بسمه تعالی

روش‌های آماری و تحلیل داده‌ها با نرم‌افزار SAS

نویسندگان:

حمیدرضا نواب‌پور

آسیه عباسی

حسن رنجی

مهر ۱۳۸۹

فصل ۱: آشنایی با نرم‌افزار SAS

۱-۱ آشنایی

۱-۲ فعال کردن سامانه‌ی SAS
۱-۲-۱ پنجره‌ی Editor
۱-۲-۲ پنجره‌ی Log
۱-۲-۳ پنجره‌ی Output
۱-۲-۴ پنجره‌های Explorer و Results
۱-۲-۵ پنجره‌ی Graph

۱-۳ قاعده‌ها و شکل دستوری پایه‌ای SAS

۱-۴ گام‌های برنامه‌ای SAS
۱-۴-۱ گام DATA
۱-۴-۲ گام PROC

۱-۵ گام DATA
۱-۵-۱ عبارت‌هایی که با گام DATA به کار می‌روند
۱-۵-۱-۱ عبارت INPUT
۱-۵-۱-۲ عبارت‌های انتساب
۱-۵-۱-۳ اجراهای شرطی
۱-۵-۱-۴ محاسبه‌های مکرر
۱-۵-۲ تابع‌های SAS

۱-۶ استفاده از نویسه‌های ویژه در عبارت INPUT
۱-۶-۱ نشانگرهای کنترلی
۱-۶-۲ مشخص‌گر سطرنگهدار @
۱-۶-۳ مشخص‌گر خطنگهدار @@
۱-۶-۴ نشانگر کنترل کننده‌ی سطر

۱-۷ مجموعه‌ی داده‌های دائمی SAS
۱-۷-۱ ساختن کتابخانه
۱-۷-۲ ایجاد مجموعه‌ی داده‌های دائمی SAS
۱-۷-۳ عبارت INFILE

۱-۸ گام PROC

فصل اول
آشنایی با نرم‌افزار SAS

۱-۱ آشنایی

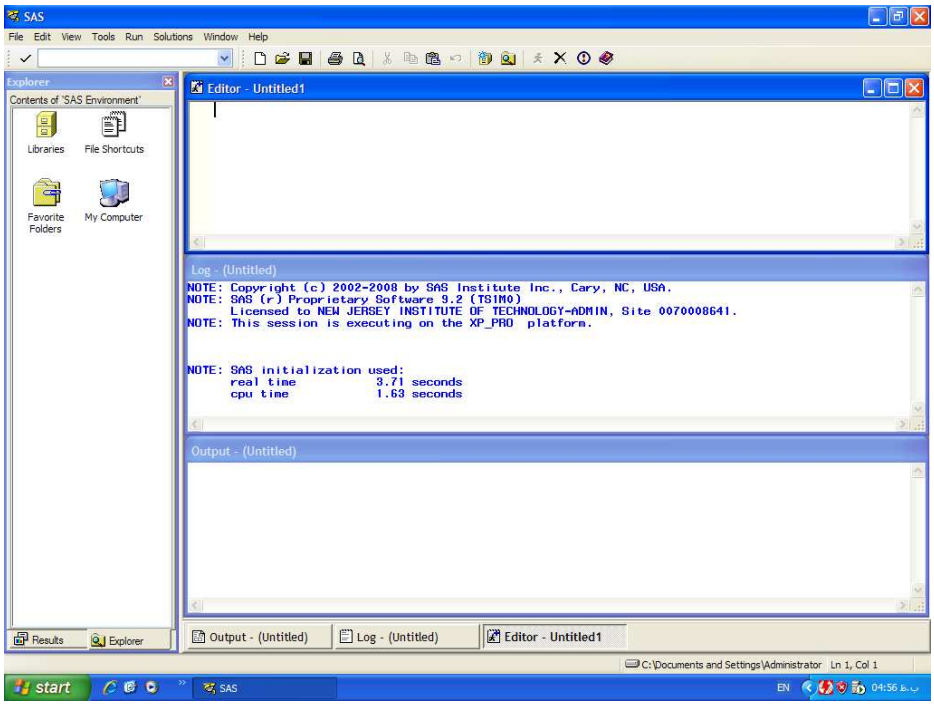
سامانه‌ی تحلیل آماری (SAS)[1] یک بسته‌ی برنامه‌ای رایانه‌ای برای انجام تحلیل‌های آماری داده‌ها است. این سامانه دارای مجموعه‌ای از شیوه‌ها برای تحلیل داده‌ها و توانایی کار با داده‌های ورودی/خروجی است. این سامانه به کاربر این امکان را می‌دهد تا برای تحلیل‌های آماری داده‌ها عبارت‌های برنامه‌ای خود را بنویسد، همچنین روال‌های SAS را که شیوه‌ها نام دارند، فراخوانی کند. عبارت‌های برنامه‌ای کاربر[5] نوشته 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 (ورودی) برای تحلیل‌های آماری موردنظر کاربر به کار می‌رود. سامانه‌ی SAS هر تحلیل آماری که کاربر به درستی مشخص کرده باشد را انجام می‌دهد.

اکثر تحلیل‌های آماری در SAS نیاز به دانستن دانش وسیع از ویژگی‌های این سامانه ندارد. اما، حتی تحلیل‌های ساده‌ی آماری از برخی توانایی‌های زبان برنامه‌ای این سامانه استفاده می‌کند. بنا بر این، دانستن تعدادی از ساختارهای برنامه‌ای SAS و نحوه‌ی کارکرد آن برای نوشتن برنامه‌های کارای SAS لازم است.

۱-۲ فعال کردن سامانه‌ی SAS

اگر SAS در محیط ویندوز نصب شده باشد، نشانه‌ای در Desktop به صورت یک منشور با کوته‌نوشته‌ی SAS وجود دارد. برای فعال کردن SAS کافی است که نشانگر را روی این علامت برده و دوبار کلیک کنید. پس از ورود به سامانه‌ی SAS، پنج پنجره قابل تشخیص است: پنجره‌ی Editor، پنجره‌ی Log، پنجره‌ی Output، پنجره‌ی Explorer، و پنجره‌ی Results. این پنجره‌ها در شکل ۱-۱ دیده می‌شوند.

[1] Statistical Analysis System



شکل ۱-۱ پنجره‌های سامانه‌ی SAS در اولین بار فعال‌سازی

۱-۲-۱ پنجره‌ی Editor

این پنجره، Enhanced Editor نیز خوانده می‌شود. پنجره Editor، مکانی است که در آن برنامه‌های SAS نوشته، اصلاح و اجرا می‌شوند. در این پنجره برنامه‌های SAS، به صورت متنی وارد می‌شوند. بنا بر این، برنامه‌های SAS قابل کپی شدن در واژه‌پردازهایی مثل Word و برعکس هستند. وقتی برنامه‌ای در SAS اجرا می‌شود، ابتدا برنامه‌ی متنی که در قالب ASCII[۲] است تبدیل به کدهای SAS شده و بعد فرمان‌ها اجرا می‌شوند. برنامه‌های SAS با قالب ASCII متنی ذخیره می‌شوند.

۱-۲-۲ پنجره‌ی Log

این پنجره رابطی بین کاربر و سامانه‌ی SAS است. زمانی که برنامه‌ای اجرا می‌شود، جزئیات اجرا در پنجره‌ی Log نوشته می‌شود. ابتدا قسمت‌هایی از برنامه که اجرا می‌شوند آورده می‌شود و سپس مدت زمان اجرای برنامه برای هر گام نمایش می‌یابد. در صورتی که برنامه خطا داشته باشد، SAS خطاها را با رنگ قرمز نشان داده و برای تصحیح برنامه به صورت محدود راهنمایی می‌کند. گاهی ممکن است برخی از دستورها به اشتباه تایپ شده باشند، در این‌گونه وضعیت‌ها SAS برنامه را اجرا می‌کند ولی برای این اشتباه‌ها هشدار می‌دهد. بنا بر این، لازم است که کاربر در اولین باری که برنامه اجرا می‌شود، پنجره‌ی Log را به دقت وارسی کند.

۱-۲-۳ پنجره‌ی Output

[2] American Standard Code for Information Interchange

زمانی که برنامه‌ای در سامانه‌ی SAS بدون خطا اجرا می‌شود، خروجی آن در پنجره‌ی Output نمایش مـی‌یابـد. خروجـی SAS می‌تواند به قالب‌های متفاوتی ساخته شود. پیش‌فرض، خروجی فهرستی است. خروجی به صورت PDF و HTML نیز قابـل ارائـه است.

۱-۲-٤ پنجره‌های Explorer و Results

وقتی SAS فعال می‌شود پنجره‌ای در سمت چپ صفحه‌ی نمایش دیده می‌شود. در انتهای این پنجره دو پـرش‌گـاه Explorer و Results دیده می‌شود. با کلیک کردن روی هر پرش‌گاه، پنجـره‌ی مربـوط بـاز مـی‌شـود. پنجـره‌ی Explorer، کتابخانـه‌هـای SAS (Library) را که قابل دسترس هستند، نشان می‌دهد. در این پنجره می‌توان کتابخانه‌ی جدید ساخت. فایل‌های داده‌ای کـه در این کتابخانه‌ها دخیره می‌شوند، دائمی هستند. به این معنا که کاربر همواره قادر به استفاده از فایل‌های داده‌ای است. کتابخانه‌ی Work که جزء سامانه است، شامل این اصل نمی‌شود. به عبارت دیگر فایل‌های داده‌ای که در این کتابخانه ساخته می‌شود، موقت هستند. به این معنا که با خروج از سامانه‌ی SAS به طور خودکار تمام فایل‌های داده‌ای این کتابخانه پـاک مـی‌شـوند. زمـانی کـه برنامه‌ای اجرا می‌شود، خروجی برنامه در پنجره‌ی Output آورده می‌شـود و در پنجـره‌ی Results، فهرسـتی از اجـرای برنامـه شامل گام‌ها، عبارت‌های موجود در گام‌ها آورده می‌شود. کاربر می‌تواند با کلیک کـردن روی هـر عبـارت، خروجـی آن قسـمت از برنامه را در پنجره‌ی Output فراخوانی کند.

۱-۲-۵ پنجره‌ی Graph

اگر برنامه‌ای خروجی نموداری داشته باشد، این نمودارها به طور پیش‌فرض در پنجره‌ای به نام Graph نمایش داده می‌شوند. ایـن پنجره نمودارها را با وضوح بالا و به ترتیبی که ایجاد می‌شوند در خود جا می‌دهد. برای دیدن نمودارها روی پرش‌گاه Graph کـه در پایین صفحه‌ی نمایش دیده می‌شود کلیک کنید. حرکت از پنجره‌ای به پنجره‌ی دیگر از طریق کلیـک کـردن روی پـرش‌گـاه مربوط در پایین صفحه‌ی نمایش امکان‌پذیر است.

۱-۳ قاعده‌ها و شکل دستوری پایه‌ای SAS

مقدار داده‌ای

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

چند مثال از مقدارهای نویسه‌ای عبارت‌اند از:

y5, 789, south west, do, a

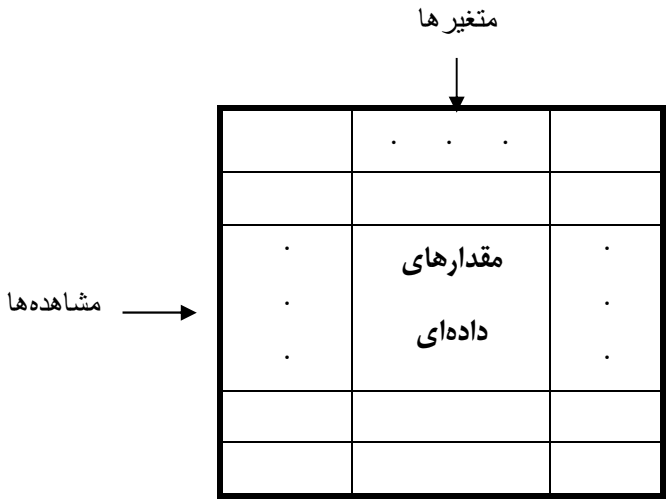
مقدار عددی، یک عدد با و یا بدون نقطه‌ی اعشاری است که قبل از آن می‌تواند علامت به‌علاوه یا منها قرار گیرد ولی نمـی‌توانـد شامل واوک (,) باشد. مثال‌هایی برای مقدار عددی عبارت‌اند از:

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

سامانه‌ی SAS همانند دیگر بسته‌های نرم‌افزاری، مجموعه‌ی داده‌های خام را نمی‌شناسد. بنا بر این برای کار با داده‌ها، ابتدا باید مجموعه‌ی داده‌های خام به مجموعه‌ی داده‌های SAS تبدیل شود. هم مجموعه‌ی داده‌های خام و هم 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.



شکل ۱-۲ نمایشی از ماتریس داده‌ها

به منظور تبدیل مجموعه‌ی داده‌های خام به مجموعه‌ی داده‌های SAS، مجموعه‌ی داده‌ها باید به صورت شکل ۱-۲ در برنامه یا در فایل ذخیره‌شده به SAS معرفی شوند.

متغیرها

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

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. اما، هر مشاهده در مجموعه‌ی داده‌های SAS می‌تواند شامل یک سطر یا بیشتر از درون‌داده‌ها باشد.

نام‌ها

برای تحلیل آماری داده‌ها باید به متغیرها و مجموعه‌ی داده‌ها نامی اختصاص داده شود. عبارت متغیر ناظر بر اطلاعاتی است که برای هر مشاهده گرداوری می‌شود. نام‌ها باید با یک حرف یا خط زیرین (_) شروع شوند و می‌توانند شامل حروف، اعداد یا – 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

نکته‌ی ۱-۱ 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 نام‌های از این گونه از قالب زیر استفاده کنید

```
'variable name'n
```

در نوشتن عبارت‌های برنامه‌ای بهتر است از نام‌هایی با طول کوتاه استفاده کرد و سپس به آن‌ها برچسب داد. کوتاه‌نویسی هم سرعت را زیادتر می‌کند و هم خطا را کم‌تر.

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. کاربر می‌تواند فهرستی دنباله‌ای از نام متغیرها به صورت AAAX–AAAY که AAA نویسه‌ها و X وY عدد هستند. برای مثال می توان نام متغیرهای Y1 Y2 Y3 Y4 Y5 را که ممکن است در عبارت های SAS ظاهر شوند به‌صورت Y1-Y5 نوشت.

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.

عبارت‌های SAS

عبارت‌های SAS دستورهایی هستند که کاربر توسط آن‌ها به سامانه‌ی SAS فرمان عملیاتی می‌دهد. معمولاً عبارت‌های SAS با یک کلیدواژه شروع می‌شود. در این کتاب کلیدواژه‌ها با حرف‌های بزرگ و سیاه‌شده نشان داده می‌شوند. کلمه‌ها با حرف‌های کوچک در عبارت‌های SAS، صورت عمومی عبارت را نشان می‌دهند.

مثلاً صورت عمومی عبارت KEEP به صورت زیر است:

```
KEEP Var1-Vark;
```

در عبارت بالا KEEP یک کلیدواژه است و پس از آن نام متغیرها (k متغیر) آورده شده است. این عبارت متغیرهای VAR1 تا VARk را در مجموعه‌ی داده‌های SAS حفظ کرده و بقیه را حذف می‌کند.

در عبارت‌های SASی که از این پس معرفی می‌شوند، گزینه‌ها داخل دو زاویه < > قرار داده می‌شوند. استفاده از گزینه‌ها در عبارت‌های SAS اختیاری است. این شیوه‌ای است که در کتاب‌های راهنمای SAS به کار گرفته شده است. چند مثال از گزینه‌ها به دنبال آورده می‌شوند:

```
OUTPUT  <data-set-name(s)>;
FILENAME fileref <device-type> <options>
         <operating-environment-options>;
```

شکل دستوری عبارت‌های SAS

چند قاعده برای نوشتن عبارت‌های SAS عبارت‌اند از:

۱- عبارت‌های SAS از هر ستون می‌توانند شروع شده و در هر ستون خاتمه یابند.

۲- عبارت‌های SAS با یک نقطه واوک (;) خاتمه می‌یابند.

۳- بیش از یک عبارت می‌توانند در یک سطر نوشته شوند.

۴- عبارت‌های SAS می‌توانند از هر ستونی در یک سطر شروع شده و در چند سطر ادامه یابند.

۵- اجزای یک عبارت SAS باید با حداقل یک فاصله از هم جدا شوند. اگر اجزای یک عبارت با نمادهایی مثل +، -، *، /، و یا = بهم مرتبط شوند، گذاشتن فاصله بین آن الزامی نیست. مثلاً تمام صورت‌های x=y، x = y، x= y، و x =y قابل قبول هستند.

۶- سامانه‌ی SAS نسبت به حرف‌های بزرگ و کوچک حساس نیست. برای مثال در شکل دستوری بالا عمل‌گر OUTPUT، output، و Output یکسان است.

داده‌های گم‌شده

مقدار گم‌شده در خطهای داده‌ای ورودی با یک فاصله و یا یک نقطه مشخص می‌شود. داده‌ی عددی گم‌شده در مجموعه‌ی داده‌های SAS با یک نقطه نشان داده می‌شود. مقدار نویسه‌ای گم‌شده در خطهای ورودی با فاصله مشخص می‌شود.
داده‌ی گم‌شده می‌تواند در عمل‌گرهای مقایسه‌ای استفاده شود. برای مثال، عمل‌گر مقایسه‌ای زیر بررسی می‌کند که اگر مقدار متغیر age برای هر مشاهده گم‌شده باشد، آن مشاهده از مجموعه‌ی داده‌ها حذف شود.

```
IF age = . THEN DELETE;
```

۱-٤ گام‌های برنامه‌ای SAS

۱-٤-۱ 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.

۳- اصلاح داده‌ها قبل از اینکه به مجموعه‌ی داده‌ها وارد شوند. این کار با عبارت‌های برنامه‌ای که توسط کاربر نوشته می‌شود، انجام می‌گیرد.

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

```
DATA <نام‌ها>;
```

۱-٤-۲ 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;
```

۱-۵ گام DATA

برای ایجاد مجموعه‌ی داده‌های SAS، اصلاح کردن متغیرها و مشاهده‌ها، ساختن متغیر جدید، ساختن زیرمجموعه‌ای از مجموعه‌ی داده‌های ورودی و ... از گام DATA استفاده می‌شود. گام DATA با عبارت DATA شروع شده و با عبارت DATALINES یا CARDS خاتمه می‌یابد. مثلاً عبارت مقایسه‌ای

```
IF exp < 1.0 and ostan = 'TEH';
```

زیرمجموعه‌ای از مشاهده‌ها را ایجاد می‌کند.

مثال ۱-۱ در زیر یک گام DATA ساده برای ایجاد یک مجموعه‌ی داده‌های SAS نوشته شده است:

```
DATA exam;
   INPUT id 1-2 gender $ midterm 6-8 final 9-11 hw 2.1;
   DATALINES;
11 M 85 91 21
6  F 54 60 20
8  F 60 65 22
19 M 70 72 25
28 M 65 60 22
12 F 90 92 27
30 M 80 85 28
26 F 77 80 26
;
RUN;

PROC PRINT;
RUN;
```

گام DATAی بالا مجموعه‌ی داده‌های SAS موقتی به نام exam ایجاد می‌کند. عبارت DATA نام exam را به مجموعه‌ی داده‌های ایجاد شده منتسب می‌کند. عبارت INPUT دو اطلاع ارائه می‌کند: نام متغیرها و موقعیت مقدار متغیرها در خط‌های داده‌ای. ملاحظه می‌شود که هر عبارت با یک کلیدواژه شروع و با یک نقطه‌واوک (;) خاتمه می‌یابد. عبارت قبل از RUN یعنی ";" عبارت پوچ نامیده می‌شود و باید در یک سطر جدا آورده شود. اگر عبارت ";" بعد از مقدار ۲۶ به صورت ;26 آورده شود، SAS آن را خطا گرفته و پردازش را متوقف می‌کند. آوردن عبارت RUN; در انتهای هر گام از SAS می‌خواهد که عبارت‌های

بالای RUN، اجرا شود. هر برنامه می‌تواند فقط شامل یک عبارت RUN در انتهای برنامه باشد و یـا شـامل چنـد عبـارت RUN. توصیه می‌شود که بعد از هر گام برنامه‌ای عبارت RUN نوشته شود.

اجرای برنامه‌ی بالا یک مجموعه‌ی داده‌های SAS به نام exam می‌سازد که در کتابخانه‌ی work نرم‌افزار SAS به طور خودکار ذخیره می‌شود. این مجموعه‌ی داده‌ها تا وقتی که کاربر در محیط SAS است، قابل دسترس می‌باشد. ولـی بـا خـروج از نـرم‌افـزار SAS، این مجموعه‌ی داده‌ها به طور خودکار پاک می‌شود. به همین دلیل این مجموعه‌ی داده‌ها را موقتی می‌گویند.

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 می‌کنـد. مقـدار ایـن متغیـر در صـورت وجـود مشـاهده 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 مشـخص‌شـده (۱۰) متوقف می‌کند.
(RENAME = ())	(RENAME = (A = D))	نام متغیر A را به D تغییر می‌دهد. توجه داشـته باشـید کـه سـایر گزینه‌ها در صورت استفاده به همراه این گزینه، بایـد از نـام‌هـای اولیه‌ی متغیرها، و نه نام‌های جدید، استفاده کنند.
(WHERE = ())	(WHERE = (A >= 10))	فقط مشاهداتی را در نظر می‌گیـرد کـه مقـدار متغیـر A در آن‌هـا 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

۱-۵-۱ عبارت‌هایی که با گام DATA به کار می‌روند

۱-۵-۱-۱ عبارت INPUT

عبارت INPUT ترتیب قرار گرفتن متغیرهای داده‌ای در هر سطر مجموعه‌ی داده‌ها را بیان می‌کند. سامانه‌ی SAS از اطلاعات ارائه شده در عبارت INPUT استفاده کرده و با خواندن مقدارهای داده‌ای برای هر یک از متغیرهایی که در عبارت INPUT فهرست شده‌اند، مشاهده‌ها را در مجموعه‌ی داده‌های SASی که تولید می‌شود، ایجاد می‌کند.

روش‌های متعددی برای ورود مقدارهای داده‌ای متغیرها برای ایجاد مجموعه‌ی داده‌ها وجود دارند. سه روش از آن‌ها به دنبال تشریح می‌شوند.

INPUT فهرستی

در این روش نام متغیرها در عبارت INPUT مقدارهای داده‌ای حداقل با یک فاصله از یکدیگر جدا می‌شوند. مثالی از این روش عبارت است از:

```
INPUT age weight height;
```

در این مثال، SAS اولین مقدار از هر خط داده‌ها را به اولین متغیر، age، دومی را به دومین متغیر، weight، و سومی را به سومین متغیر، height، منتسب می‌کند. هنگامی‌که از INPUT فهرستی استفاده می‌شود مقدارهای داده‌ای در خطهای داده‌ها نیز باید با حداقل یک فاصله از هم جدا شوند.

از INPUT فهرستی می‌توان هم برای خواندن مقدارهای عددی و هم نویسه‌ای استفاده کرد. برای مشخص کردن متغیرهای نویسه‌ای در INPUT فهرستی از نماد $ در جلوی نام متغیر استفاده می‌شود. برای مثال وقتی INPUT فهرستی به صورت

```
INPUT  ostan $ exp act;
```

باشد، SAS فرض می‌کند که متغیر ostan شامل مقدارهای نویسه‌ای و متغیرهای exp و act شامل مقدارهای عددی هستند. سامانه‌ی SAS به‌صورت پیش‌فرض، ماکسیمم طولی به اندازه‌ی ۸ نویسه (بایت) به مقدارهای داده‌ای می‌دهد. اگر مقدار داده‌ای کمتر از ۸ نویسه طول داشته باشد آن را از سمت راست نوشته و باقی ستون‌ها را خالی می‌گذارد. متغیرهای نویسه‌ای که شامل مقدارهایی با بیش از ۸ نویسه هستند را می‌توان با استفاده از روش INPUT فورمت‌شده به SAS معرفی کرد.

به‌هنگام کاربست INPUT فهرستی، اگر SAS مقداری برای متغیر در سطر جاری نیابد، به سطر داده‌های بعدی رفته و مقدار داده‌ای برای متغیر اختیار می‌کند. به این منظور، هنگام کاربست INPUT فهرستی، داده‌های گم‌شده حتما باید با یک نقطه (.) به همراه یک ستون خالی در دو طرف نقطه مشخص شوند. در INPUT فهرستی نمی‌توان از فاصله‌ی خالی برای مشخص‌سازی داده‌ی گم‌شده استفاده کرد.

INPUT فورمت‌شده

روش INPUT فهرستی ممکن است برای بسیاری از مجموعه‌ی داده‌هایی که در برنامه آورده می‌شوند و یا از رسانه‌های ذخیره‌ساز مثل دیسک یا CD خوانده می‌شوند، مناسب نباشد. زیرا مقدارهای داده‌ای به منظور صرفه‌جویی در فضا ممکن است به‌طور متوالی و بدون فاصله داده‌آمایی شده باشند و یا ممکن است مقدارهای داده‌ای با واوک از یکدیگر جدا شده باشند. در این حالت‌ها باید از اینفورمت‌های SAS برای خواندن داده‌ها استفاده کرد.

از اینفورمت‌ها می‌توان برای خواندن داده‌هایی که به هر صورت در خطوط داده‌ها وارد شده باشند، استفاده کرد. اینفورمت‌ها اطلاعاتی نظیر تعداد ستون‌هایی که یک مقدار داده‌ای اشغال کرده، چگونه مقدار داده‌ای خوانده شود، و اینکه چگونه مقدار داده‌ای در مجموعه‌ی داده‌های SAS ذخیره شود را ارائه می‌کنند.

برای خواندن مقدارهای داده‌ای از خط‌های داده‌ای، کاربر باید ستونی که مقدار داده‌ای از آن شروع می‌شود، مقدار ستون‌هایی که باید خوانده شوند، این‌که مقدار داده‌ای عددی است و یا نویسه‌ای و نقطه‌ی اعشاری از مقدارهای عددی کجا باید قرار داده شود را مشخص کند.

برای مشخص کردن اینکه یک مقدار داده‌ای از چه ستونی خوانده شود، از نویسه‌ی ویژه‌ی "@" و شماره‌ی ستون که به دنبال آن می‌آید و این دو قبل از نام متغیر می‌آیند، استفاده می‌شود. مثلاً

```
INPUT @10 age @20 weight @36 height;
```

به SAS می‌گوید مقدار متغیر age از ستون ۱۰، مقدار weight از ستون ۲۰ و مقدار height از ستون ۳۶ شروع می‌شوند. در اینجا فرض شده است که مقدارهای داده‌ای با یک فاصله خالی از هم جدا شده‌اند، در غیر این‌صورت باید از اینفورمتی استفاده شود که به دنبال معرفی می‌شود.

برای متغیرهای عددی، اینفورمت "W." مشخص می‌کند که W ستون که از ستون جاری شروع می‌شود به عنوان مقدار متغیر باید خوانده می‌شود. مقدار W باید عدد صحیح و مثبت باشد. برای مثال

```
INPUT @20 weight 3.;
```

به SAS می‌گوید که به ستون ۲۰ رفته و سه ستونی که از ستون ۲۰ شروع می‌شوند را خوانده و به عنوان مقدار متغیر weight در مجموعه‌ی داده‌های SAS ذخیره کند.

اینفورمت "W.d" به SAS می‌گوید که مقدار متغیر را به صورت بالا خوانده، سپس نقطه‌ی اعشاری را قبل از "d"رقم آخر قرار دهد. مثلاً

```
INPUT @10 age 6.2;
```

به SAS می‌گوید به ستون ۱۰ رفته و ۶ ستون بعدی (شامل ستون ۱۰) را به عنوان مقدار متغیر age خوانده و نقطه‌ی اعشاری را قبل از ۲ رقم آخر قرار دهد. اگر مقدار داده‌ای شامل نقطه‌ی اعشاری باشد، SAS از مشخص‌سازی که در اینفورمت صورت گرفته است، صرف‌نظر می‌کند. در صورت اخیر "W" در "W.d" باید شامل نقطه‌ی اعشاری باشد.

برای متغیر نویسه‌ای، اینفورمت "$W." به SAS می‌گوید از ستون جاری W ستون بعدی را به عنوان مقدار نویسه‌ای بخواند. برای مثال

```
INPUT @15 name $45.;
```

به SAS می‌گوید که ستون‌های ۱۵ الی ۶۹ را به عنوان مقدار متغیر نویسه‌ای name بخواند. اگر کاربر بخواهد که متغیر name شامل فاصله‌ی خالی باشد (مثلا فاصله بین نام کوچک و نام فامیلی)، می‌تواند از اینفورمت "$CHARW." استفاده کند.

فرض کنید که یک خط داده‌ای شامل مقدارهای زیر باشد.

0001TEH045507205

که ۰۰۰۱ شماره‌ی خانوار نمونه‌ای، TEH نمایانگر استان محل سکونت خانوار نمونه‌ای، ۴/۵۵مجموع هزینه‌های خانوار در ماه گذشته، ۰/۷۲ درصد هزینه‌های خوراکی خانوار، و ۵ بعد خانوار است. گیرید، id، ostan، expenditure، percent و member نام متغیرهای متناظر باشند. یک عبارت INPUT مناسب برای خواندن مقدارهای داده‌ای به صورت زیر است:

```
INPUT id 4. ostan $3. expenditure 4.2 percent 3.2 member 2.;
```

فرض کنید که خطهای داده‌ای دارای صورت زیر باشند.

0001XXXXTEH155X072XXXXXX1390

که Xها ستون‌های داده‌ای هستند که مایل نیستیم در تحلیل مشارکت کنند. برای پریدن از این ستون‌ها می‌توان از نماد "@" و یا نماد "+" استفاده کرد. برای مثال بعد از خواندن مقدار متغیر id مقدار متغیر ostan از ستون ۹ با استفاده از نماد "@9" خوانده شده و بعد مقدار متغیر exp خوانده می‌شود و سپس از یک ستون با "+1" رد می‌شود و متغیرهای percent و year خوانده می‌شود. عبارت INPUT زیر این شیوه‌ی خواندن را نشان می‌دهد.

```
INPUT id 4. @9 ostan $3. exp 3.2 +1 percent 3.2 @25 year 4.;
```

نمادهای "@" ، "+" که در عبارت INPUT به کار می‌روند را نشانگرهای کنترلی می‌گویند.

نام متغیرها و اینفورمت‌هایی که در عبارت INPUT به کار می‌روند را می‌توان در دو فهرست مجزا و درون دو کمان گروه‌بندی کرد. عبارت INPUT بالا را می‌توان به صورت زیر نوشت:

```
INPUT (id ostan exp percent year) (4. @9 $3. 3.2 +1 3.2 @25 4.);
```

در این فهرست دنباله‌ای هر اینفورمت و یا ترکیب اینفورمت–نشانگر کنترلی مرتبط با یک نام متغیر است.

INPUT ستونی

وقتی مقدار داده‌ها با حداقل یک فاصله(ی خالی) از هم جدا نشده باشند و کاربر ترجیح دهد که از اینفورمت استفاده نکند، INPUT ستونی بدیل دیگری برای INPUT فهرستی است. در این حالت مقدارها باید ستون‌های یکسان را در تمام خطهای داده‌ای اشغال کنند. در این روش به دنبال نام متغیرها در عبارت INPUT شماره‌ی ستون‌هایی که آن متغیر اشغال می‌کند به صورت عدد شروع – عدد خاتمه آورده می‌شود.

برای مثال، فرض کنید خطهای داده‌ای به صورت زیر باشند

0001TEH1.55 ৳ 721390

در این صورت INPUT ستونی به صورت زیر نوشته می‌شود:

```
INPUT id 1-4 ostan $5-7 exp 8-11 .2 percent 12-14 .2 year 15-18;
```

عبارت بالا مقدار متغیر id را از ستون‌های ۱ تا ۴ به صورت عدد صحیح، مقدار متغیر ostan را به صورت رشته‌ی نویسه‌ای از دو ستون بعد می‌خواند. در این عبارت 2. به SAS می‌گوید که موقع خواندن مقدار متغیر exp، نقطه‌ی اعشاری کجا باید قرار گیرد. برای مثال از این خط داده‌ای مقدار متغیر percent ۰/۷۲ خوانده می‌شود.

ترکیب روش‌های مختلف

عبارت INPUT می‌تواند ترکیبی از روش‌های بالا را شامل شود. برای مثال خط داده‌ای زیر را در نظر بگیرید.

0001TEH1.55 721390

با ترکیب INPUT فهرستی، فورمت‌شده، و ستونی می‌توان عبارت INPUT زیر را برای خط داده‌های بالا به صورت زیر نوشت:

```
INPUT id 1-4 ostan $3. exp percent 2.2 year 15-18;
```

در این حالت، SAS از INPUT ستونی برای خواندن مقدار متغیر id، از INPUT فورمت‌شده برای خواندن مقدار متغیر ostan، از INPUT فهرستی برای خواندن مقدار متغیر exp استفاده می‌کند. در این‌جا بعد از خوانده شدن مقدار متغیر exp، نشانگر به ستون ۱۳ برای خواندن متغیر percent می‌رود. زیرا یک فاصله باید بین مقدار متغیر exp و percent باشد، بنا بر این پهنای فیلد مربوط به متغیر percent باید ۲ باشد.

۱-۵-۱-۲ عبارت‌های انتساب

ساده‌ترین راه برای منتسب کردن فورمت به یک متغیر این است که در عبارت PUT فورمت موردنظر به دنبال نام متغیر آورده شود. در مثال زیر فورمت .DOLLAR برای نوشتن مقادیر عددی برحسب دلار به‌کار رفته است.

```
DATA money;
   Amount=1412.55;
   PUT amount DOLLAR10.2;
RUN;
```

عبارت‌های بالا مقدار $1,412.55 را تولید کرده و در پنجره Log چاپ می‌کند.

در فورمت DOLLAR10.2 مقادیر ۱۰ و ۲ نمایانگر این هستند که حداکثر ۱۰ ستون برای مقدار داده وجود دارد که ۲ ستون آن مربوط به قسمت اعشاری است و از ۸ ستون دیگر یک ستون برای نقطه اعشاری، یک ستون برای واوک، یک ستون برای علامت دلار، چهار ستون برای قسمت صحیح مقدار داده و (اگر عدد منفی باشد) یک ستون برای علامت منفی نگه‌داشته شده است.

مثال ۱-۲ فرض کنید مقدار NETPAY برابر ۵۷۲۱/۶۵ دلار است و می‌خواهیم آن را در ستونهای ۱۵ تا ۲۴ خط خروجی چاپ کنیم. عبارت زیر مقدار $5,721.65 را چاپ می‌کند.

```
PUT  @15  netpay  DOLLAR10.2;
```

روش دیگر انتساب برای ایجاد متغیر جدید و تغییر مقدار داده‌های موجود به صورت زیر است.

نام متغیر = عبارت محاسباتی;

متغیرهای جدید را می‌توان با ترکیب یک یا چند متغیر موجود از طریق عبارت‌های محاسباتی ایجاد کرد. این‌کار با استفاده‌ی ترکیبی از عمل‌گرهای ریاضی، تابع‌های SAS و سایر عبارت‌های محاسباتی درون دو کمان و انتساب نتیجه‌ی آن به یک متغیر جدید، انجام می‌شود.

مثال ۱-۳ گام DATAی زیر شامل چند صورت از عبارت‌های انتساب است.

```
DATA exp1_3;
   INPUT (x1-x7)(@5 3*5.1 4*6.2);
   y1=x1+x2**2;
   y2=SQRT(x3);
   y3=LOG(x7);
   x5=ABS(x5);
   DATALINES;
```

خط‌های داده‌ها

```
;
RUN;
```

گام DATAی بالا سه متغیر جدید y1، y2، و y3 را ایجاد می‌کند. مقدار y1 برای هر مشاهده، حاصل‌جمع مقدار x1 و تـوان دوم مقدار متغیر x2ی آن مشاهده است. مقدار y2 برای هر مشاهده ریشه‌ی دوم مقدار x3 و مقدار y3 لگاریتم طبیعـی مقـدار x7 آن مشاهده است. در این گام DATA، هر مقدار متغیر x5 با قدرمطلق آن برای هر مشاهده جایگزین می‌شود. در گام DATAی بـالا، نمادهای SQRT، LOG، و ABS تابع‌های SAS هستند که شناسه‌های این تابع‌ها در داخل دو کمان آورده شده‌اند.

عبارت‌های ریاضی مطابق با قاعده‌هایی که قاعده‌های تقدم گفته می‌شوند، انجام می‌گیرند. این قاعده‌ها در زیر آورده شده‌اند:

قاعده‌ی ۱. اول عبارت‌های درون دو کمان انجام می‌شوند،

قاعده‌ی ۲. عمل‌گرها در گروه‌های با رتبه‌ی بالای زیر، اولویت بالاتری برای اجرا شدن دارند.

گروه I **، +ی پیشوند، -ی پیشوند، (NOT)، ><، <>

گروه II *، /

گروه III +ی پسوند، -ی پسوند

گروه IV ||

گروه V <، =>، =، =<، >

گروه VI & (AND)

گروه VIII | (OR)

قاعده‌ی ۳. عمل‌گرها با اولویت یکسان (در یک گروه) از سمت چپ به راست عبارت محاسبه می‌شوند، به‌جز عمل‌گرهـای گـروه I که از سمت راست به چپ انجام می‌شوند.

در قاعده‌های بالا به نمادهای & و | عمل‌گرهای منطقی می‌گویند. عمل‌گر || مقدارهای متغیرهایی را که در طـرف چـپ و راسـت آن قرار می‌گیرند، در کنار هم قرار می‌دهد. عمل‌گرهای =< و => به ترتیب نمایانگر بزرگ‌تر یا برابر و کوچک‌تر یا برابر می‌باشند.

۱-۵-۱-۳ 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 نشانگر درست 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 SAS باشد.

مثال ۱-٤

```
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 ۱-۵

```
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 ۱-٦ 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;
```

در عبارت‌های بالا کلمه‌ی poverty یک برچسب است. در صورتی‌که عبارت 0.0<= exp <=0.8 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 نیز می‌توان استفاده کرد. مثال زیر نحوه‌ی استفاده از DO-END را نشان می‌دهد.

مثال ۱-۷

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

مثال ۱-۸ 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;

DATA group3;
   SET group2;
   IF scoregroup='F' THEN PUT id='NOT PASSED';
RUN;
```

```
PROC PRINT;
RUN;
```

در مثال بالا نماد "@@" به کاربر اجازه می‌دهد کـه در هـر خـط داده‌ای مقـدارهای بـیش از یـک مشـاهده وارد شـود. در غیـر این‌صورت در هر سطر داده‌ای باید یک مشاهده وارد شود. عبارت SET از عبارت‌هایی است که در گام DATA به کار می‌رود و بـه SAS می‌گوید که داده‌های مجموعه‌ی داده‌هایی که نامش بعد از SET می‌آید را در مجموعه‌ی داده‌هایی با نامی که بعد از عبـارت DATA می‌آید با اعمال عبارت‌های شرطی که به دنبال عبارت SET می‌آیند، بریزد. بـرای مثـال مجموعـه‌ی داده‌هـای group3 شامل مشاهده‌های مجموعه‌ی داده‌های group2 است که در مقابل نمره‌هـای F آن‌هاعبـارت "NOT PASSED" نوشـته شـده است.

خروجی مثال ۱–۸ در شکل‌های ۱–۳ و ۱–۴ نمایش داده شده است. در این مثال سه گام DATAی مختلف دیده می‌شود کـه بـه ترتیب مجموعه‌ی داده‌های موقت SAS به نام‌های group1، group2، و group3 را ایجاد می‌کنند.

در گام DATAی اول مجموعه‌ی داده‌های SAS به نام group1 با ۱۰ مشاهده و دو متغیر ساخته می‌شود. در گام DATAی دوم مجموعه‌ی داده‌هـای SAS بـا نـام group2 بـا اسـتفاده از مشـاهده‌هـا در group1 بـا یـک متغیـر نویسـه‌ای اضـافه بـه نـام scoregroup که مقدار آن برای هر مشاهده با توجه به مقدار score تعیین می‌شود، ایجاد مـی‌شـود. در گـام DATAی سـوم، مجموعه‌ی داده‌های SAS به نام group3 با یک اطلاع اضافی "NOT PASSED" برای idهایی که نمره‌ی F گرفته‌اند سـاخته می‌شود (شکل ۱–۴).

Obs	id	score	scoregroup
1	1	18	A
2	2	14	B
3	3	09	F
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

شکل ۱–۳

شکل ۱–۴

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 ...
```

۱-۵-۱-٤ محاسبه‌های مکرر

انجام محاسبه‌های تکراری علاوه بر استفاده از عبارت DO-END می‌تواند با استفاده از عبارت‌های DO نیز 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 ۱-۹ 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 ۱-۹ 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 شاخص i 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 خطاهای 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 شروع 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;
```

خطاهای داده‌ای

```
;
RUN;
```

در این مثال ارائه‌ی day مرتبط با متغیرهای d1 تا d7 و ارائه‌ی 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.

مثال ۱-۱۱

```
DATA IDO;
   DO i=1 to 4;
      DO j=2,4,6,8;
         k=(i-1)*5+j;
         OUTPUT;
      END;
   END;
```

```
RUN;

PROC PRINT DATA=IDO;
   TITLE 'Iterative DO Loop in a Single DATA Step';
RUN;
```

این مثال نمایشی از برنامه‌ای شامل یک حلقه‌ی DOی آشیانه‌ای ارائـه مـی‌دهـد. در برنامـه‌ی بـالا متغیرجدیـد k بـا اسـتفاده از عبارت‌های DO و متغیرهای شمارشی i و j ایجاد می‌شود. این فن برای ساختن سطح‌های عاملی در ترکیب با عامل‌ها یا تقاطع‌ها در آزمایش‌های عاملی به کار می‌رود. عبارت OUTPUT داخل حلقه‌ی DO موجب نوشـته شـدن متغیـر جدیـد k در مجموعـه‌ی داده‌های SASی که خواهد شد، می‌شود. اجرای برنامه‌ی SAS بالا ۱۶ مشاهده تولید می‌کند. خروجی برنامه به صورت زیر است.

Obs	i	j	k
1	1	2	2
2	1	4	4
3	1	6	6
4	1	8	8
5	2	2	7
6	2	4	9
7	2	6	11
8	2	8	13
9	3	2	12
10	3	4	14
11	3	6	16
12	3	8	18
13	4	2	17
14	4	4	19
15	4	6	21
16	4	8	23

شکل ۱–۵ برونداد مثال ۱–۱۱

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 کـه در DO UNTIL، شرط در انتهای حلقه بررسی می‌شود. به عبارت دیگر، حلقه‌ی DO WHILE ممکن است هیچ‌گـاه اجـرا نشـود، 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’
CATX(‘separator’, arg1, arg2, …)	دو یا چند رشته‌ی نویسه‌ای را، پس از حذف فاصله‌های خالی ابتدا و انتهای آن‌ها و قرار دان جداگر بین آن‌ها، به هم می‌چسباند.	a = ‘ cat ‘; b = ‘dog’; c = CATX(‘ ’, a, b);	c = ‘cat dog’
COMPRESS(arg, ‘chars’)	فاصله‌ی خالی یا نویسه‌های مشخص‌شده را از یک رشته‌ی نویسه‌ای حذف می‌کند.	a = ‘cat & dog’; b = compress(a); c = compress(a, ‘&’);	b = ‘cat&dog’ c = ‘cat dog)’

نتیجه	مثال	شرح	تابع
b = 8	a = ‘ My cat ’; b = LENGTH(a);	طـول یـک رشـته‌ی نویسـه‌ای 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 می‌کنـد. طـول to و from باید یکسان باشد.	TRANSLATE(arg, to1, from1, …)
b = ‘Ms. Smith’	a = ‘Mrs. Smith’; b = TRANWRD(a, ‘Mrs.’, ‘Ms.’);	replacement را جـــــــایگزین target موجود در arg می‌کند.	TRANWRD(arg, target, replacement)
b = ‘MY CAT’	a = ‘My cat’; b = UPCASE(a);	تمام حروف لاتـین موجـود 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 بزرگ‌تـر یـا مساوی arg را برمی‌گرداند.	CEIL(arg)
a = 2; b = -3;	a = FLOOR(2.3); b = FLOOR(-2.3);	بزرگ‌ترین عدد صحیح کوچک‌تـر یـا مساوی arg را برمی‌گرداند.	FLOOR(arg)
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, …)

نتیجه	مثال	شرح	تابع
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:);
```

۱-۵-۳ 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.

۱-۵-٤ 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.

۱-۵-٤-۱ 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.

۱-۵-٤-۲ 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;
```

۱-۵-۵ 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

۱-۵-٦ 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 15 107 14 105 11 103 17
RUN;
DATA science;
   INPUT id score;
   CARDS;
103 15 107 17 109 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;
```

۱-۵-۷ 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;
```

۱-٦ استفاده از نویسه‌های ویژه در عبارت INPUT

در این بخش نشانگرهای کنترلی ستونی @ و +، مشخص‌گرهای سطر نگهدار @ و @@ و نشانگر کنترل سطری #n بحث می‌شوند.

۱-٦-۱ نشانگرهای کنترلی

هنگامی‌که عبارت INPUT شروع به خواندن مقدارهای داده‌ای برای هر متغیر می‌کند از نشانگری برای چک کردن موقعیت داده‌ها استفاده می‌کند. در شروع اجرای عبارت INPUT موقعیت نشانگر در ابتدای داده‌های ورودی است، سپس این نشانگر با اینفورمت‌های متوالی و یا نشانگر کنترلی در عبارت INPUT حرکت می‌کند. برای مثال، عبارت INPUT زیر را در نظر بگیرید:

```
INPUT id 4. @10 ostan $3. inc 5.2 +1 exp 5.2 @28 month 2.;
```

این عبارت INPUT برای خواندن سطرهای داده‌ای به صورت زیر نوشته شده است:

```
0001XXXXXTEH46550X74050XXXXX02
```

در ابتدا نشانگر به موقعیت ستون ۱ رفته و متغیر id را با اینفورمت .4 می‌خواند و بعد به موقعیت ستون ۵ می‌رود. نشانگر کنترلی @10 سبب می‌شود که نشانگر به ستون ۱۰ رفته و مقدار متغیر نویسه‌ای ostan را با اینفورمت .$3 با مقدار TEH بخواند، سپس نشانگر به ستون ۱۳ می‌رود و مقدار متغیر عددی inc با مقدار 46550 را با اینفورمت 5.2 یعنی ۴۶۵/۵۰ می‌خواند. نشانگر کنترلی +۱ نشانگر را یک ستون به جلو یعنی به ستون ۱۹ می‌برد. پنج ستون با اینفورمت 5.2 برای مقدار جاری متغیر exp خوانده می‌شود. نشانگر کنترلی @28، نشانگر را به ستون ۲۸ برده و مقدار عددی month را با اینفورمت .2 می‌خواند. توجه داشته باشید که به جای نشانگر کنترلی @28 می‌توانستیم از +5 نیز استفاده کنیم. وقتی نشانگر به ستون ۳۱ می‌رود، SAS تشخیص می‌دهد که نشانگر به انتهای عبارت INPUT رسیده است، لذا خط داده‌ای بعدی با اینفورمت‌های عبارت INPUT خوانده می‌شود تا نشانگر به نقطه واوک که انتهای خط داده‌ای است برسد. در اینجا مجموعه‌ی داده‌های SAS کامل شده و آماده‌ی تحلیل با شیوه‌های SAS است.

۱-٦-۲ مشخص‌گر سطرنگهدار @

گاهی لازم می‌شود که بعد از خوانده شدن برخی از مقدارهای داده‌ای، عبارت‌های برنامه‌ای اجرا شوند. یکی از این حالت‌ها خواندن بقیه‌ی مقدارهای داده‌ای بسته به مقدار متغیری است که خوانده شده است. در این حالت لازم است که عبارت INPUT دومی برای خواندن بقیه‌ی مقدارهای داده‌ای وجود داشته باشد. این فن توسط مشخص‌گر سطر نگهدار @ انجام می‌شود.

نماد @ در انتهای عبارت INPUT درست قبل از نقطه‌واوک (;) ظاهر می‌شود. نماد @ سبب می‌شود که SAS نشانگر را در موقعیت‌های جاری نگهداشته و به SAS اجازه می‌دهد که عبارت INPUT دیگری اجرا شود.

مثال ۱-۱۲

```
DATA popu;
```

```
   INPUT ostan: $15. n @;
   DO i=1 to n;
      INPUT shahr:$10. Pop 4. @;
      OUTPUT;
   END;
   DROP i;
   LABEL pop='Population in 1000 Person';
   DATALINES;
TEHRAN 4 TEHRAN 7000 SHAHRERAY 500 SHEMIRAN 700 FIROOZKOH 800 YAZD 5
TAFT 200 MEHRIZ 150 ARDEKAN 250 MAYBOD 80
;
RUN;

PROC PRINT DATA=popu LABEL;
   TITLE 'Population Of 3 OstanS in 1385 Population Census';
RUN;
```

در این برنامه اینفورمت .$15 به SAS می‌گوید که مقدار متغیر ostan، ۱۵ ستون را به خود اختصاص می‌دهد و نماد : قبل از آن سبب می‌شود که .$15 وقتی به یک فاصله‌ی خالی می‌رسد آن را به عنوان جدا کننده‌ی مقدار داده‌ای بگیرد. بنا بر این با استفاده از نماد : دیگر لازم نیست که ۱۵ ستون در هر سطر داده‌ای برای متغیر ostan نگه داشته شود. نماد @ در انتهای عبارت INPUT اول سبب توقف نشانگر در موقعیت فعلی شده و به SAS اجازه می‌دهد که عبارت INPUT بعدی را اجرا کند.

عبارت INPUT زیر:

```
INPUT ostan: $15. n @;
```

سبب نگهداشتن خط داده‌ای بعد از خواندن مقدارهای متغیرهای ostan و n می‌شود. متغیر n شامل تعداد زوج مقدارهای متغیرهای shahr و pop که در این خط داده‌ای هستند، می‌باشد. این زوج مقدارها با عبارت .INPUT shahr:$10. Pop 4 @; خوانده می‌شوند. این عبارت INPUT در حلقه‌ی DOیی که به تعداد مقدار متغیر n اجرا می‌شود، حضور دارد.

نماد @ در عبارت INPUT دوم سبب نگهداشتن خط داده‌ای پس از خواندن زوج مقدار مربوط به متغیرهای shahr و pop شده و نشانگر را در وضعیت درست برای اجرای بعدی این عبارت INPUT قرار می‌دهد. پس از اینکه زوج مقدارهای متغیرهای shahr و pop خوانده شد، عبارت OUTPUT سبب نوشته شدن مقدارهای چهار متغیر ostan، n، shahr و pop در مجموعه‌ی داده‌های SAS به نام popu می‌شود. خروجی این برنامه در شکل ۱-۶ نشان داده شده است.

Population Of 3 OstanS in 1385 Population Census

Obs	ostan	n	shahr	Population in 1000 Person

1	TEHRAN	4	TEHRAN	7000
2	TEHRAN	4	SHAHRERAY	500
3	TEHRAN	4	SHEMIRAN	700
4	TEHRAN	4	FIROOZKOH	800

شکل ۱-۶ خروجی مثال ۱-۱۲

۱-٦-۳ مشخص‌گر خطنگهدار @@

وقتی داده‌های خام در گام DATA و در عبارت DATALINES آورده می‌شوند، SAS انتظار دارد که هر خط داده‌ای، شامل مقدارهای متغیرهایی باشد که در عبارت INPUT معرفی شده‌اند. بنا بر این پیش فرض SAS خواندن مقدارهای متغیرهایی یک مشاهده در هر سطر داده‌ای است. نماد @@ این امکان را به کاربر می‌دهد تا برای استفاده‌ی بیشتر از صفحه‌ی نمایش و نیز کنترل بیشتر بر برنامه، بیش از یک مشاهده در هر سطر وارد کند. مثال زیر نحوه‌ی استفاده از این نماد را نشان می‌دهد.

مثال ۱-۱۳

```
DATA score;
   INPUT name$ exam1 exam2 @@;
   Final=(exam1+exam2)/2;
   DATALINES;
AKBARI 15 17 AHSAN 17 14 ARBABZADEH 16 17 HASANI 18 16 HAGHIGHI 17 15
HUSEINI 18 19 JABARI 12 13 KARAMI 16 16 SAADAT 19 18
;
RUN;

PROC PRINT DATA=score;
RUN;
```

در مثال ۱-۱۳ در هر خط داده‌ای چند مشاهده شامل مقدار متغیرهای name، exam1، exam2 وجود دارد. عبارت final=(exam1+exam2)/2; میانگین نمره‌ها را به عنوان نمره‌ی نهایی محاسبه کرده و در متغیری به نام final در مجموعه‌ی داده‌های SAS که به نام score وارد می‌کند.

Obs	name	exam1	exam2	Final
1	AKBARI	15	17	16.0
2	AHSAN	17	14	15.5
3	ARBABZAD	16	17	16.5
4	HASANI	18	16	17.0
5	HAGHIGHI	17	15	16.0
6	HUSEINI	18	19	18.5
7	JABARI	12	13	12.5
8	KARAMI	16	16	16.
9	SAADAT	19	18	18.5

شکل ۱-۷ خروجی مثال ۱-۱۳

۱-٦-٤ نشانگر کنترل کننده‌ی سطر

گاهی وقت‌ها ممکن است هر مشاهده در مجموعه‌ی داده‌های خام به دلیل تعداد زیاد متغیرها در چند خط نوشته شود. در این حالت برای خواندن متغیرها باید شماره‌ی خط مربوط نیز به SAS داده شود. استفاده از گزینه n#، که n یک عدد صحیح و مثبت است، در عبارت INPUT نشانگر را به خط داده‌ای مناسب برای خواندن مقدار متغیرهای موردنظر حرکت می‌دهد.

مثال ۱-۱۴

```
DATA survey;
   INPUT #1 ID 1-4 gender $1. age 2. (marital educ inc)(1.)
         #2 @5 (q1-q10)(1.);
   DATALINES;
0001F22123
    1341034312
0002M30212
    2410332243
0003F48234
    3410231104
0004M45342
    1034113201
0005M50244
    3210342131
;
RUN;

PROC PRINT;
RUN;
```

در این مثال هر مشاهده در ماتریس داده‌ها در دو خط داده‌ای آورده شده است. بنا بر این عبارت INPUT باید قادر باشد تا متغیرها را در این دو سطر بخواند. نماد n# در عبارت INPUT، نشانگر را به خط داده‌ای متغیر موردنظر رهنمون می‌کند. در این مثال نماد ۱# در عبارت INPUT به SAS می‌گوید که متغیرهایی که بعد از ۱# آورده می‌شوند را در خط داده‌ای ۱ از مشاهده‌ی جاری بخواند. برای مثال متغیرهای id، gender، age، marital، edu، و inc با اینفورمت‌های مشخص‌شده در سطر ۱ از خط داده‌ای مشاهده‌ی جاری خوانده می‌شود و متغیرهای q1 تا q10 (پاسخ سوال‌های ۱ تا ۱۰ پرسشنامه‌ی آمارگیری) در سطر دوم خط داده‌ای مشاهده‌ی جاری و از ستون پنجم به بعد خوانده می‌شوند. شکل ۱-۸ خروجی مثال ۱-۱۴ را نمایش می‌دهد.

Obs	ID	gender	age	married	educ	inc	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10
1	1	F	22	1	2	3	1	3	4	1	0	3	4	3	1	2
2	2	M	30	2	1	2	2	4	1	0	3	3	2	2	4	3
3	3	F	48	2	3	4	3	4	1	0	2	3	1	1	0	4
4	4	M	45	3	4	2	1	0	3	4	1	1	3	2	0	1
5	5	M	50	2	4	4	3	2	1	0	3	4	2	1	3	1

شکل ۱-۸ خروجی برنامه‌ی مثال ۱-۱۴

در شکل ۱-۸ ستون obs نمایانگر شماره‌ی مشاهده‌ها است که به صورت پیش‌فرض در خروجی آورده شده است. اگر کاربر مایل به چاپ این ستون نباشد می‌تواند گزینه‌ی NOOBS را در ;PROC PRINT بنویسد.

نکته‌ی ۱-۲ در مثال ۱-۱۴ و در عبارت PROC PRINT نام مجموعه‌ی داده‌ها نوشته نشده است. پیش‌فرض در این حالت استفاده از آخرین مجموعه‌ی داده‌های SASی ایجاد شده است. اگر برنامه‌ای چند مجموعه‌ی داده‌های SAS تولید کرده باشد، برای استفاده از یک مجموعه‌ی داده‌های مشخص، کاربر باید نام مجموعه‌ی داده‌ها را در گزینه‌ی DATA= در عبارت PROC PRINT بیاورد.

۱-۷ مجموعه‌ی داده‌های دائمی SAS

پیش از این اشاره شد که مجموعه‌ی داده‌های SAS می‌توانند به صورت موقت و دائمی ایجاد شوند. هر مجموعه‌ی داده‌های SAS پس از ایجاد شدن باید در یک کتابخانه ذخیره شود. سامانه‌ی SAS دارای کتابخانه‌ای به نام Work است که تمام مجموعه‌ی داده‌هایی که در گام DATA تولید می‌شوند به طور خودکار در این کتابخانه به صورت موقت ذخیره می‌شوند، مگر اینکه کتابخانه‌ی دیگری برای ذخیره‌سازی در نظر گرفته شود. در این صورت مجموعه‌ی داده‌ها به صورت دائمی در این کتابخانه ذخیره می‌شود. دائمی بودن مجموعه‌ی داده‌ها به این معنا است که کاربر هر زمان که مایل باشد می‌تواند از آن استفاده کند. مجموعه‌ی داده‌هایی که در کتابخانه‌ی Work ذخیره می‌شوند، با خروج کاربر از محیط SAS از بین می‌روند و این معنای موقتی بودن مجموعه‌ی داده‌ها است.

۱-۷-۱ ساختن کتابخانه

پس از فعال کردن سامانه‌ی SAS، در پنجره‌ی Explorer، نماد Libraries را کلیک کنید. پس از این عمل در سمت چپ نمایشگر صفحه‌ای باز خواهد شد که نام کتابخانه‌های ایجاد شده را نشان می‌دهد. در بار اول که این عمل انجام می‌شود، در این پنجره تنها کتابخانه‌های Work، Sashelp، و Sasuser وجود دارند. هنگامی که در محیط سامانه‌ی SAS، پنجره‌ای فعال می‌شود، نشانه‌های نوار ابزار (این نشانه‌ها در نوار بالایی صفحه‌ی نمایش وجود دارند) تغییر می‌کنند. در نوار ابزار، نشانه‌ی کتابخانه‌ی جدید ( ) را کلیک کنید، صفحه‌ی جدیدی با عنوان NewLibrary باز می‌شود. در این قسمت در جعبه‌ی Name، نام کتابخانه را وارد کنید، در جعبه‌ی Engine کلمه‌ی Default نوشته شده است، تغییری در این جعبه ندهید. جعبه‌ای با عنوان Enable at startup را کلیک کنید تا نماد √ در آن ظاهر شود. در جعبه‌ی Path آدرسی که باید کتابخانه در آن ساخته شود را وارد یا از Browse استفاده کنید. سپس OK را کلیک کنید. در این زمان کتابخانه‌ی مورد نظر ساخته شده و نام آن در صفحه‌ی کتابخانه‌ها نوشته می‌شود. در اینجا شما قادر خواهید بود مجموعه‌ی داده‌های SAS دائمی در این کتابخانه ذخیره کنید.

علاوه بر این، با استفاده از عبارت LIBNAME نیز می‌توانید یک کتابخانه‌ی SAS ایجاد کنید. شکل عمومی تعریف کتابخانه‌ی SAS به‌صورت زیر است:

```
LIBNAME نام کتابخانه 'مسیر';
```

مثال: دستور زیر کتابخانه‌ای به نام MyLib برای دسترسی به مجموعه‌های داده‌های موجود در مسیر C:\My Files می‌سازد:

```
LIBNAME MyLib 'C:\My Files';
```

شما می‌توانید کتابخانه‌ای تعریف کنید که به بیش از یک مسیر دسترسی داشته باشد، حتی ترکیبی از چند کتابخانه‌ی موجود باشد:

```
LIBNAME NewLib (Mylib 'D:\Data' 'E:\Data');
```

با توجه به دو دستور بالا، کتابخانه‌ی NewLib به داده‌های موجود در کتابخانه‌ی MyLib (مسیر C:\My Files) و مسیرهای D:\Data و E:\Data دسترسی خواهد داشت. توجه داشته باشید در صورتی که بخواهید روی این کتابخانه، مجموعه‌ی داده‌هایی ایجاد کنید، این مجموعه‌ی داده‌ها روی اولین بخش مشخص‌شده یعنی مسیر C:\My Files ایجاد خواهد شد. اگر مجموعه‌ی داده‌هایی با یک نام مشترک روی دو یا چند مسیر مشخص‌شده موجود باشد، هنگام خواندن داده‌ها، ملاک اولین مسیر خواهد بود.

۱-۷-۲ ایجاد مجموعه‌ی داده‌های دائمی SAS

در گام DATA نام مجموعه‌ی داده‌ها مشخص می‌شود. این نام‌ها شامل دو بخش هستند. این دو بخش با یک نقطه از هم جدا می‌شوند. بخش اول شامل نام کتابخانه است و بخش دوم شامل نام مجموعه‌ی داده‌ها است. برای مثال عبارت DATAی زیر، مجموعه‌ی داده‌هایی به نام score در کتابخانه‌ی mydat ایجاد می‌کند.

```
DATA  mydat.score;
```

اگر در گام DATA نام بخش اول، توسط برنامه‌نویس مشخص نشود، SAS به طور خودکار نام Work را به بخش اول اختصاص می‌دهد. به عبارت دیگر در این حالت، مجموعه‌ی داده‌هایی که ایجاد می‌شود در کتابخانه‌ی Work به صورت موقت ذخیره می‌شود.

اگر برنامه‌نویس نامی غیر از Work به بخش اول اختصاص دهد، مجموعه‌ی داده‌های SAS با نام score به صورت دائمی در این کتابخانه ذخیره می‌شود. فراخوانی مجموعه‌ی داده‌های دائمی SAS با مشخص کردن نام کتابخانه و نام مجموعه‌ی داده‌ها صورت می‌گیرد. برای مثال عبارت

```
PROC PRINT DATA=mydat.score;
```

مجموعه‌ی داده‌های دائمی SAS به نام score که در کتابخانه‌ی mydat ذخیره شده است را برای چاپ کردن فراخوانی می‌کند.

۱-۷-۳ عبارت INFILE

در مثال‌هایی که تا به حال دیدیم، مجموعه‌ی داده‌های خام در برنامه و در گام DATA وارد می‌شد. اگر مجموعه‌ی داده‌های خام در یک فایل خارجی، برای مثال در یک دیسک سخت، لوح فشرده و یا ... قبلا ذخیره شده باشد، فراخوانی آن در SAS با عبارت

INFILE صورت می‌گیرد. در مثال زیر مجموعه‌ی داده‌های خام (با فورمت متنی) به نـام score و بـا پسـوند dat کـه در آدرس C:\survey\data قرار دارد با عبارت INFILE فراخوانی شده و پـس از تبـدیل بـه مجموعـه‌ی داده‌هـای SAS در کتابخانـه‌ی mydat با نام score به صورت دائمی ذخیره می‌شود.

مثال ۱-۱۵

```
DATA mydat.score;
   INFILE 'C:\survey\data\score.dat';
   INPUT name$ exam1 exam2 hw;
   final = (exam1 + exam2) / 2 + hw;
RUN;
```

۱-۸ گام PROC

مجموعه‌ی داده‌های SAS، می‌توانند با استفاده از روش‌های آماری تحلیل شوند. این روش‌ها برنامه‌های پیش‌نوشته‌ای هستند کـه در گام PROC فراخوانی می‌شوند. هر گام PROC شامل عبارت PROC و عبارت‌هایی است که در گام PROC به کار مـی‌رونـد. این عبارت‌ها صورت دستوری شیوه‌ی مورد نظر را تشکیل می‌دهند. برخی از عبارت‌ها عمومی هستند، به این معنا که در هـر گـام PROC به کار می‌روند و برخی اختصاصی. صورت عمومی یک گام PROC به شکل زیر است:

```
PROC    نام شیوه  <فهرست گزینه‌ها> ;
< عبارت‌هایی که اطلاعات به شیوه می‌دهند ;>
< عبارت‌هایی که درباره‌ی متغیرها اطلاع می‌دهند ;>
RUN;
```

بیشتر شیوه‌های SAS را می‌توان با عبارت PROC ساده فعال کرد. مثال زیر ساده‌ترین شکل استفاده از شیوه‌ی PRINT را نشـان می‌دهد.

مثال ۱-۱۶

```
PROC PRINT DATA=mydat.score;
RUN;
```

این برنامه از شیوه‌ی PRINT برای چاپ مجموعه‌ی داده‌های score استفاده می‌کند. مجموعه‌ی داده‌های score در کتابخانه‌ی mydat به صورت دائمی ذخیره شده است. اگر کاربر مایل باشد که تنها فهرستی از نام‌ها و نمره‌ی نهایی داشـته باشـد مـی‌توانـد برنامه‌ی زیر را اجرا کند.

```
PROC PRINT DATA=mydat.score;
   VAR name final;
```

```
RUN;
```

عبارت VAR که کوته‌نوشته‌ی VARIABLE است، متغیرهایی را که باید در تحلیل شرکت داشته باشند را مشخص می‌کند.

نکته‌ی ۱-۳ برنامه‌ها در پنجره‌ی Editor نوشته می‌شود و با کلیک کردن روی نشانه‌ی  در نوار ابزار، اجرا می‌شود.

نکته‌ی ۱-٤ بعد از اجرای هر برنامه و قبل از مرور خروجی در پنجره‌ی Output بهتر است پیغام‌های پنجره‌ی Log برای اطلاع از درستی برنامه بررسی شوند.

نکته‌ی ۱-۵ در صورتی‌که عبارت VAR به کار نرود، پیش فرض SAS استفاده از تمام متغیرهای عددی در مجموعه‌ی داده‌ها است. در شیوه‌ی PRINT اگر VAR مشخص نشود، متغیرهای عددی و نویسه‌ای برای چاپ استفاده می‌شوند.

مثال ۱-۱۷ مجموعه‌ی داده ی SAS ایجاد شده در مثال ۱-۱ به نام exam که در کتابخانه‌ی work ذخیره شده است را در نظر بگیرید. می‌خواهیم تعداد مشاهده‌ها، میانگین و خطای استاندارد متغیرهای midterm و final را با یک رقم اعشار برای این مجموعه‌ی داده‌ها محاسبه کنیم. برنامه‌ی زیر این تحلیل را انجام می‌دهد.

```
PROC MEANS DATA=exam N MEAN STD MAXDEC=1;
   VAR midterm final
Run;
```

کلیدواژه‌ی MEANS در عبارت PROC شیوه‌ی آماری مناسب تحلیل داده‌ها را فراخوانی می‌کند. سایر گزینه‌ها در عبارت PROC اختیاری هستند. در گزینه‌ی =DATA نام مجموعه‌ی داده‌هایی که باید تحلیل شوند آورده می‌شود (exam). اگر این گزینه حذف شود، SAS به طور پیش‌فرض آخرین مجموعه‌ی داده‌های تولیدشده را به کار می‌گیرد. گزینه‌های N، MEANS، و STD به ترتیب از SAS می‌خواهند تا تعداد مشاهده‌ها (هشت مشاهده)، میانگین و خطای استاندارد متغیرهای معرفی‌شده در عبارت VAR را محاسبه کرده و چاپ کند. گزینه‌ی MAXDEC=1 از SAS می‌خواهد که آماره‌های خروجی را با یک رقم اعشار چاپ کند. عبارت VAR که کوته‌نوشته‌ی VARIABLE است به SAS می‌گوید که کدام متغیرها را در تحلیل وارد کند. اگر این گزینه حذف شود، SAS به‌طور پیش‌فرض تمام متغیرهای عددی موجود در مجموعه‌ی داده‌ها را به کار می‌گیرد.

۱-۸-۱ مشخص‌سازی گزینه‌ها در عبارت PROC

برخی گزینه‌ها در عبارت PROC عمومی هستند، یعنی می‌توانند در هر عبارت PROC به کار روند. برای مثال اگر کاربر مایل نباشد که آخرین مجموعه‌ی داده‌های SAS ایجاد شده را تحلیل کند، می‌تواند نام مجموعه‌ی داده‌های SAS مورد نظر را در گزینه‌ی =DATA مشخص کند. برای مثال عبارت PROC PRINT DATA=score مجموعه‌ی داده‌های SAS به نام score را چاپ می‌کند.

به عنوان مثالی دیگر، عبارت PROC MEANS mean std; شیوه‌ی MEANS را فعال می‌کند. گزینه‌های mean، و std از این شیوه می‌خواهند که برای تمام متغیرهای عددی در آخرین مجموعه‌ی داده‌های SAS ایجاد شده، آماره‌های میانگین و انحراف استاندارد را چاپ کند. پیش‌فرض SAS چاپ آماره‌های خلاصه‌ی: تعداد مشاهده‌ها، میانگین، انحراف استاندارد، مینیمم و ماکسیمم داده‌ها است.

در مثال اخیر گزینه‌های mean، و std گزینه‌های اختصاصی برای شیوه‌ی MEANS هستند.

۱-۸-۲ عبارت‌هایی که اطلاعات به شیوه می‌دهند

برخی از گزینه‌های اختیاری که اطلاعاتی به شیوه‌ی موردنظر برای اجرا می‌دهند را می‌توان در گام PROC وارد کرد. در مثال قبل، عبارت PROC MEANS mean std; میانگین و انحراف استاندارد را برای تمام متغیرهای عددی در مجموعه‌ی داده‌های SAS ایجاد شده محاسبه می‌کند. اگر کاربر محاسبه‌ی این آماره‌ها را برای متغیرهای خاصی بخواهد می‌تواند از عبارت VAR استفاده کند. برای مثال

```
PROC MEANS std mean;
   VAR X Y;
RUN;
```

آماره‌های mean و std را تنها برای متغیرهای X و Y محاسبه و چاپ می‌کند.

یکی از عبارت‌های اطلاعاتی شیوه‌ای، عبارت BY است که به شیوه‌های SAS اجازه می‌دهد که بر پایه‌ی مقدارهای متغیر (یا متغیرها) که در عبارت BY فهرست شده‌اند، زیرمجموعه‌های مجموعه‌ی داده‌های مشخص شده را پردازش کند. به عبارت دیگر SAS شیوه‌ی موردنظر را به‌صورت پیاپی روی زیر مجموعه‌ی داده‌ها اجرا می‌کند. شکل عمومی عبارت BY به صورت زیر است:

BY فهرست متغیرها;

هنگامی‌که عبارت BY در گام PROC ظاهر می‌شود، شیوه‌ی موردنظر انتظار دارد که مجموعه‌ی داده‌ها بر حسب مقدارهای متغیر (یا متغیرهای) فهرست‌شده در عبارت BY مرتب شده باشد.

مثال ۱-۱۸

```
PROC MEANS DATA = mydat.score;
   BY gender inc;
   VAR midterm final;
RUN;
```

برنامه‌ی بالا شیوه‌ی MEANS را روی گروه‌های به وجود آمده از مرتب‌سازی مجموعه‌ی داده‌ها بر حسب مقدارهای متغیرهای gender و inc به صورت مجزا و تنها برای متغیر final و midterm اجرا می‌کند.

۱-۸-۳ عبارت‌های ویژگی متغیر

عبارت‌های ویژگی متغیر به کاربر SAS اجازه می‌دهد تا فورمت، اینفورمت، برچسب و طول متغیرها را در یک گام PROC مشخص کند. این مشخص‌سازی ویژگی‌های متغیرها به هنگام اجرای یک گام PROC در این گام انجام می‌شود. این عبارت‌ها نیز می‌تواند در گام DATA برای مشخص‌سازی ویژگی‌های متغیرها به کار روند، در این صورت این مشخصه‌ها به صورت دائمی با متغیرها در مجموعه‌ی داده‌های ایجاد شده در گام DATA مرتبط می‌شوند. در این حالت این ویژگی‌ها در شیوه‌های بعدی SAS درون یک گام PROC در دسترس خواهند بود. عبارت‌های FORMAT و LABEL عبارت‌های ویژگی متغیرها هستند که در گام PROC به کار می‌روند.

عبارت FORMAT را می‌توان هم در گام DATA و هم در گام PROC برای مشخص‌سازی فورمت‌ها به کار برد. فورمت‌های SAS برای تبدیل مقدارهای داده‌ای ذخیره شده در مجموعه‌ی داده‌های SAS به صورتی که موردنظر کاربر برای چاپ کردن است، به کار می‌روند. فورمت‌ها اطلاعاتی به SAS در خصوص تعداد نویسه‌های مقدار داده‌ای، صورت مقدار داده‌ای که باید در چاپ ظاهر شود، نمادهایی که باید در چاپ مقدار داده‌ای استفاده شوند مثل نقطه‌ی اعشاری، واوک، علامت دلار ($) می‌دهند.

مثال ۱-۱۹

```
DATA rev;
   INPUT region : $8. ostan $3. +1 month monyy5.
         sper revenue expenses;
   FORMAT month monyy5. Revenue dollar12.2;
   LABEL region='Sale Region'
   sper='Sale Personnel';
   DATALINES;
SOUTHERN YAZ JAN98 10 10000 8000
SOUTHERN YAZ FEB98 10 11000 8500
SOUTHERN YAZ MAR98  9 13000 9500
SOUTHERN KER JAN98  5  8000 2000
SOUTHERN KER FEB98  7  6000 2000
NORTHERN AGH MAR98  3  1200 1700
NORTHERN ASH FEB98  4  2000 4000
NORTHERN GIL MAR98  5  5000 7000
NORTHERN MAZ FEB98  7  4000 6000
CENTRAL  QUM JAN98 13  2100 4000
CENTRAL  QUM FEB98 14  2200 4500
CENTRAL  ESF JAN98 10 10000 8000
CENTRAL  ESF FEB98  9 11000 8200
CENTRAL  ESF MAR98 10 12000 8900
CENTRAL  MAR JAN98  4  6100 2000
CENTRAL  MAR FEB98  4  6050 2100
;
```

```
RUN;

PROC SORT DATA=rev;
   BY region ostan month;
RUN;

PROC PRINT;
RUN;

PROC PRINT DATA=rev LABEL;
   BY region ostan;
   FORMAT expenses dollar10.2;
   LABEL  ostan = Ostan month = Month
        Revenue = 'Ostan Revenue'
       Expenses = 'Overhead Expenses';
   ID region expenses;
   SUM revenue expenses;
   SUMBY region;
   TITLE 'Sales Reported by Ostan and Region';
RUN;
```

مثال بالا برنامه‌ی SASی است که چند ویژگی گام‌های DATA و PROC را نشان می‌دهد. در این برنامه دو گـام PRINT بـرای چاپ یک مجموعه‌ی داده‌های SAS وجود دارند. گام PROC اول ساده‌ترین شکل فعال کردن شیوه‌ی PRINT است، در صـورتی که گام PROC دوم شامل چند عبارت اطلاعاتی شیوه‌ای و ویژگی متغیری است.

Obs	region	ostan	month	sper	revenue	expenses
1	CENTRAL	ES	JAN09	8	$10.00	10000
2	CENTRAL	ES	FEB09	8	$9.00	11000
3	CENTRAL	ES	MAR09	8	$10.00	12000
4	CENTRAL	MA	JAN09	8	$4.00	6100
5	CENTRAL	MA	FEB09	8	$4.00	6050
6	CENTRAL	QU	JAN09	8	$13.00	2100
7	CENTRAL	QU	FEB09	8	$14.00	2200
8	NORTHERN	AGH	MAR98	3	$1,200.00	1700
9	NORTHERN	ASH	FEB98	4	$2,000.00	4000
10	NORTHERN	GIL	MAR98	5	$5,000.00	7000
11	NORTHERN	MAZ	FEB98	7	$4,000.00	6000
12	SOUTHERN	KER	JAN98	5	$8,000.00	2000
13	SOUTHERN	KER	FEB98	7	$6,000.00	2000
14	SOUTHERN	YAZ	JAN98	10	$10,000.00	8000
15	SOUTHERN	YAZ	FEB98	10	$11,000.00	8500
16	SOUTHERN	YAZ	MAR98	9	$13,000.00	9500

شکل ۱-۹ خروجی گام PRINT اول مثال ۱-۱۹

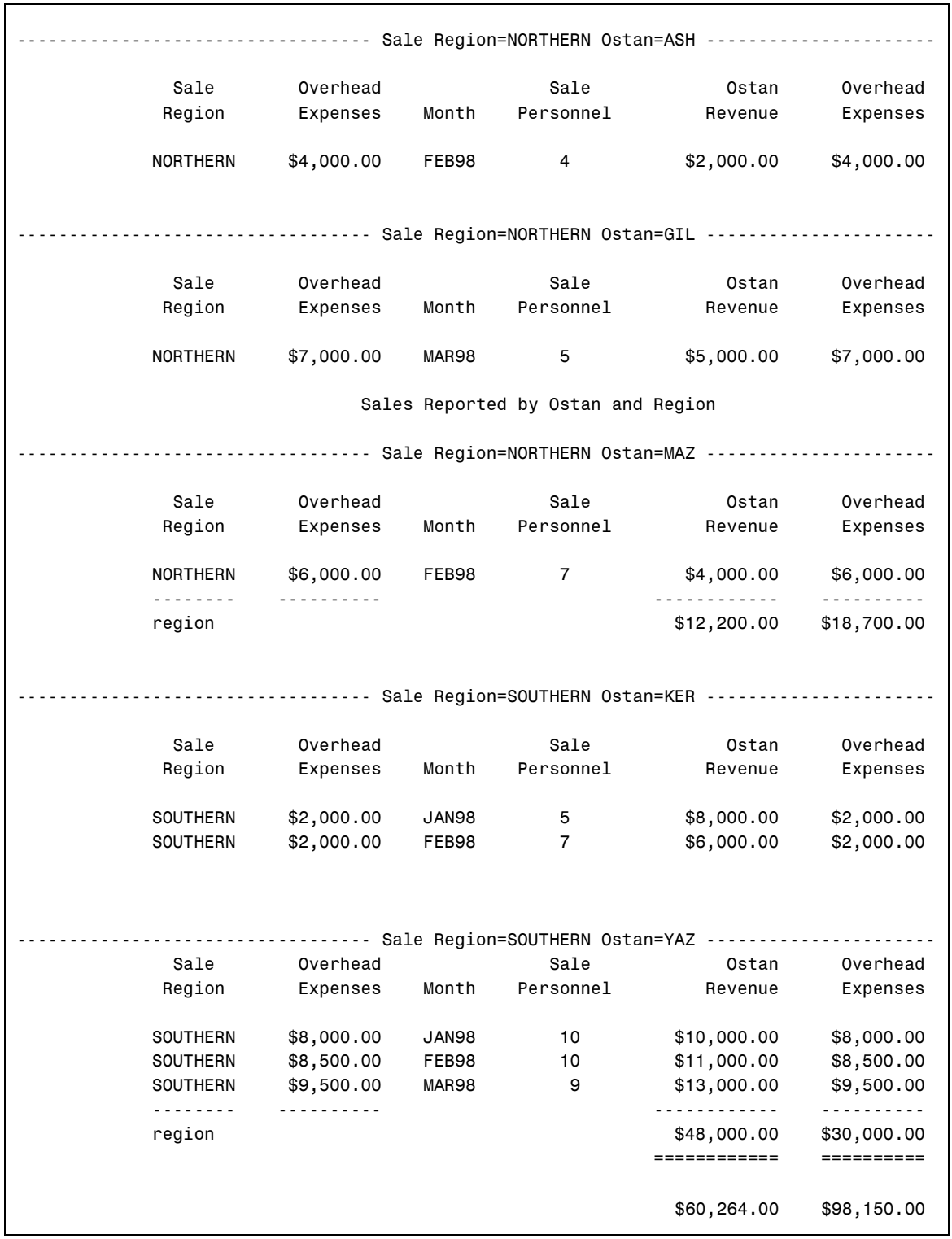
در عبارت INPUT اصلاح‌گر **":"** برای خواندن مقدار متغیر region و از اینفورمت تاریخ .monyy5 برای خواندن مقدار تاریخ استفاده شده است. در گام DATA فورمت‌هایی برای متغیرهای month و revenue مشخص شده‌اند. این فورمت‌ها در گام PRINT اول برای چاپ مقدارهای این متغیرها به کار می‌روند. شکل ۱-۹ خروجی گام PRINT اول را نشان می‌دهد.

عبارت `BY region ostan month;` در گام SORT سبب شده است تا مشاهده‌ها به ترتیب افزایشی بر حسب مقدارهای month در داخل گروه‌های ایجاد شده با مقدارهای یکسان region و ostan مرتب و در آخر مشاهده‌ها در مجموعه‌ی داده‌ها بر حسب مقدارهای region به صورت افزایشی مرتب شوند.

شیوه‌ی PRINT اول فهرستی از مشاهده‌ها که با مقدار متغیرهای month، ostan، و region مرتب شده‌اند، چاپ می‌کند.

در گام PRINT دوم عبارت `BY region ostan ;` سبب می‌شود که فهرستی از گروه‌هایی که با مقدارهای متغیر ostan داخل مقدارهای متغیر region تعیین می‌شوند، چاپ می‌شود. شکل ۱-۱۰ خروجی این گام را نشان می‌دهد.

```
                              Sales Reported by Ostan and Region

---------------------------------- Sale Region=CENTRAL Ostan=ES ----------------------

             Sale        Overhead                Sale             Ostan      Overhead
            Region       Expenses    Month    Personnel         Revenue      Expenses

            CENTRAL    $10,000.00    JAN09        8              $10.00    $10,000.00
            CENTRAL    $11,000.00    FEB09        8               $9.00    $11,000.00
            CENTRAL    $12,000.00    MAR09        8              $10.00    $12,000.00


---------------------------------- Sale Region=CENTRAL Ostan=MA ----------------------

             Sale        Overhead                Sale             Ostan      Overhead
            Region       Expenses    Month    Personnel         Revenue      Expenses

            CENTRAL     $6,100.00    JAN09        8               $4.00     $6,100.00
            CENTRAL     $6,050.00    FEB09        8               $4.00     $6,050.00



---------------------------------- Sale Region=CENTRAL Ostan=QU ----------------------

             Sale        Overhead                Sale             Ostan      Overhead
            Region       Expenses    Month    Personnel         Revenue      Expenses

            CENTRAL     $2,100.00    JAN09        8              $13.00     $2,100.00
            CENTRAL     $2,200.00    FEB09        8              $14.00     $2,200.00
            -------     ----------                              ------------    ----------
            region                                                $64.00    $49,450.00


---------------------------------- Sale Region=NORTHERN Ostan=AGH ----------------------

             Sale        Overhead                Sale             Ostan      Overhead
            Region       Expenses    Month    Personnel         Revenue      Expenses

            NORTHERN    $1,700.00    MAR98        3           $1,200.00     $1,700.00
```



```
--------------------------------- Sale Region=NORTHERN Ostan=ASH ----------------------

              Sale        Overhead                Sale             Ostan      Overhead
             Region       Expenses    Month    Personnel          Revenue      Expenses

           NORTHERN     $4,000.00    FEB98         4           $2,000.00     $4,000.00


--------------------------------- Sale Region=NORTHERN Ostan=GIL ----------------------

              Sale        Overhead                Sale             Ostan      Overhead
             Region       Expenses    Month    Personnel          Revenue      Expenses

           NORTHERN     $7,000.00    MAR98         5           $5,000.00     $7,000.00

                                Sales Reported by Ostan and Region

--------------------------------- Sale Region=NORTHERN Ostan=MAZ ----------------------

              Sale        Overhead                Sale             Ostan      Overhead
             Region       Expenses    Month    Personnel          Revenue      Expenses

           NORTHERN     $6,000.00    FEB98         7           $4,000.00     $6,000.00
           --------     ----------                            ------------   ----------
           region                                             $12,200.00    $18,700.00


--------------------------------- Sale Region=SOUTHERN Ostan=KER ----------------------

              Sale        Overhead                Sale             Ostan      Overhead
             Region       Expenses    Month    Personnel          Revenue      Expenses

           SOUTHERN     $2,000.00    JAN98         5           $8,000.00     $2,000.00
           SOUTHERN     $2,000.00    FEB98         7           $6,000.00     $2,000.00



--------------------------------- Sale Region=SOUTHERN Ostan=YAZ ----------------------
              Sale        Overhead                Sale             Ostan      Overhead
             Region       Expenses    Month    Personnel          Revenue      Expenses

           SOUTHERN     $8,000.00    JAN98        10          $10,000.00     $8,000.00
           SOUTHERN     $8,500.00    FEB98        10          $11,000.00     $8,500.00
           SOUTHERN     $9,500.00    MAR98         9          $13,000.00     $9,500.00
           --------     ----------                            ------------   ----------
           region                                             $48,000.00    $30,000.00
                                                              ============   ==========

                                                              $60,264.00    $98,150.00
```

شکل ۱-۱۰ خروجی گام PRINT دوم مثال ۱-۱۹

در شیوه‌ی PRINT دوم عبارت‌های `SUM revenue expenses;` و `SUM region;` نحوه‌ی محاسبه و چاپ مقدار کل را برای متغیرهای عددی revenue و expenses در مجموعه‌ی داده‌ها را مشخص می‌کند. خروجی این گام نشان

می‌دهد که برای هر گروهی که توسط مقدارهای مشابه متغیر region تعریف می‌شود، جمع برای مقدارهای متغیرهای revenue و expenses محاسبه و چاپ شده است.

برچسب‌هایی که برای متغیرهای region و sper در عبارت LABEL در گام DATA تعریف شده‌اند در خروجی اولین PROC PRINT ظاهر نشده‌اند. این امر به این دلیل است که ویژگی متغیری در مجموعه‌ی داده‌ها در برخی گام‌های PROC به کار گرفته نمی‌شوند مگر اینکه توسط کاربر و با گزینه‌ای درخواست شوند. گزینه‌ی LABEL در PROC PRINT دوم از SAS می‌خواهد که برچسب‌های تعریف‌شده برای متغیرهای region و sper را چاپ کند.

در گام PRINT دوم عبارت FORMAT expenses dollar10.2; سبب می‌شود که مقدار متغیر expenses با فورمت dollar10.2 در خروجی چاپ شود. این فورمت موجب نوشته شدن نماد $ قبل از مقدار متغیر و جدا شدن هر سه رقم از سمت راست با یک واوک می‌شود.

عبارت LABEL در این گام به متغیرهای ostan، month، revenue، و expenses برچسب‌های مشخص‌شده را آن‌چنان‌که در خروجی ۱-۱۰ نشان داده شده است، منتسب می‌کند.

عبارت ID region ostan; از چاپ شدن ستون obs که به صورت خودکار چاپ می‌شود جلوگیری کرده و مقدار متغیرهای region، و ostan را به عنوان مشخص‌ساز مشاهده‌ها به کار می‌برد.

۱-۸-۳ ایجاد فورمت‌های جدید با استفاده از شیوه‌ی FORMAT

برای ایجاد فورمت‌های مورد نظر خود می‌توانید از شیوه‌ی FORMAT استفاده کنید. شکل عمومی این شیوه به‌صورت زیر است:

```
PROC FORMAT;
   VALUE name range-1 = 'formatted-text-1'
              range-2 = 'formatted-text-2'
                      .
                      .
                      .
              range-n = 'formatted-text-2';
RUN;
```

اگر فورمت برای داده‌های نویسه‌ای به کار می‌رود باید نام آن با $ آغاز شود. هر range مقادیری از متغیر مورد نظر است که متن داخل گیومه به آن منتسب می‌شود. نمونه‌هایی از rangeهای معتبر را در زیر می‌بینید:

```
          'A' = 'Asia'
1, 3, 5, 7, 9 = 'Odd'
500000 - HIGH = 'Not Affordable'
     13 -< 20 = 'Teenager'
    0 <- HIGH = 'Positive Non Zero'
```

```
        OTHER = 'Bad Data'
```

مثال: دستورات زیر فورمت‌هایی برای متغیرهای مورد نظر ساخته و آن‌ها را مورد استفاده قرار می‌دهد:

```
DATA cars;
   INPUT Age Sex Income Color $;
   CARDS;
19 1 14000 Y
45 1 65000 G
72 2 35000 B
31 1 44000 Y
58 2 83000 W
RUN;
PROC FORMAT;
   VALUE gender 1 = 'Male'
                2 = 'Female';
   VALUE agegroup 13 -< 20 = 'Teen'
                  20 -< 65 = 'Adult'
                 65 - HIGH = 'Senior';
   VALUE $col 'W' = 'Moon White'
              'B' = 'Sky Blue'
              'Y' = 'Sunburst Yellow'
              'G' = 'Rain Cloud Gray';
RUN;
PROC PRINT DATA = cars;
   FORMAT Sex gender. Age agegroup. Color $col. Income DOLLAR8.;
RUN;
```

Obs	Age	Sex	Income	Color
1	Teen	Male	$14,000	Sunburst Yellow
2	Adult	Male	$65,000	Rain Cloud Gray
3	Senior	Female	$35,000	Sky Blue
4	Adult	Male	$44,000	Sunburst Yellow
5	Adult	Female	$83,000	Moon White

۱-۹ عبارت‌هایی که در هر جای برنامه می‌آیند

عبارت‌هایی که در برنامه‌ی SAS می‌آیند یا باید در گام DATA و یا در گام PROC قرار بگیرند. برخی از عبارت‌ها تنها در گام DATA معتبرند و برخی تنها در گام PROC.

اگر عبارت‌ها درخارج این گام‌ها قرار بگیرند، برای مثال بعد از عبارت ;RUN نوشته شوند، برنامه اجرا نمی‌شود و SAS در پنجره‌ی Log در این عبارت خطا اعلام کرده و پردازش را متوقف می‌کند. در SAS برخی عبارت‌ها سراسری هستند. این عبارت‌ها در هر قسمت از برنامه می‌توانند مورد استفاده قرار گیرند. مهم‌ترین این عبارت‌ها عبارت‌اند از: TITLE، FOOTNOTE، OPTIPONS، و GOPTIONS.

۱-۹-۱ عبارت TITLE

عبارت TITLE به برنامه‌نویس این امکان را می‌دهد که به‌تواند عنوان و یا عنوان‌های مورد نظر خود را در خروجی برنامه تعیین کند. برنامه‌نویس می‌تواند با عبارت ' 'TITLE*n* که n یک عدد از ۱ تا ۱۰ است حداکثر ۱۰ عنوان برای خروجی برنامه بنویسد. محتوای عنوان در ' ' قرار می‌گیرد. اگر *n* از عبارت TITLE*n* حذف شود، SAS آن را ۱ فرض می‌کند. اگر برنامه‌نویس از عبارت TITLEاستفاده کرده و عنوانی را برای خروجی به نویسد، این عنوان در تمام صفحه‌های خروجی آورده می‌شود. در صورتی‌که کاربر مایل باشد در قسمتی از برنامه عنوان دیگری نوشته شود باید از عبارت TITLE*n* جدیدی استفاده کند. در صورتی که کاربر به خواهد در قسمتی از خروجی برنامه عنوان حذف شود می‌تواند از عبارت ; ' 'TITLE*n* استفاده کند.

۱-۹-۲ عبارت FOOTNOTE

این عبارت امکان پانویسی را به کاربر SAS می‌دهد. شکل کلی این عبارت ;' 'FOOTNOTE*n* است. در این عبارت *n* تعداد پانویس‌ها است و می‌تواند مقدار ۱ تا ۱۰ بگیرد. محتوای پانویس در ' ' نوشته می‌شود.

۱-۹-۳ عبارت OPTIONS

سامانه‌ی SAS تنظیم‌هایی دارد که وقتی کاربر وارد محیط SAS می‌شود آن‌ها فعال می‌شوند. برای مثال محیط SAS این گونه تنظیم شده است که در خروجی برنامه‌ها تاریخ اجرای برنامه، شماره مسلسل صفحه، در هر صفحه ۶۰ سطر و در هر سطر ۸۰ ستون قرار داشته باشد. اگر برنامه‌نویس چند برنامه اجرا کند صفحه‌های خروجی به صورت پیاپی شماره داده می‌شود. به عبارت دیگر در هر اجرای برنامه، شماره‌ی صفحه خروجی از ۱ شروع نمی‌شود. کاربر با استفاده از عبارت OPTIONS می‌تواند به صورت موقت این تنظیم‌ها را تغییر دهد. موقتی بودن تنظیم‌ها به این معنا است که وقتی کاربر از محیط SAS خارج می‌شود، تنظیم‌ها به صورت اولیه باز می‌گردند. شکل کلی عبارت OPTIONS به صورت زیر است.

```
OPTIONS گزینه‌ها;
```

در عبارت بالا، گزینه‌ها محیط SAS را تنظیم می‌کنند.

مثال ۱-۲۰

```
OPTIONS NODATE PAGENO=1 PS=65 LS=85;
```

در عبارت بالا گزینه‌ی NODATE به SAS می‌گوید که لازم نیست تاریخ اجرای برنامه در خروجی چاپ شود. گزینه‌ی PAGENO=1 از SAS می‌خواهد که خروجی هر برنامه‌ای که اجرا می‌شود، از شماره‌ی ۱ شروع شود. گزینه‌های PS=65 و LS=85 که کوته‌نوشته‌ی PAGE SIZE= و LINE SIZE= هستند به ترتیب از SAS می‌خواهند که هر صفحه‌ی خروجی ۶۵ سطر و هر سطر خروجی شامل ۸۵ ستون باشد.

۱-۹-٤ عبارت GOPTIONS

هنگامی‌که کاربر به منظور تحلیل داده‌ها، نمودارهایی را در برنامه تقاضا می‌کند، می‌تواند با این عبارت پنجره نموداری SAS را تنظیم کند. این عبارت در فصل دوم که تحلیل اکتشافی دادها تشریح می‌شود معرفی خواهد شد.

۱-۱۰ مدیریت خروجی‌های SAS

خروجی‌های برنامه‌های SAS را می‌توان را با نظام حمل خروجی (ODS[3]) کنترل کرد. نظام حمل خروجی‌های SAS این امکان را به کاربر می‌دهد تا خروجی‌های SAS را به صورت مجموعه‌ی داده‌های SAS ذخیره کند. این نظام به کاربر اجازه می‌دهد تا نتیجه‌های پردازش داده‌ها با گام PROC یا حاصل از گام DATA را به صورت زیر درخواست کند.

۱- LISTING: خروجی متنی معمولی با قلم monospace.

۲- RTF[4]: خروجی را می‌توان با پردازش‌گر Word خواند.

۳- PRINTER: فایل چاپگری.

۴- HTML[5]: محتوا و صفحه‌های خروجی درون قالب و جدول قرار می‌گیرد.

۵- PDF[6]: خروجی به فورمت pdf تبدیل می‌شود.

۶- OUTPUT: خروجی به صورت یک فایل داده‌ای ارائه می‌شود.

۱-۱۰-۱ LISTING

پیش‌فرض SAS باز بودن مقصد LISTING و بسته بودن سایر مقصدها است. در این حالت، نتیجه‌های پردازش داده‌ها به صورت متنی معمولی عرضه می‌شوند. این مقصد با اجرای عبارت زیر بسته می‌شود.

```
ODS LISTING CLOSE;
```

[3] Output Delivery System

[4] Rich Text Format

[5] Hyper Text Markup Language

[6] Portable Document Format

این مقصد با اجرای عبارت ODS LISTING; دوباره باز می‌شود. در محیط ویندوز، محتوای مقصد LISTING را می‌توان با استفاده از گزینه‌های "File" و "Save As ..." ذخیره کرد.

RTF ۲-۱۰-۱

فایل ذخیره‌شده در زیر بخش ۱-۱۳-۱ را می‌توان با استفاده از "Save as type" به صورت RTF ذخیره کرد. این فورمت را می‌توان در پردازش‌گر Word ویرایش کرد. فایل RTF را برای نگهداشتن نتیجه‌های پردازش، می‌توان با اجرای عبارت

```
ODS RTF FILE= 'نام فایل.rtf';
```

فعال کرد و با اجرای عبارت ODS RTF CLOSE; آن را بست.

PRINTER ۳-۱۰-۱

اجرای عبارت ODS PRINTER FILE= 'نام فایل.ps'[۷]; سبب ایجاد فایل چاپگری برای نگهداشتن نتیجه‌های پردازش می‌شود. گزینه‌ی ps موجب ارسال نتیجه‌ی پردازش با فورمت پست‌اسکریپت به فایل 'ps. نام فایل' می‌شود. اجرای عبارت ODS PRINTER CLOSE; این مقصد را می‌بندد.

HTML ٤-۱۰-۱

اجرای عبارت ODS HTML BODY='bodyfile.htm' این مقصد را باز و اجرای ODS HTML CLOSE; آن را مسدود می‌کند. در صورت باز بودن این مقصد، نتیجه‌های پردازش‌ها در فایل 'htm. نام فایل' ذخیره می‌شوند.

PDF ٥-۱۰-۱

نوع فایل دیگری که می‌توان نتیجه‌های پردازش‌های SAS را در آن نگهداشت، فایل PDF است. اجرای عبارت ODS PDF FILE= 'نام فایل.pdf'; آن را باز کرده و تمام نتیجه‌های پردازش‌ها تا وقتی که این مسیر با عبارت ODS PDF CLOSE; بسته نشده است در آن نگهداشته می‌شود.

OUTPUT ٦-۱۰-۱

با استفاده از این گزینه در عبارت ODS می‌توان نام مجموعه‌ی داده‌های SASی که ایجاد شده‌اند را تغییر داد. شکل دستوری برای این کار به صورت زیر است:

```
ODS OUTPUT نام فعلی مجموعه‌ی داده‌ها=نام جدید مجموعه‌ی داده‌ها ;
```

[7] Post-Script

زمانی که این عبارت اجرا شود، عمل کامل می‌شود و نیاز به عبارتی برای بستن مسیر وجود ندارد.

۱-۱۱ تبدیل مجموعه‌ی داده‌ها به مجموعه داده‌های SAS

برای تحلیل داده‌ها با شیوه‌های SAS، ابتدا باید داده‌ها در هر قالبی که هستند به مجموعه‌ی داده‌های SAS تبدیل شوند. به علاوه، گاهی لازم است که مجموعه‌ی داده‌های SAS به قالبی تبدیل شود که سایر نرم‌افزارها مثل EXCEL، ACCESS، SPSS، و STATA بتوانند آن را پردازش کنند. شیوه‌های IMPORT و EXPORT به ترتیب برای تبدیل داده‌های با سایر قالب‌ها به مجموعه‌ی داده‌های SAS و تبدیل مجموعه‌ی داده‌های SAS به سایر قالب‌ها به کار می‌روند.

۱-۱۱-۱ شیوه‌ی IMPORT

این شیوه می‌تواند فایل‌های داده‌ای را با ساختارهای ویژه بخواند. این فایل‌ها می‌توانند فایل متنی باشند که در آن مقدار متغیرها با جداگری (مثل واوک) از هم جدا شده باشند یا فایل داده‌ای باشند که با نرم‌افزار دیگری مثل EXCEL تولید شده باشند. مثالی از فایل متنی که مقدار متغیرها با جداگر واوک از یکدیگر جدا شده‌اند، مجموعه‌ی داده‌های زیر است. سطر اول شامل نام متغیرها است.

مثال ۱-۲۱ مثالی از مجموعه‌ی داده‌ها با قالب جداگر واوک در زیر دیده می‌شود.

```
id,age,gender,height,weight
01,21,F,175,63
02,20,F,170,55
03,22,M,172,62
04,24,F,180,70
05,27,M,177,71
06,19,M,167,66
07,23,F,160,53
08,18,M,181,77
09,25,F,175,70
10,20,M,171,69
```

برای تبدیل این داده‌ها به مجموعه‌ی داده‌های SAS از شیوه‌ی IMPORT به صورت زیر استفاده می‌شود.

```
PROC IMPORT DATAFILE='آدرس فایلی که باید تبدیل شود'
     OUT=نام داده‌های تبدیل‌شده DBMS= قالب فایلی که باید تبدیل شود REPLACE;
     "گزینه‌ی مناسب برای قالب فایلی که باید تبدیل شود"
     GETNAMES=YES;
RUN;
```

در عبارت‌های بالا

- قالب فایلی که باید تبدیل شود در DBMS، می‌تواند DLM (جداگر)، EXCEL (XLS)، ACCESS (فایل‌های MS ACCESS)، DTA (فایل‌های STATA)، SAV (فایل‌های SPSS) یا قالبی دیگر باشد.
- گزینه‌ی REPLACE از SAS می‌خواهد اگر فایل داده‌ای با نامی که در =OUT مشخص شده است وجود داشته باشد، با فایلی که ایجاد می‌شود جایگزین شود.
- نام مجموعه‌ی داده‌های تبدیل‌شده در =OUT، نامی دو بخشی است. بخش اول شامل نام کتابخانه و بخش دوم شامل نام مجموعه‌ی داده‌ها است. اگر نام بخش اول حذف شود، مجموعه‌ی داده‌های SAS در کتابخانه‌ی WORK ایجاد می‌شود.
- «عبارت مناسب برای قالب فایلی که باید تبدیل شود» می‌تواند DELIMITER باشد اگر قالب فایل جداگرباشد، یا ; نام =SHEET باشد اگر قالب فایل EXCEL باشد.
- عبارت ;GETNAMES=YES از SAS می‌خواهد که نام متغیرها را از سطر اول دریافت کند. اگر این عبارت حذف شود، SAS به ترتیب نام‌های var1، var2، و ... را به متغیرها منتسب می‌کند.

مثال ۱-۲۲ برنامه‌ی زیر مجموعه‌ی داده‌های با فورمت جداگر واوک را تبدیل به مجموعه‌ی داده‌های SAS می‌کند.

```
PROC IMPORT DATAFILE='C:\mydata\fitness.dat'
   OUT=mydat.fitness   DBMS=DLM   REPLACE;
   DELIMITER=',';
   GETNAMES=YES;
RUN;
```

اجرای این برنامه، مجموعه‌ی داده‌های SAS به نام fitness را در کتابخانه‌ی mydat ذخیره می‌کند.

مثال ۱-۲۳ فرض کنید مجموعه‌ی داده‌های مثال ۱-۲۲ در قالب EXCEL2003 ذخیره شده باشد. برنامه‌ی زیر این فایل را خوانده، به مجموعه‌ی داده‌های SAS تبدیل کرده و در کتابخانه‌ی mydat به صورت دائمی ذخیره می‌کند.

```
PROC IMPORT DATAFILE='C:\mydata\fitness.xls'
   OUT=mydat.fitness   DBMS=EXCEL   REPLACE;
   GETNAMES=YES;
RUN;
```

مثال ۱-۲٤ فرض کنید مجموعه‌ی داده‌های مثال ۱-۲۲ در قالب MS ACCESS ذخیره شده باشد. برنامه‌ی زیر این فایل را به مجموعه‌ی داده‌ی SAS تبدیل می‌کند.

```
PROC IMPORT OUT=mydat.fitness
```

```
   DBMS=ACCESS REPLACE
   DATATABLE="Table1";
   DATABASE="C:\mydata\fitness.mdb";
RUN;
```

۱-۱۱-۲ شیوه‌ی EXPORT

تبدیل مجموعه‌ی داده‌های SAS به فایل داده‌ای که قابل خواندن توسط سایر نرم‌افزارها مثل EXCEL باشد، توسط شیوه‌ی EXPORT انجام می‌شود. نحوه‌ی استفاده از PROC EXPORT مشابه استفاده از PROC IMPORT است. مثال زیر نحوه‌ی استفاده از این شیوه را نشان می‌دهد.

مثال ۱-۲۵ فرض کنید مجموعه‌ی داده‌های SAS دائمی به نام fitness وجود دارد. برنامه‌ی زیر این مجموعه‌ی داده‌ها را به داده‌هایی در قالب EXCEL 2003 تبدیل کرده و در آدرس C:\mydata ذخیره می‌کند.

```
PROC EXPORT DATA=fitness DBMS=EXCEL2003
   OUTFILE='C:\mydata\fitness.xls'
   REPLACE;
RUN;
```

۱-۱۱-۳ تبدیل مجموعه‌ی داده‌ها با استفاده از منوها

تبدیل مجموعه‌ی داده‌ها در قالب‌های خاص به مجموعه‌ی داده‌های SAS و برعکس، می‌تواند با استفاده از نوار منو انجام شود. برای IMPORT یا EXPORT کردن مجموعه‌ی داده‌ها، روی File در نوار منو کلیک کرده و گزینه‌های Import Data یا Export Data را انتخاب کرده، گزینه‌های مناسب در جعبه‌هایی که باز می‌شود را دنبال کنید تا مجموعه‌ی داده‌های دلخواه ایجاد شود. پس از انتخاب قالب مورد نظر و گزینه‌های مناسب در آخرین پنجره‌ای که باز می‌شود می‌توانید نامی برای دستورات برنامه‌ای SAS متناظر با عملیات انجام‌شده انتخاب کرده و این دستورات را در برنامه‌های خود مورد استفاده قرار دهید.

۱-۱۲ زبان ماتریسی محاوره‌ای SAS

همان‌طور که در بخش‌های قبل گفته شد، نرم‌افزار SAS دارای شیوه‌های متعددی است که امکان کار با مجموعه‌ی داده‌ها و تحلیل‌های مختلف آماری را فراهم می‌کنند. یکی از شیوه‌های بسیار مفید در نرم‌افزار SAS، شیوه‌ی IML[8] است. در این شیوه مجموعه‌ی داده‌های SAS در قالب ماتریس، بردار یا اسکالر معرفی شده و عملیات ماتریسی مختلفی را می‌توان بر روی آن‌ها انجام داد.

[8] Interactive Matrix Language

شیوه‌ی IML با عبارت ;PROC IML آغاز و با ;QUIT خاتمه می‌یابد. مابین این دو عبارت، عبارت‌های برنامه‌ای قرار مـی‌گیرنـد که هر خط آن با ; پایان می‌یابد. با عبارت‌های برنامه‌ای می‌توان ماتریس‌ها را تعریف و یا یک مجموعه‌ی داده‌ها را فراخوانی و به ماتریس تبدیل کرد. همچنین می‌توان از عمل‌گرها و تابع‌های SAS برای کار با ماتریس داده‌ها استفاده کرد.

۱-۱۲-۱ تعریف یک ماتریس، بردار یا اسکالر

ماتریس‌ها، گونه‌ای از نمایش داده‌ها هستند که دارای سطر و ستون می‌باشند که تعداد سطرها و ستون‌ها، بعد ماتریس را می‌سازد. هر یک از مولفه‌های یک ماتریس می‌تواند عددی و یا نویسه‌ای باشد. قاعده‌های تعیین نام یک ماتریس کاملا مشابه با تعیین نـام یک مجموعه‌ی داده‌ها در SAS می‌باشد که در بخش ۱-۳ شرح داده شد.

برای تعریف یک ماتریس در شیوه‌ی IML مولفه‌های ماتریس داخل { } نوشته، مولفه‌های هر سطر با فاصله از هم جدا شده و بین سطرها نیز از واوک استفاده می‌شود. به عنوان مثال ماتریس X،

$$X = \begin{pmatrix} ۱ & 0 & ۳ \\ ۴ & ۱ & ۳ \\ 0 & ۳ & ۱ \end{pmatrix}_{۳\times۳}$$

یک ماتریس ۳×۳ است که به صورت زیر در شیوه‌ی IML تعریف می‌شود:

```
PROC IML;
   RESET PRINT;
   X = {1 0 3,
        4 1 3,
        0 3 1};
QUIT;
```

دستور ;RESET PRINT که در ابتدای شیوه‌ی IML می‌آید، باعث می‌شود تمام ماتریس‌های تعریف شده در شیوه چاپ شـوند. در صورتی‌که مایل به چاپ همه آن‌ها نباشیم، می‌تـوان از دسـتور PRINT در داخـل شـیوه‌ی IML بـه صـورت زیـر اسـتفاده و نـام ماتریس‌های مورد نظر را در مقابل آن ذکر کرد.

```
PROC IML;
   X = {1 0 3,
        4 1 3,
        0 3 1};
   PRINT X;
QUIT;
```

نکته‌ی ۱-٦ عبارت ;QUIT سبب می‌شود که SAS از محیط IML خارج شود. در صورتی‌که این عبارت و یـا شـیوه‌ی دیگـری اجرا نشود، کاربر هم‌چنان در محیط IML باقی می‌ماند.

برای معرفی ماتریس‌ها با مولفه‌های نویسه‌ای مشابه آنچه گفته شد، عمل می‌شود با این تفاوت که اگر هر مولفه را بین علامت ' ' قرار داده نشود، با حروف بزرگ چاپ می‌شود. برای مولفه‌های خالی و ویژه نیز باید از علامت ' ' استفاده کرد. به عنوان مثال

```
PROC IML;
   X = {'a' 'b' 'c',
         1   0   3,
        ''   1   3,
         0   3   1};
   PRINT X;
QUIT;
```

خروجی برنامه‌ی بالا به صورت زیر است:

X

a	b	c
1	0	3
	1	3
0	3	1

واضح است که بردار ماتریسی با سطر یا ستون ۱ است. بنا بر این تعریف بردار نیز کاملاً مشابه تعریف ماتریس است. در مثال زیـر X به صورت یک بردار سطری و Y یک بردار ستونی تعریف شده است.

```
PROC IML;
   X = {1 2 3 4 5 6};
   Y = {1,
        2,
        3,
        4,
        5};
   PRINT X, Y;
QUIT;
```

خروجی برنامه فوق در شکل زیر آمده است.

			X		
1	2	3	4	5	6
			Y		
			1		
			2		
			3		
			4		
			5		

برای تعریف یک مقدار اسکالر نیازی با استفاده از نماد { } نیست. تنها کافی است مانند مثال زیر به آن نامی اختصاص دهید:

```
PROC IML;
   K = 3;
   PRINT K;
QUIT;
```

برای اینکه هر یک از سطرها یا ستون‌های ماتریس نامی داشته باشند، ابتدا برای سطرها یا ستون‌های ماتریس، می‌توان نـام‌هـا را به صورت یک بردار مشخص و از آن در زمان چاپ ماتریس مورد نظر در دستور PRINT به صورت زیر استفاده کرد.

```
PROC IML;
   X = {1 0 3,
        4 1 3,
        0 3 1};
   Names = {A1,
            A2,
            A3};
   PRINT X [ROWNAME = names];
QUIT;
```

خروجی برنامه‌ی بالا به صورت زیر است:

		X	
A1	1	0	3
A2	4	1	3
A3	0	3	1

شکل ۱-۱۱ نمایشی از ماتریس سطر نام‌داده شده

اگر بخواهیم، به ستون‌ها نامی اختصاص دهیم، در برنامه‌ی بالا به جای ROWNAME باید از گزینه‌ی COLNAME استفاده کنید.

مثال‌های بالا، روش معرفی ماتریس داده‌ها را در برنامه نشان می‌دهد. گاهی با مجموعه‌ی داده‌های بزرگ سر و کار داریم که با استفاده ار روش‌های معرفی‌شده در بخش‌های قبل در یکی از کتابخانه‌های SAS ذخیره شده‌اند. شما پردازش این مجموعه‌ی داده‌ها باید آن‌ها را در شیوه‌ی IML فراخوانی کرد. برای این‌کار می‌توان از دستور USE به صورت زیر استفاده کرد.

USE <نام مجموعه‌ی داده‌ها> <VAR <نام متغیرهایی که می‌خواهید فراخوانی کنید> <WHERE (عبارت شرطی)>;

در این دستور پس از درج نام دو بخشی مجموعه‌ی داده‌ها (شامل نام کتابخانه و نام مجموعه‌ی داده‌ها)، نام متغیرهایی که مورد نظر است وارد می‌شوند. در صورتی که متغیری مشخص نشود، SAS تمام متغیرها را فراخوانی می‌کند. از عبارت WHERE برای محدود کردن مجموعه‌ی داده‌ها با استفاده از یک شرط استفاده می‌شود.

مثال ۱-۲۶ اگر مجموعه‌ی داده‌های class که در کتابخانه Sashelp وجود دارد را باز کنید، خواهید دید این مجموعه‌ی داده‌ها، اطلاعات دانش‌آموزان یک کلاس شامل پنج متغیر name، sex، age، height و weight را در خود دارد. دستور زیر مجموعه‌ی داده‌های class را به طور کامل می‌خواند:

```
PROC IML;
   USE sashelp.class;
   SHOW CONTENTS;
QUIT;
```

عبارت ;show contents اطلاعات مربوط به متغیرها از جمله نام متغیرها، نوع و اندازه‌ی هر کدام را نمایش می‌دهد. اگر فقط اطلاعات مربوط به نام و جنسیت دانش‌آموزان مونث موردنظر باشد، می‌توان از دستور زیر استفاده کرد.

```
PROC IML;
   USE sashelp.class VAR{Name, Sex} WHERE(Sex = 'F');
   SHOW CONTENTS;
QUIT;
```

دستور READ مجموعه‌ی داده‌های فراخوانی شده به IML را به صورت یک ماتریس معرفی می‌کند. با استفاده از این دستور می‌توان تمام یا بخشی از اطلاعات فراخوانی شده را به ماتریس یا ماتریس‌های مختلف عددی یا نویسه‌ای تخصیص داد. شکل کلی این دستور به صورت زیر است:

```
Read ALL var{نام متغیرهای موردنظر} _نوع متغیرها_ INTO <نام ماتریس جدید>
      <WHERE(عبارت شرطی)>;
```

نوع متغیر می‌تواند گزینه‌های _NUM_ یا _CHAR_ را اختیار کند. اگر نام متغیرها معرفی نشوند و فقط نوع عبارت وارد شود، IML تمام متغیرها از آن نوع را نگه می‌دارد.

به عنوان مثال با استفاده از برنامه‌ی زیر می‌توان مجموعه‌ی داده‌های class را به دو ماتریس جدید به نام‌های men و women تبدیل کرد که یکی شامل اطلاعات مردان و دیگری شامل اطلاعات زنان است.

```
Read  ALL  VAR{height, weight} INTO  women WHERE(sex='F');
Read  ALL  VAR{height, weight} INTO  men   WHERE(sex='M');
```

۱-۱۲-۲ عمل‌گرها و تابع‌های ماتریسی

در بخش قبل نحوه‌ی معرفی ماتریس‌ها به دو روش شرح داده شد. ماتریس‌ها برای این به محیط IML معرفی می‌شوند که عملیاتی بر روی آن‌ها انجام شود. عملیات ماتریسی متنوعی وجود دارد که هر کدام به فراخور مسئله‌ی مورد بررسی اهمیت پیدا کرده و مورد استفاده قرار می‌گیرند. برای انجام این عملیات می‌توانید به سادگی از عمل‌گرها و تابع‌های از پیش تعریف شده در IML استفاده کرد. اما گاهی هم ناچار خواهیم بود با ترکیب چند عمل‌گر و تابع، عملیات ماتریسی مورد نظر خود را انجام دهیم. برای مثال یافتن وارون یک ماتریس را می‌توان با استفاده از دستور INV محاسبه کرد. اما برای محاسبه‌ی آماره‌ای مانند T^2 هتلینگ باید از ترکیب چند عمل‌گر و تابع استفاده کرد. در این بخش ابتدا عمل‌گرهای ماتریسی و پس از آن توابع IML را معرفی می‌کنیم. به این منظور ابتدا سه ماتریس زیر را در نظر بگیرید.

$$A=\begin{pmatrix}۲ & ۲\\ ۳ & ۴\end{pmatrix}_{۲\times۲} \quad , \quad B=\begin{pmatrix}۴ & ۵\\ 0 & ۱\end{pmatrix}_{۲\times۲} \quad , \quad C=\begin{pmatrix}۱ & ۱ & 0\\ 0 & ۱ & ۱\end{pmatrix}_{۲\times۳}$$

جدول زیر، مهم‌ترین عمل‌گرهای مقدماتی بین ماتریس‌های بالا را به همراه خروجی مربوط را نشان می‌دهد.

عمل‌گر	کاربرد	مثال	خروجی
+	جمع کردن مولفه‌های متناظر دو ماتریس	M1=A+B;	$M_۱=\begin{pmatrix}۶ & ۷\\ ۳ & ۵\end{pmatrix}$
-	تفریق کردن مولفه‌های متناظر دو ماتریس	M2=A-B;	$M_۲=\begin{pmatrix}-۲ & -۳\\ ۳ & ۳\end{pmatrix}$
*	ضرب ماتریسی دو ماتریس	M3=A*C;	$M_۳=\begin{pmatrix}۴ & ۹ & ۵\\ ۰ & ۱ & ۱\end{pmatrix}$

#	ضرب مولفه‌های یک ماتریس در مولفه‌های متناظر یک ماتریس دیگر	M4=A#B;	$M_{۴}=\begin{pmatrix}۸ & ۱۰\\ ۰ & ۴\end{pmatrix}$
##	به توان رساندن مولفه‌های یک ماتریس	M5=B##2;	$M_{۵}=\begin{pmatrix}۱۶ & ۲۵\\ ۰ & ۱\end{pmatrix}$
/	تقسیم مولفه‌های یک ماتریس بر مولفه‌های متناظر یک ماتریس دیگر	M6=A/B;	$M_{۶}=\begin{pmatrix}۰/۵ & ۰/۴\\ ۰ & ۴\end{pmatrix}$
||	قرار دادن دو ماتریس در کنار هم	M7=A||B;	$M_{۷}=\begin{pmatrix}۲ & ۲ & ۴ & ۵\\ ۳ & ۴ & ۰ & ۱\end{pmatrix}$
//	قرار دادن دو ماتریس به صورت عمودی	M8=A//B;	$M_{۸}=\begin{pmatrix}۲ & ۲\\ ۳ & ۴\\ ۴ & ۵\\ ۰ & ۱\end{pmatrix}$

جدول ۱-۱

عمل‌گرهای فوق، عملی بین دو ماتریس را انجام می‌دهند. در جدول زیر نحوه‌ی کار چند عمل‌گر دیگر بر روی یک ماتریس نشان داده شده است.

عمل‌گر	کاربرد	مثال	خروجی
+	مجموع مقدارهای مؤلفه‌های ماتریس	M1=A[+];	۱۱
<>	ماکسیمم مقدار مؤلفه‌های ماتریس	M2=A[<>];	۴
><	مینیمم مقدار مؤلفه‌های ماتریس	M3=A[><];	۲
:	میانگین مؤلفه‌های ماتریس	M4=A[:];	۲/۷۵
#	حاصل‌ضرب مؤلفه‌های ماتریس	M5=A[#];	۴۸
##	مجموع توان‌های دوم مؤلفه‌های ماتریس	M6=A[##];	۳۳
[n1:nk,m1:mk]	سطر n_1 تا n_k و ستون‌های m_1 تا m_k ماتریس را جدا می‌کند.	M7=C[1:2,2:3];	$M_{۷}=\begin{pmatrix}۱ & ۱\\ 0 & ۱\end{pmatrix}$

جدول ۱-۲

آخرین عمل‌گر جدول بالا، بخشی از یک ماتریس را جدا می‌کند و به صورت یک ماتریس جدید در می‌آورد. می‌توان این عمل‌گر را با عمل‌گرهای دیگر ترکیب کرد. قاعده‌ی این ترکیب به این صورت است که برای انجام یک عملیات بر روی سطر خاصی، ابتدا آن را جدا کرده و در قسمت بعد عمل‌گر معرفی می‌شود. شکل کلی این عمل‌گرهای ترکیبی سطری به صورت زیر است.

```
A[n1: nk, عمل‌گر];
```

به طور مشابه برای انجام عملیات ستونی، عمل‌گر ترکیبی ستونی به‌صورت زیر است.

```
A[عمل‌گر, m1: mk];
```

به عنوان مثال، میانگین مولفه‌های ستون دوم ماتریس A با استفاده از عبارت A[2,:] به دست می‌آید که حاصل آن برابر با ۳/۵ است. به طور مشابه، عبارت A[+,2]، مجموع مقدارهای ستون دوم ماتریس A را محاسبه می‌کند که ۶ می‌شود.

نکته‌ی ۱-۷ اگر سطر یا ستونی مشخص نشود و جای آن را خالی باشد، SAS به‌صورت پیش‌فرض تمامی سطرها یا ستون‌های ماتریس را انتخاب می‌کند.

جدول زیر نیز چند تابع مهم ماتریسی را به همراه مثال و خروجی آن‌ها در شیوه‌ی IML را معرفی می‌کند.

عمل‌گر	کاربرد	مثال	خروجی
T( )	محاسبه‌ی ترانهاده‌ی ماتریس	`M1 = T(A);`	$M_{۱}=\begin{pmatrix}۲ & ۳\\ ۲ & ۴\end{pmatrix}$
DET( )	محاسبه‌ی دترمینان ماتریس	`M۲ = DET(A);`	۲
INV( )	وارون ماتریس	`M3 = INV(A);`	$M_{۳}=\begin{pmatrix}۲ & -۱\\ -۱/۵ & ۱\end{pmatrix}$
DIAG( )	محاسبه‌ی ماتریس قطری از روی یک ماتریس	`M4 =DIAG(A);`	$M_{۴}=\begin{pmatrix}۲ & ۰\\ ۰ & ۴\end{pmatrix}$
J(n,m,k)	ساختن ماتریس $n\times m$ با مولفه‌های k	`M5 =J(3,2,5);`	$M_{۵}=\begin{pmatrix}۵ & ۵\\ ۵ & ۵\\ ۵ & ۵\end{pmatrix}$
I(n)	ساختن ماتریس همانی $n\times n$	`M6 =I(2);`	$M_{۶}=\begin{pmatrix}۱ & ۰\\ ۰ & ۱\end{pmatrix}$
EIGVAL( )	محاسبه‌ی ویژه‌مقدارهای ماتریس	`M7=EIGVAL(A)`	5.6457513 0 0.3542487 0
EIGVEC( )	محاسبه‌ی ویژه‌بردارهای ماتریس	`M8=EIGVEC(A)`	0.4809652 -0.772178 0.8767397 0.6354064

عمل‌گر	کاربرد	مثال	خروجی

جدول ۱-۳ نمایشی از کاربرد چند تابع ماتریسی

۱-۱۲-۳ آماره‌های خلاصه در IML

اگر بخواهید آماره‌های خلاصه‌ی یک یا چند ستون یک ماتریس را مشاهده کنید می‌توانید از دستور SUMMARY که شکل کلی آن به صورت زیر است، استفاده کنید.

```
SUMMARY <VAR {ستون‌های مورد نظر}> <CLASS{ستون‌های گروه‌بندی شده} > <STAT{آماره‌ها}>
        <WHERE(عبارت شرط)>;
```

در عبارت VAR نام ستون‌هایی را نوشته می‌شوند که آماره‌های خلاصه‌ی آن‌ها موردنظر است. اگر لازم باشد این آماره‌ها به تفکیک زیرگروه‌هایی از یک یا چند ستون محاسبه شوند، در عبارت CLASS نام ستون‌هایی که گروه‌بندی شده هستند را، نوشته می‌شوند. این دستور آماره‌های مختلفی را محاسبه می‌کند که عمده‌ترین آن‌ها در جدول زیر معرفی شده‌اند.

شرح آماره	آماره
مجموع توان‌های تصحیح‌شده	CSS
مقدار ماکسیمم	MAX
میانگین	MEAN
مقدار مینیمم	MIN
تعداد مشاهده‌ها	N
تعداد مشاهده‌های گم‌شده	NMISS
انحراف استاندارد	STD
مجموع	SUM
مجموع توان‌های دوم تصحیح‌نشده	USS
واریانس	VAR

جدول ۱-۴

اگر عبارت STAT و آماره‌های مورد نظر مشخص نشوند، SAS به‌صورت پیش‌فرض آماره‌های MAX، MEAN، MIN، و STD را محاسبه و چاپ می‌کند.

از گزینه‌ی WHERE وقتی استفاده می‌شود که کاربر بخواهد دستور مورد نظر تنها برای بخشی از جامعه انجام شود. لذا، با نوشتن عبارت شرطی در مقابل WHERE محاسبه‌ی آماره‌های خلاصه محدود به همان زیر بخش می‌شود.

مثال ۱-۲۷ برنامه‌ی زیر، آماره‌های میانگین و انحراف استاندارد را برای متغیرهای قد و وزن دانش‌آموزان به تفکیک جنس را برمی‌گرداند:

```
PROC IML;
   USE sashelp.class;
   SUMMARY VAR{height weight} CLASS{sex} STAT{MEAN STD};
QUIT;
```

خروجی این برنامه به صورت زیر است.

SEX	Nobs	Variable	MEAN	STD
		F		
F	9	HEIGHT	60.58889	5.01833
		WEIGHT	90.11111	19.38391
		M		
M	10	HEIGHT	63.91000	4.93794
		WEIGHT	108.95000	22.72719
		All		
All	19	HEIGHT	62.33684	5.12708
		WEIGHT	100.02632	22.77393

شکل ۱-۱۲ خروجی مثال ۱-۲۷

در ستون دوم خروجی، Nobs، مشاهدات به تفکیک هر زیر گروه را می‌دهد که همواره در خروجی این دستور چاپ می‌شود.

۱-۱۲-٤ ساختن مجموعه‌ی داده‌ها در IML

گاهی نیاز داریم ماتریس‌هایی که در شیوه‌ی IML ساخته می‌شوند را به صورت یک مجموعه‌ی داده‌های SAS در اختیار داشته باشیم. برای مثال مجموعه‌ی داده‌های ایجاد شده در IML برای استفاده در سایر شیوه‌های SAS مورد نیاز هستند. بنا بر این، باید برای ماتریس‌های ایجاد شده، یک نام مطابق با قاعده‌ی نام‌گذاری انتخاب کرده و برای ستون‌های ماتریس مورد نظر که در مجموعه‌ی داده‌های جدید متغیرها هستند نیز نام‌هایی انتخاب کرد. دستور زیر یک ماتریس در شیوه‌ی IML را به یک مجموعه‌ی داده‌های SAS تبدیل می‌کند:

```
CREATE نام مجموعه‌ی داده‌های جدید FROM نام ماتریس
        <[COLNAME=بردار نام ستون‌ها ROWNAME=بردار نام سطرها]>;
APPEND FROM نام ماتریس ;
```

مثال ۱-۲۸ فرض کنید می‌خواهیم ماتریس women را که در بخش ۱-۱۱-۱ ساخته شد، به یک مجموعه‌ی داده‌های SAS به نام stuwomen تبدیل کنیم و ستون‌های آن را به ترتیب we و he نام‌گذاری کنیم.

```
PROC IML;
   USE sashelp.class;
   Read ALL VAR{height, weight} INTO women WHERE(sex = 'F');
   name = {we, he};
   CREATE stuwomen FROM women [COLNAME = name];
   APPEND FROM women;
QUIT;
```

با اجرای برنامه‌ی بالا، مجموعه‌ی داده‌هایی به نام stuwomen در کتابخانه‌ی Work ایجاد خواهد شد.

در این بخش، مقدمات کار با ماتریس‌ها ارائه شد، هر چند انجام بسیاری از تحلیل‌های آماری بی‌نیاز از شیوه‌ی IML است اما اهمیت کار کردن با ماتریس‌ها غالبا در مسائل پیشرفته آماری مانند مثال زیر خود را نشان می‌دهد.

مثال ۱-۲۹ با استفاده از تابع‌های تعریف‌شده ضریب کاپا را برای ماتریس X به‌دست می‌آوریم.

```
PROC IML;
   /* Enter table of counts */
   x = {17  4  8,
         5 12  0,
        10  3 13};
   /* compute number of rows and columns for x */
   nr = NROW(x);
   nc = NCOL(x);

   /* compute row and column totals */
   xr = x(|, +|);
   xc = x(|+, |);

   /* compute table of expected counts for independent
      randomgrement */
   e = xr * xc;

   /* compute overall total count */
   t = SUM(x);

   /* compute kappa */
   k1 = SUM(DIAG(x)) / t;
   k2 = SUM(DIAG(e)) / (t ** 2);
```

```
   K3 = SUM(DIAG(x) * DIAG(xr + t(xc))) / (t ** 2);
   j1 = J(1, nr);
   j2 = J(nc, 1);
   tt1 = ((t(xc) * j1 + j2 * t(xr)) ## 2) # X;
   k4 = SUM(tt1) / (t ** 3);
   kappa = (k1 - k2) / (1 - K2);

   /* print results */
   PRINT , , kappa;
QUIT;
```

خروجی برنامه‌ی بالا به صورت یک عدد است که ضریب را نشان می‌دهد. برای ماتریس X ضریب کاپا برابر با ۰/۳۶۲۳ است.

تمرین‌ها

۱- بردار داده‌های برنامه‌ای یک گام DATAی SAS دارای متغیرهایی با مقدارهای زیر است:

Code='VLC'

Size='M'

V1=2

V2=3

V3=7

V4= .

نتیجه‌ی عبارت‌های زیر را تعیین کنید:

a. (V1 + V2 – V3)/3

b. V3 – V2 / V1

c. V1 * V2 – V3

d. V2 * V3/V1

e. V1 * * 2 + V2 * * 2

f. code = 'VLc'

g. code = 'VLc' & Size = 'M'

h. code = 'VLc' | Size = 'M'

i. code = 'cLV' & V4 = .

j. (V3 = .) + (V2 = 3)

k. (V1 + V2 + V3)^ = 12

l. code = ' VLc' | (size = 'M' & V1=3)

m. 3 < V2 < 5

راهنمایی: به خاطر بیاورید که عبارت‌های منطقی برای حالت 'True' مقدار یک و برای حالت 'False' مقدار صفر را در نظر می‌گیرند.

۲- یک گام DATA، خطوط داده‌های زیر را می‌خواند.

1 ƀ ƀ 1.1 ƀ 2 ƀ 2.1 ƀ 3 ƀ 3.1

4. ƀ 4.1

5.

5.1

شکل مجموعه‌ی داده‌های SAS در صورتی‌که عبارت INPUT گام DATA یکی از حالت‌های زیر باشد، چیست؟

a. `input x y;`

b. `input x y @@;`

c. `input x;`

d. `input x y z;`

تذکر: نماد ␣ نماینده‌ی ستون خالی است.

۳- در هر یک از حالت‌های زیر مشاهده‌هایی را که در مجموعه‌ی داده‌های SAS، وقتی خطوط داده‌ها توسط عبارت INPUT خوانده می‌شود و سپس نوشته می‌شوند را نشان دهید (نام متغیرها و مقدار داده‌ای متناظر با آن‌ها).

A.
```
INPUT id gender $ age height weight;
101␣M␣23␣68␣155
102␣F␣.␣61␣␣102
103␣␣M␣␣55␣␣␣70␣␣␣202
```

B.
```
INPUT id dob : mmddyy8. dx;
1␣10/21/86␣256.20
2␣9/15/84␣232.4
```

C.
```
INPUT id $ 1-4 @8 pulse 3. +2 weight 4.1 pushups;
L4WP236␣85␣92517␣28
M5XQ␣47␣␣␣␣␣1423␣32
␣␣␣␣158␣79␣81145␣19
```

D.
```
INPUT name : $12. Age 2. Income 6.2;
Ahmadi ␣26␣53740
Hoseinzadehha␣34276520
```

۴- مجموعه‌ی داده‌هایی تحت عنوان PROB_4.DAT در آدرس "C:\TEMP\DATA" ذخیره شده است. برنامه‌ای با کد SAS بنویسید تا یک مجموعه‌ی داده‌های دائمی SAS به نام PROB_4 در کتابخانه‌ی DATASETS بسازد. به علاوه یک خروجی از آن بگیرید به طوری که:

الف) خروجی در آدرس "A:\OUTPUT" به صورت HTML ذخیره شود،

ب) به مقادیر متغیر SEX برچسب‌های F ⟶ FEMALE و M ⟶ MALE نسبت داده شود،

پ) شامل تاریخ نباشد،

ت) شماره‌ی صفحه در خروجی از ۱ شروع شود،

ح) ترتیب متغیرها به صورت NAME، SEX، X_2 و X_1 باشد، و

د) متغیر id متغیر شناسایی باشد.

داده‌های این مسئله در پیوست۱ با نام PROB_4.DAT آورده شده است.

۵- خروجی ناشی از اجرای برنامه‌ی SAS زیر را ترسیم کنید. جریان عملیاتی را که گام DATA برای ایجاد این مجموعه‌ی داده‌ها طی می‌کند، تشریح کنید.

```
DATA SET1;
      INPUT score@@;
      IF  score>50 THEN DO;
            control ='y';
            index=score-50;
      END;
      ELSE IF score<=50 THEN DO;
            control='n';
      END;
DATALINES;
47 49 50 52 58 .
;
RUN;
```

۶- مقدارهای متغیر kilometers که در مجموعه‌ی داده‌های SAS به نام distance ذخیره می‌شود را نشان دهید.

```
DATA distance;
     INPUT kilometers 5.2;
DATALINES;
1
12
123
1234
12345
1.
1.2
12.
123.4
;
RUN;
```

۷- یک گام DATAی SAS بنویسید که مجموعه‌ی داده‌های SAS به نام corn با متغیرهای variety و yield با داده‌های ورودی زیر ایجاد کند.

```
A␣24.2␣␣B␣31.5␣␣B␣32.0␣␣C␣43.9
```

␣␣C␣45.2␣␣A␣21.8

␣␣B␣36.1␣␣A␣27.2␣␣C␣34.6

۸- گام DATAی SAS زیر را در نظر بگیرید:

```
DATA set1;
      INPUT type c1 c2;
DATALINES;
4 0 1
6 2 1
. 0 0
;

PROC PRINT;
RUN;
```

اگر هر مجموعه از عبارت‌های زیر بین عبارت‌های INPUT و DATALINES قرار گیرند، خروجی اجرای برنامه را نمایش دهید.

A.
```
index = (2*C1) + C2;
```

B.
```
IF type <= 5 THEN DO;
      index = (2*C1) + C2;
      OUTPUT;
  END;
```

C.
```
IF type <= 5 THEN DO;
      index = (2*C1) + C2;
   END;
   ELSE DELETE;
```

D.
```
IF type > 5 THEN DELETE;
```

E.
```
IF type > 5 THEN DELETE;
      index = (2*C1) + C2;
```