عددی است.

```
proc sql;
      create table sql.newstates
            (state char(2),
             date num
                  informat = date9.
                  format = date9.,
             population num);
quit;
```

ایجاد جداول از نتایج یک پرس‌وجو: برای ایجاد جدول از نتایج یک جستجو، عبارت CREATE TABLE را پیش از عبارت SELECT قرار دهید. وقتی جدولی از این طریق ایجاد می‌شود، داده‌های آن از جدول یا دیدی که در بند FROM پرس‌وجو آمده است، گرفته می‌شود. نام ستون‌های جدول جدید همان نام‌هایی است که در بند SELECT فهرست شده است. ویژگی‌های ستون‌ها (نوع، طول، اینفرمت، و فرمت) همان ویژگی‌های ستون‌های انتخاب شده است.

مثال زیر جدولی به نام densities از جدول countries می‌سازد.

```
proc sql;
      create table sql.densities as
      select Name 'Country' format $15.,
             Population format=comma10.0,
             Area as SquareMiles,
             Population / Area format=6.2 as Density
      from sql.countries;
quit;
```

در این شکل عبارت CREATE TABLE، اختصاص یک نام مستعار به یک ستون، نام ستون را (در جدول خروجی) تغییر می‌دهد، در حالی که اختصاص یک برچسب، نام ستون را تغییر نمی‌دهد. در این مثال، ستون Area به SquareMiles تغییر نام داده است، و ستون محاسبه شده، Densities نامیده شده است. ستون Name نام خود را حفظ کرده است و برچسب آن Country شده است.

ایجاد جداولی شبیه یک جدول موجود: برای ایجاد یک جدول خالی که دارای ستون‌ها و ویژگی‌های یک جدول یا دید موجود است، از بند LIKE در عبارت CREATE TABLE استفاده کنید.

مثال

```
proc sql;
      create table sql.newcountries
      like sql.countries;
quit;
```

استفاده از گزینه‌های مجموعه‌ی داده‌ها: شما می‌توانید از گزینه‌های مجموعه‌ی داده‌ها در عبارت CREATE TABLE استفاده کنید. مثال زیر جدول countries2 را از روی جدول countries می‌سازد. گزینه‌ی = DROP ستون UNDate را حذف می‌کند، و این ستون در جدول countries2 نخواهد بود.

```
proc sql;
```

```
      create table countries2 as
      select * from sql.countries (drop = UNDate);
quit;
```

اضافه کردن سطرهایی به جداول موجود: از عبارت INSERT برای اضافه کردن مقادیر داده‌ای به جداول استفاده کنید. عبارت INSERT ابتدا یک سطر جدید به جدول موجود اضافه می‌کند، سپس مقادیری را که شما مشخص کرده‌اید در آن سطر وارد می‌کند. شما مقادیر را با استفاده از بند SET یا VALUES مشخص می‌کنید. شما همچنین می‌توانید نتایج یک پرس‌وجو را به یک جدول اضافه کنید.

اضافه کردن سطرها با استفاده از بند SET: با بند SET، شما مقادیر را به ستون‌ها بر حسب نام اختصاص می‌دهید. ستون‌ها می‌توانند به هر ترتیبی در بند SET ظاهر شوند. مثال زیر از بندهای SET برای اضافه کردن دو سطر به جدول Newcountries استفاده می‌کند:

```
proc sql;
      insert into sql.newcountries
      set name='Bangladesh',
          capital='Dhaka',
          population=126391060
      set name='Japan',
      capital='Tokyo',
      population=126352003;
quit;
```

به ویژگی‌های بند SET توجه کنید:

- همانند سایر بندهای SQL، از ویرگول برای جدا کردن ستون‌ها استفاده کنید. به‌علاوه، باید از علامت «;» فقط بعد از آخرین بند SET استفاده کنید.
- اگر داده‌ی یک ستون را مشخص نکنید، مقدار آن ستون، گم‌شده خواهد بود.
- برای مشخص کردن یک مقدار گم‌شده، برای مقادیر نویسه‌ای از فاصله‌ی خالی بین دو علامت نقل قول تکی (' ') و برای مقادیر عددی از نقطه (.) استفاده کنید.

اضافه کردن سطرها با استفاده از بند VALUES: با بند VALUES، مقادیر را بر اساس موقعیت ستون‌ها تخصیص می‌دهید. مثال زیر، از چند بند VALUES برای اضافه کردن سطرهایی به جدول Newcountries استفاده می‌کند. به یاد بیاورید که این جدول دارای ۶ ستون است، بنا بر این لازم است برای هر ۶ ستون این جدول یک مقدار یا مقدار گم‌شده مشخص شود.

```
proc sql;
      insert into sql.newcountries
      values ('Pakistan', 'Islamabad', 123060000, ., ' ', .)
      values ('Nigeria', 'Lagos', 99062000, ., ' ', .);
quit;
```

به ویژگی‌های بند VALUES توجه کنید:

- همانند سایر بندهای SQL، از ویرگول برای جدا کردن ستون‌ها استفاده کنید. به‌علاوه، باید از علامت «;» فقط بعد از آخرین بند VALUES استفاده کنید.
- اگر مقدار یک ستون را مشخص نکنید و مقدار گم‌شده را نیز برای آن در نظر نگیرید، یک پیغام خطا دریافت می‌کنید و آن سطر نیز اضافه نمی‌شود.
- برای مشخص کردن یک مقدار گم‌شده، برای مقادیر نویسه‌ای از فاصله‌ی خالی بین دو علامت نقل قول تکی (' ') و برای مقادیر عددی از نقطه (.) استفاده کنید.

اضافه کردن نتایج یک پرس‌وجو به یک جدول: می‌توانید سطرهای حاصل از نتایج یک جستجو را بـه یـک جدول اضافه کنید. پرس‌وجوی زیر، سطرهای مربوط به کشورهای بزرگ (دارای جمعیت بیش از ۱۳۰ میلیون نفر) را از جدول Countries انتخاب می‌کند. عبارت INSERT ایـن داده‌هـا را بـه جـدول Newcountries اضافه می‌کند:

```
proc sql;
      insert into sql.newcountries
      select * from sql.countries
      where population > 130000000;
quit;
```

اگر پرس‌وجوی شما فاقد د اده‌هایی برای تمام ستون‌ها باشد، یک پیغام خطا دریافت می‌کنید و آن سطرها نیز به جدول اضافه نمی‌شوند.

بهنگام کردن مقادیر داده‌های درون یک جدول

می‌توانید از عبارت UPDATE برای اصلاح داده‌های جـداول و دیـدها اسـتفاده کنیـد. عبـارت UPDATE داده‌ها را در ستون‌های موجود بهنگام می‌کند، این عبارت ستون جدید ایجاد نمی‌کند.

مثال زیر جمعیت تمام کشورها را پنج درصد افزایش می‌دهد:

```
proc sql;
      update sql.newcountries
      set population = population * 1.05;
quit;
```

اگر بخواهید برخی از مقادیر، نـه تمـام مقـادیر، یـک سـتون را بهنگـام کنیـد، بایـد از WHERE در عبـارت UPDATE استفاده کنید. می‌توانید از چندین عبارت UPDATE، که هـر یـک دارای WHERE متفـاوتی است، استفاده کنید. هر چند، هر عبارت UPDATE فقط می‌تواند دارای یک بند WHERE باشد. مثال زیـر برای کشورهای مختلف افزایش‌های جمعیتی مختلفی را در نظر می‌گیرد:

```
proc sql;
      update sql.newcountries
      set population = population * 1.05
      where name like 'B%';

      update sql.newcountries
      set population = population * 1.07
      where name in ('China', 'Russia');
quit;
```

شما می‌توانید نتیجه‌ی بالا را با استفاده از عبارت CASE نیز به‌دست آورید:

```
proc sql;
      update sql.newcountries
      set population = population *
          case
             when name like 'B%' then 1.05
             when name in ('China', 'Russia') then 1.07
             else 1
          end;
quit;
```

تذکر: حتماً از بند ELSE استفاده کنید. اگر این بد را حذف کنید، هر سطری که در هیچ از شرایط موجـود در بندهای WHERE صدق نکند، یک مقدار گم‌شده برای ستون مربوط دریافت می‌کند. دلیل این امر آن اسـت

که عبارت CASE یک مقدار گم‌شده به بند SET برمی‌گرداند، و جمعیت در یک مقدار گم‌شده ضرب می‌شود، که حاصل یک مقدار گم‌شده خواهد بود.

حذف سطرها

برای حذف یک یا چند سطر از یک جدول از عبارت DELETE استفاده کنید. مثال زیر، کشورهایی که نام آن‌ها با R شروع می‌شود را حذف می‌کند.

```
proc sql;
      delete
      from sql.newcountries
      where name like 'R%';
quit;
```

تذکر: اگر از عبارت DELETE بدون بند WHERE استفاده کنید، تمام سطرها حذف می‌شود.

اصلاح ستون‌ها

در جداول موجود، عبارت ALTER TABLE ستون‌هایی را اضافه، اصلاح، یا حذف می‌کند. از این عبارت فقط می‌توانید برای جداول استفاده کنید، این عبارت با دیدها کار نمی‌کند.

اضافه کردن یک ستون: بند ADD یک ستون جدید را به یک جدول موجود اضافه می‌کند. شما باید نام و نوع ستون را مشخص کنید. شما می‌توانید طول (= LENGTH)، فرمت (= FORMAT)، اینفرمت (= INFORMAT)، و برچسب (= LABEL) ستون را نیز مشخص کنید. مثال زیر، ستون عددی Density را به جدول Newcountry اضافه می‌کند:

```
proc sql;
      alter table sql.newcountries
      add density num label = 'Population Density'
          format = 6.2;
quit;
```

ستون جدید به جدول Newcountries اضافه می‌شود، اما هیچ داده‌ای ندارد. عبارت UPDATE زیر مقادیر گم‌شده‌ی ستون Density را به تراکم جمعیتی هر کشور تبدیل می‌کند.

```
proc sql;
      update sql.newcountries
      set density = population / area;
quit;
```

اصلاح یک ستون: از بند MODIFY می‌توانید برای تغییر طول، اینفرمت، فرمت، و برچسب یک ستون استفاده کنید. برای تغییر نام یک ستون از گزینه‌ی مجموعه‌ی داده‌های = RENAME استفاده کنید. شما نمی‌توانید با استفاده از بند MODIFY نوع داده‌ی یک ستون را تغییر دهید.

مثال زیر فرمت ستون Population را تغییر می‌دهد:

```
proc sql;
      alter table sql.newcountries
      modify population format = comma15.;
quit;
```

حذف یک ستون: بند DROP ستون‌هایی را از یک جدول حذف می‌کند. مثال زیر، ستون UNDate را از Newcountries حذف می‌کند:

```
proc sql;
      alter table sql.newcountries
      drop undate;
quit;
```

حذف یک جدول

برای حذف یک جدول، از عبارت DROP TABLE استفاده کنید:

```
proc sql;
      drop table sql.newcountries;
quit;
```

ایجاد و استفاده از دیدها

یک دید حاوی یک پرس‌وجوی ذخیره شده است، و وقتی در یک شیوه‌ی SAS یا گام داده‌ای مورد استفاده قرار گیرند، اجرا می‌شوند. شیوه‌ها به دلایل زیر سودمند هستند:

- باعث صرفه‌جویی در فضا می‌شوند، چون یک دید در مقایسه با جدولی که داده‌هایش را از آن می‌خواند بسیار کوچک‌تر است.
- کاربران را از اجرای پی در پی پرس‌وجوها برای کنار گذاشتن ستون‌ها یا سطرهای ناخواسته بی‌نیاز می‌سازد.
- از داده‌های حساس یا محرمانه در برابر کاربران محافظت می‌کند در عین حالی که اجازه‌ی دیدن سایر ستون‌های همان جدول را به همان کاربران می‌دهد.
- اطمینان می‌دهند که مجموعه‌های داده‌های ورودی همیشه جاری هستند، چون داده‌ها در زمان اجرا از جداول استخراج می‌شوند.
- اتصال‌ها یا پرس‌وجوهای پیچیده را از کاربران پنهان می‌کنند.

ایجاد دیدها: از عبارت CREATE VIEW برای ایجاد دید استفاده کنید، همانند مثال زیر:

```
proc sql;
      create view sql.newcontinents as
      select continent, sum(population) as totpop,
             sum(area) as totarea
      from sql.countries
      group by continent;
quit;
```

حذف یک دید: از عبارت DROP VIEW برای حذف یک دید استفاده کنید:

```
proc sql;
      drop view sql.newcontinents;
quit;
```

تعیین دیدهای در-خط: در برخی حالات، ممکن است بخواهید در بند FROM، به جای یک جدول یا دید از یک پرس‌وجو استفاده کنید. شما می‌توانید یک دید ایجاد کنید و در بند FROM به آن ارجاع دهید، اما این فرایند مستلزم اجرای دو گام است. برای صرفه‌جویی در گام اضافی، دید را به‌صورت در-خط، که بین دو پرانتز محصور شده است، در بند FROM مشخص کنید.

یک دید در-خط پرس‌وجویی است که در بند FROM ظاهر می‌شود. یک دید در-خط، یک جدول را به‌صورت داخلی تولید می‌کند که پرس‌وجوی بیرونی از آن برای انتخاب داده‌ها استفاده می‌کند. بر خلاف دیدها که با عبارت CREATE VIEW ایجاد می‌شوند، نامی به دیدهای در-خط اختصاص نمی‌یابد، و نمی‌توان در سایر پرس‌وجوها یا شیوه‌های SAS به آن‌ها ارجاع داد. به یک دید بر-خط فقط در داخل پرس‌وجویی که آن را تعریف کرده است می‌توان ارجاع داد.

در پرس‌وجوی زیر، مجموع جمعیت تمام کشورهای کارائیب و آمریکای مرکزی را در یک دید در-خط محاسبه می‌کند. بند WHERE این مجموع را با جمعیت هر کشور مقایسه می‌کند. فقط کشورهایی که دارای جمعیتی بیش از مجموع جمعیت کشورهای کارائیب و آمریکای مرکزی دارند نمایش می‌یابد.

```
proc sql;
      select w.Name, w.Population, c.TotCarib
      from (select sum(population) as TotCarib format=comma15.
            from sql.countries
            where continent = 'Central America and Caribbean')
as c,
           sql.countries as w
      where w.population > c.TotCarib;
quit;
```