The DATA statement

data dataset-name ;

or
data _null_;

- _NULL_ option allows you to execute a DATA step without creating a SAS data set.
- used together with FILE statement and PUT statement to output data to a file or to the output window

Example:

DATA _NULL_; 
  INFILE "/group/ftp/pub/kcowles/datasets/patients.dat" PAD;
  FILE PRINT; ***Send output to the output window;
  * FILE "/space/kcowles/172/Clean/errorlist.txt" ;
  TITLE "Listing of Missing Values";
  ***Note: We will only input those variables of interest;
  INPUT $1 PATHO $1.
  $5 VISIT MMDDYY10.
  $15 HR 3.
  $27 AE $1;
  IF VISIT = . THEN PUT "Missing or invalid visit date for ID " PATHO;
  IF HR = . THEN PUT "Missing or invalid HR for ID " PATHO;
  IF AE = ' ' THEN PUT "Missing or invalid AE for ID " PATHO;
  RUN;

Creates the following output in output window or in named file:

Missing or invalid visit date for ID 007
Missing or invalid HR for ID 010
Missing or invalid visit date for ID 011
Missing or invalid AE for ID 013
Missing or invalid visit date for ID 015
Missing or invalid visit date for ID 123
Missing or invalid visit date for ID 321
Missing or invalid visit date for ID 020
Missing or invalid visit date for ID 027
Missing or invalid HR for ID 027
Missing or invalid HR for ID 029

The INFILE statement

INFILE file-specification <options> ;

Options on an infile statement

- Note: you do not use a slash before the options on an INFILE statement
- DELIMITER
  - specifies a delimiter for list input.
  - default delimiter is blank space
  - delimiters can be expressed as a list of delimiting characters or as a character variable.
  * 'list-of-delimiting-characters’

  Specifies one or more (up to 200) characters to be read as delimiters. The list of characters must be enclosed in quotes.
* character-variable

Specifies a character variable whose value
is to be used as the delimiter.

```sas
example
data example;
infile 'c:\temp\myfile.dat' dim = ',';
* Comma-delimited file;
* infile 'c:\temp\myfile.dat' dim = '\09'x;
* tab-delimited file;
.
.
run;
```

-DSD

* Changes how SAS treats delimiters when
  list input is used and sets the default de-
  limiter to a comma.
* When you specify DSD, SAS treats two
  consecutive delimiters as a missing value.

-EXPANDTABS | NOEXPANDTABS

* Specifies whether to expand tab charac-
  ters to the standard tab setting, which
  is set at 8-column intervals starting at
column 9.
* NOEXPANDTABS is the default.
* EXPANDTABS is useful if you are read-
ing data that contain the tab character
native to your operating environment.

-FIRSTOBS = record-number

* Specifies a record number that SAS uses
  to begin reading input data records in
  the input file.
  * Default is 1.

-FLOWOVER

* Causes an INPUT statement that uses
  list input to continue to read the next
  input data record if it does not find val-
  ues in the current input line for all the
  variables in the statement.
* This is the default behavior of the IN-
  PUT statement.
* opposite of MISSOVER

-LRECL=record-length

* specifies the record length (in bytes).
* Under Windows, the default is 256. (You
don’t have to use LRECL for input data
files with record lengths shorter than de-
fault.)
* The value of record-length can range from
  1 to 1,048,576 (1 megabyte).

-MISSOVER

* Prevents a SAS program from going to a
  new input line if, when using list input,
  it does not find values in the current line
  for all the INPUT statement variables.
* When an INPUT statement reaches the
  end of the current record, variables with-
  out any values assigned are set to miss-
ing.

-OBS= record-number

* Specifies the record number of the last
  record to read in an input file that is
  read sequentially.
* You can use the OBS= and the FIRSTOBS=
  options together to read a range of records
  from the middle of a file.

-PAD | NOPAD

* Controls whether records read from an
  external file are padded with blanks to
  the length specified in the LRECL= op-
tion.
* NOPAD is the default.

- PRINT|NOPRINT
- Specifies whether the input file contains carriage control characters.
- NOPRINT is the default.
- Using the PRINT option allows you to use a print file as input to a DATA step without removing the carriage control characters.
- You don’t need this much with SAS under Windows, but it is very useful when running SAS under Unix but using data files created in Windows.

- STODOVER
- Stops processing the DATA step when an INPUT statement using list input reaches the end of the current record without finding values for all variables in the state-

Example: fake dataset

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>03/18/72</td>
<td>$5.00</td>
</tr>
<tr>
<td>May</td>
<td>02/24/89</td>
<td>$10.00</td>
</tr>
<tr>
<td>Helen</td>
<td>03/30/77</td>
<td></td>
</tr>
<tr>
<td>Doris</td>
<td>04/18/98</td>
<td>$15.00</td>
</tr>
<tr>
<td>Gerald</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- When an input line does not contain the expected number of values, SAS sets _ERROR_ to 1, stops building the data set as though a STOP statement has executed, and prints the incomplete data line.

- TRUNCOVER
- Overrides the default action of the INPUT statement when an input record is not as long as expected by the INPUT statement.
- By default, the INPUT statement reads the next record automatically.
- TRUNCOVER causes the DATA step to continue processing and to write any available values to the variables in question.
- The TRUNCOVER option enables you to read variable-length records when some records are shorter than expected by the INPUT statement.

Four ways to try to read it
1. with flowover

```sas
options linesize = 75 ;
data fake ;
infile 'fakedat.txt' flowover ;
input name $ @10 date mmdyy8. value dollar8. ;
format date mmdyy8. ;
run ;

proc print data = fake ;
title 'flowover' ;
run ;
```

From log file

NOTE: The infile 'fakedat.txt' is:
   File Name=/space/kowles/172/Lects/fakedat.txt,
   Owner Name=UNKNOWN,Group Name=UNKNOWN,
   Access Permission=rw-------.
   File Size (bytes)=111

NOTE: Invalid data for value in line 4 1-8.
RULE: -------1--------2--------3--------4--------5---
NOTE: The file 'fakedat.txt' is:

File Name=/space/kcowens/172/Lects/fakedat.txt.
Owner Name=UNKNOWN,Group Name=UNKNOWN,
Access Permission=rw-------.
File Size (bytes)=111

NOTE: 5 records were read from the file 'fakedat.txt'.
The minimum record length was 6.
The maximum record length was 28.
NOTE: The data set WORK.FAKE has 3 observations and 3 variables
NOTE: DATA statement used:
real time 0.02 seconds
cpu time 0.02 seconds

4
Doris 04/18/98 $15.00 28
name=Helen date=03/30/77 value=. _ERROR_=1 _N_=4
NOTE: LOST CARD.
name=Gerald date= value=_ERROR_=1 _N_=4

NOTE: 5 records were read from the file 'fakedat.txt'.
The minimum record length was 6.
The maximum record length was 28.
NOTE: SAS went to a new line when INPUT statement reached past
a line.
NOTE: The data set WORK.FAKE has 3 observations and 3 variables
NOTE: DATA statement used:
From proc print
Flows over 15:20 Saturday, J
Obs name date value
1  Joe 03/18/72 5
2  May 02/24/89 10
3 Helen 03/30/77 .

5
Doris 04/18/98 $15.00 28
name=Helen date=03/30/77 value=. _ERROR_=1 _N_=3
NOTE: LOST CARD.
name=Gerald date= value=_ERROR_=1 _N_=4
2. with missover
data fake;
infile 'fakedat.txt' missover;
input name $10 date mmddyy8. value dollar8. ;
format date mmddyy8. ;
run;
proc print data = fake;
title 'missover';
run;
From log file
NOTE: The infile 'fakedat.txt' is:

File Name=/space/kcowens/172/Lects/fakedat.txt.
Owner Name=UNKNOWN,Group Name=UNKNOWN,
Access Permission=rw-------.
File Size (bytes)=111

NOTE: 5 records were read from the file 'fakedat.txt'.
The minimum record length was 6.
The maximum record length was 28.
NOTE: The data set WORK.FAKE has 3 observations and 3 variables
NOTE: DATA statement used:
real time 0.02 seconds
cpu time 0.02 seconds

3. with truncanover
data fake;
infile 'fakedat.txt' truncanover;
input name $10 date mmddyy8. value dollar8. ;
format date mmddyy8. ;
run;
proc print data = fake;
title 'truncanover';
run;

Log file
NOTE: The infile 'fakedat.txt' is:

File Name=/space/kcowens/172/Lects/fakedat.txt.
Owner Name=UNKNOWN,Group Name=UNKNOWN,
Access Permission=rw-------.
File Size (bytes)=111

NOTE: 5 records were read from the file 'fakedat.txt'.
The minimum record length was 6.
The maximum record length was 28.
NOTE: The data set WORK.FAKE has 5 observations and 3 variables
From proc print

```
truncover

<table>
<thead>
<tr>
<th>Obs</th>
<th>name</th>
<th>date</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe</td>
<td>03/18/72</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>May</td>
<td>02/24/89</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Helen</td>
<td>03/30/77</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>Doris</td>
<td>04/18/98</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>Gerald</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

4. with **stopover**

data fake;
in file 'fakedat.txt' stopover;
input name $ @10 date mmdyy8. value dollar8.;
format date mmdyy8.;
run;

cproc print data = fake;
title 'stopover';
run;

Log file

```
RULE: 00001------2--------3--------4--------5-----
3  Helen 03/30/77 17
name=Helen date=03/30/77 value= _ERROR_=1 _N_=3
NOTE: 3 records were read from the in file 'fakedat.txt'.
The minimum record length was 17.
The maximum record length was 28.
NOTE: The SAS System stopped processing this step because of er
```

Using **INFILE** options with data inside data step

Put **DATALINES** as file name in **INFILE** statement

Example

data morefake;
in file datalines missover;
in put name $ 010 date mmdyy8. value dollar8.;
datalines;
Joe 03/18/72 $5.00
May 02/24/89 $10.00
Helen 03/30/77
Doris 04/18/98 $15.00
Gerald ;

NOTE: SAS set option OBS=0 and will continue to check statement
This may cause NOTE: No observations in data set.
WARNING: The data set WORK.FAKE may be incomplete. When this s stopped there were 2 observations and 3 variables.
WARNING: Data set WORK.FAKE was not replaced because this step
NOTE: DATA statement used:
real time 0.01 seconds
cpu time 0.02 seconds

From proc print: NOTHING!
Using the FILE and PUT statements to write data to a plain text file

- plain text files are readable by most software packages

Formatting data written to a text file

- PUT statement can use any methods for specifying columns or formats that are valid in INPUT statement

Example:

```plaintext
Example
data poverty ;
infile 'C:\temp\poverty.dat' ;
length Country $ 20 ;
input liveBirth dthrate infdth life@pm life@spf pcgnp group Country ;
file 'C:\temp\partialpov.dat' ;
put Country liveBirth infdth ;
run ;
```

produces a file `c:\temp\partialpov.dat` that begins

Albania 24.7 30.8
Bulgaria 12.5 14.4
Czechoslovakia 13.4 11.3
Former_E._Germany 12 7.6
Hungary 11.6 14.8
Poland 14.3 10
Romania 13.6 26.9
Yugoslavia 14 20.2
USSR 17.7 23
Byelorussian_SSR 15.2 13.1
Ukrainian_SSR 13.4 13

The INPUT statement

INPUT < specification(s)> <@ | @@> ;

Specifications that can go in an input statement

- variable
  - names the variable whose value is to be read.
- (variable-list)
  - specifies a list of variables that are assigned input values.
- $  
  - indicates to store the variable values as character values rather than numeric values.
- pointer control
  - moves the input pointer to a specified line or column in the input buffer.
column-specifications
   - specifies columns in data lines where variable values are to be read.

format-modifier
   - allows modified list input or controls the amount of information reported in the SAS log when an error in an input value occurs. Modified list input can be used to read data that cannot be read with simple list input.

informat
   - specifies an informat to use to read the variable values.

(informat-list)
   - specifies a list of informats to use to read the values for the preceding list of variables

@
   - holds an input record for the execution of the next INPUT statement within the same iteration of the DATA step. This line-hold specifier is called trailing @.

@@
   - the double trailing @ holds the input record for the execution of the next INPUT statement, even across iterations of the DATA step.