Escape Trap – External Data Interaction with SAS

ABSTRACT

Sometimes we need to read external data from sources like files, Oracle, MySQL, Redshift, DB2 database to SAS dataset in our daily work. Have you ever run into any hurdles that data were not created as you expected? For instance, the data were truncated, or the data format was not correct, or even worse - the data were not recognized.
Do you know what do you need to pay attention to between external data and SAS? Or do you want to adjust the options to best fit your requirement? This paper will illustrate some useful tips and hints of interactions between external data suppliers and the SAS system.

INTRODUCTION

Data come in many different forms from different engines. During our routine work there might be a mass of data which are non-SAS file types. How do we manipulate all kinds of data in the SAS system? In addition to the original way of data step statement such as IMPORT/INFILE, we can also use SAS/Access Interface, a bridge between SAS system and DBMS system. It is an out-of-the-box solution for integrating SAS and third-party databases. SAS/ACCESS enables us read, write and update data regardless of the database or platform.
SAS supports many types of DBMS, no matter RDBMS or non-RDBMS, you can always find an appropriate solution to suit your data.
The following screenshot lists some DBMS types that SAS/Access Interface supports:

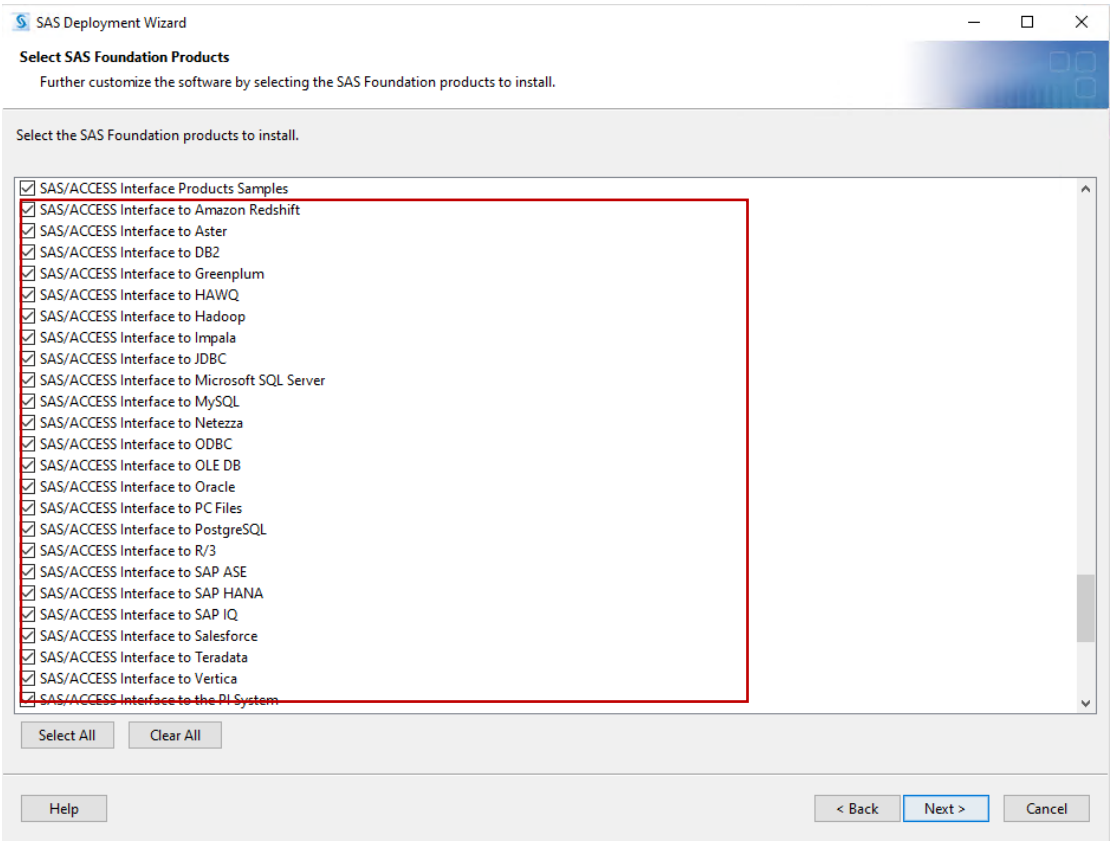


Figure 1.1

At first, let's summarize the approaches that how we manipulate external data.

The interaction strategies between external data and SAS system can be totally transparent within SAS or directly controlled "outside".

You might already know there are several approaches to access/import external data into SAS, such as INFILE statement / Proc import step / Libname engine / Proc SQL. It's decided by DBMS type that in which way you should manipulate the data.

Below are some access approaches commonly used:

✓ **Create data by INFILE statement:**

```
data user_number;
  infile 'C:\Ping\test.txt';
  input userid $ number;
proc print;
run;
```

✓ **Create data by Proc IMPORT:**

```
PROC IMPORT OUT=WORK.TEST
    FILE="C:\Ping\test_space.txt"
    DBMS=DLM REPLACE;
    DELIMITER='20'x;
    GETNAMES=NO;
    SCANTEXT=YES;
    GUESSINGROWS=10;
```

```
RUN;
```

✓ **Create data by Libname and Data step:**

```
libname libref <dbms> <connection options>;
data work;
    set libref.dbms_tab;
run;
```

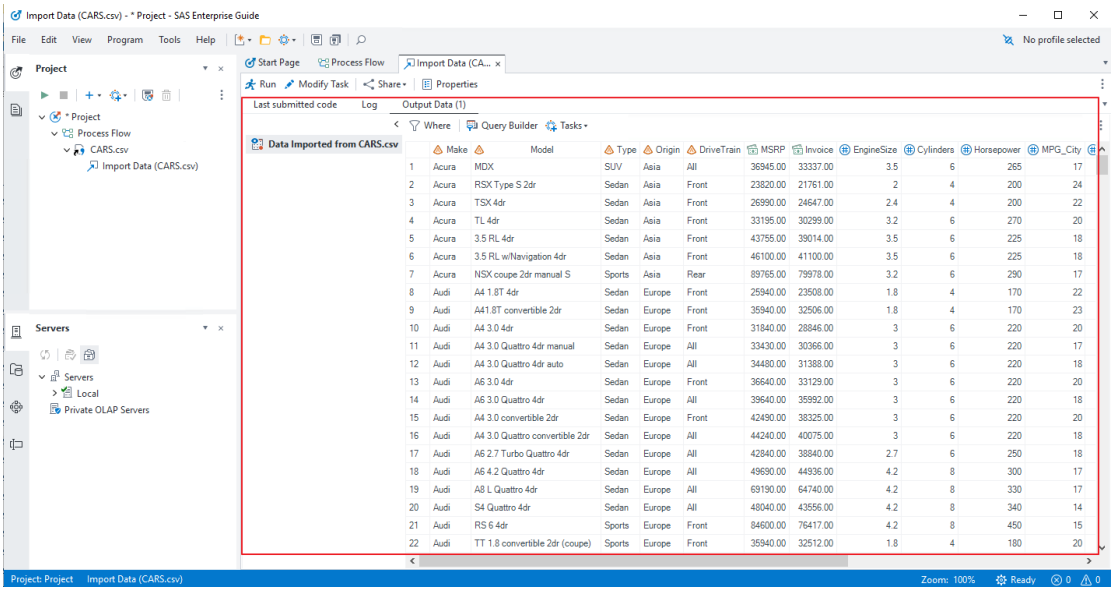
✓ **Manipulate dataset directly by proc sql:**

```
proc sql;
   connect to <dbms> ( <connection options> );
   select * from connection to <dbms>
   ( <DBMS SQL Select statement> );
   exec ( <DBMS DDL statement> ) by <dbms>;
quit;
```

For most non-RDBMS, we can use IMPORT or INFILE to get data from external file and manipulate it at SAS system. But there is an exception for **xlsx file** type. We can directly access it without importing it into SAS first, which is similar to RDBMS.

```
libname cars XLSX "C:\test\cars_xlsx.xlsx";
proc sql;
   select make, model, type from cars.sheet
   where make="Audi";
quit;
```

Besides, you can also import external via interactive UI. **SAS Enterprise Guide**, **SAS Display Manager** provide the import wizard which is more convenient and easier for end users who prefer graphical user interface.



	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City
1	Acura	MDX	SUV	Asia	All	36945.00	33337.00	3.5	6	265	17
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	23820.00	21761.00	2	4	200	24
3	Acura	TSX 4dr	Sedan	Asia	Front	26990.00	24647.00	2.4	4	200	22
4	Acura	TL 4dr	Sedan	Asia	Front	33195.00	30299.00	3.2	6	270	20
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	43755.00	39014.00	3.5	6	225	18
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	46100.00	41100.00	3.5	6	225	18
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	89765.00	79978.00	3.2	6	290	17
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front	25940.00	23508.00	1.8	4	170	22
9	Audi	A4 1.8T convertible 2dr	Sedan	Europe	Front	35940.00	32506.00	1.8	4	170	23
10	Audi	A4 3.0 4dr	Sedan	Europe	Front	31840.00	28846.00	3	6	220	20
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	33430.00	30366.00	3	6	220	17
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	34480.00	31388.00	3	6	220	18
13	Audi	A6 3.0 4dr	Sedan	Europe	Front	36640.00	33129.00	3	6	220	20
14	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	39640.00	35992.00	3	6	220	18
15	Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	42490.00	38325.00	3	6	220	20
16	Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	All	44240.00	40075.00	3	6	220	18
17	Audi	A6 2.7 Turbo Quattro 4dr	Sedan	Europe	All	42840.00	38840.00	2.7	6	250	18
18	Audi	A6 4.2 Quattro 4dr	Sedan	Europe	All	49690.00	44936.00	4.2	8	300	17
19	Audi	A8 L Quattro 4dr	Sedan	Europe	All	69190.00	64740.00	4.2	8	330	17
20	Audi	S4 Quattro 4dr	Sedan	Europe	All	48040.00	43556.00	4.2	8	340	14
21	Audi	RS 6 4dr	Sports	Europe	Front	84600.00	76417.00	4.2	8	450	15
22	Audi	TT 1.8 convertible 2dr (coupe)	Sports	Europe	Front	35940.00	32512.00	1.8	4	180	20

Figure 1.2

Or we can import/export data via SAS Studio as below:

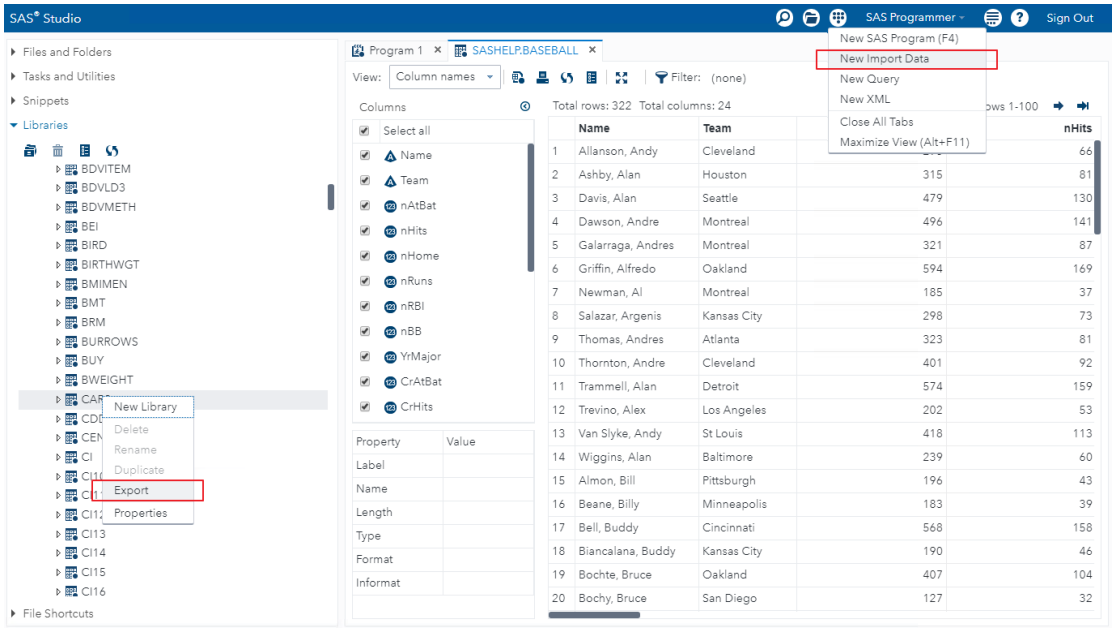


Figure 1.3

Whether accessing directly in DBMS or manipulating in SAS have their specific application scenarios. That is beyond the scope of this paper and worth another paper. This paper will focus on the gaps and approaches to solve the issues when we deal with such data between SAS & DBMS. I'll group them into several categories from general purpose.

Manage the incoming data content - proc import

Some dataset might use the first row as table column name. While other of them don't need to keep the first row as table column name. How do we deal with the data in different scenarios? We'll introduce 2 useful settings in this section:

Let's look at the example below. This is the file that includes column name, which is also the most common file format in csv/txt file.

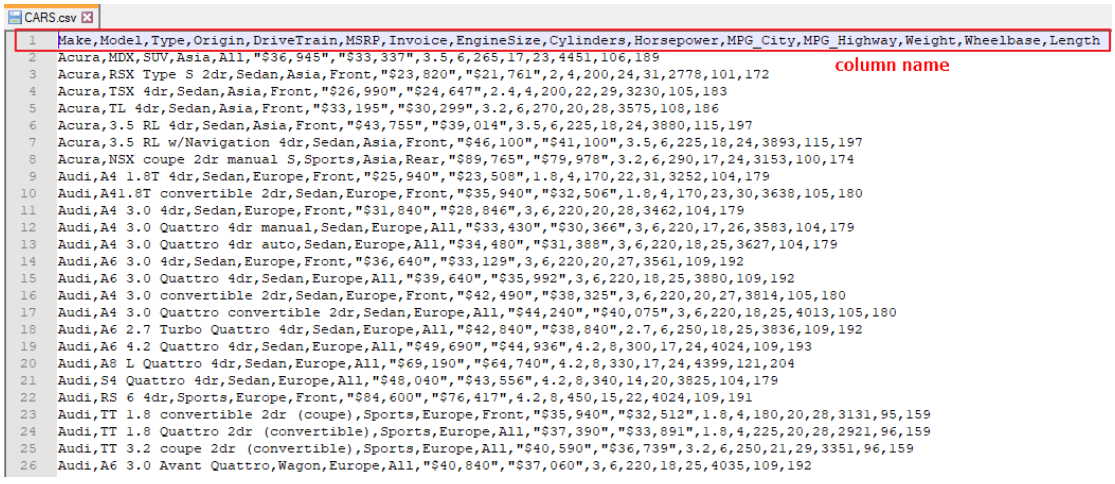
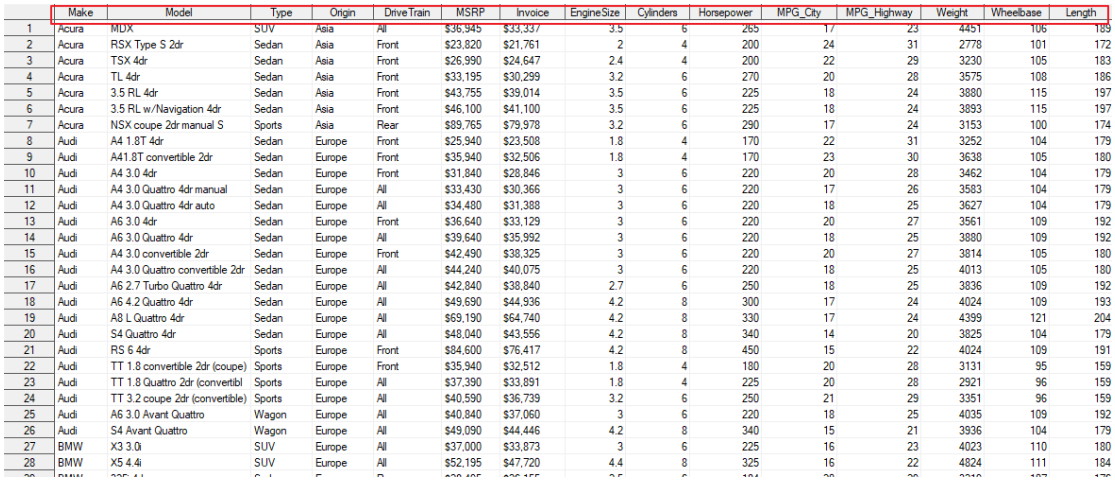


Figure 2.1

Because we want to keep the column name in first row, we need to code like this:

```
PROC IMPORT OUT=WORK.cars_csv FILE="C:\Ping\cars.csv" DBMS=csv REPLACE;
   GETNAMES=YES; /*<YES is by default> */
RUN;
```

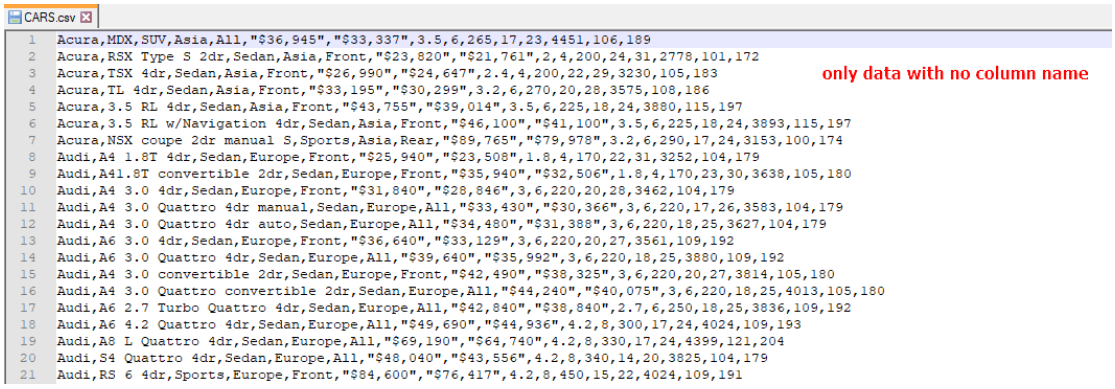
Look at the result dataset, all column names are reserved:



	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	Cylinders	Horsepower	MPG_City	MPG_Highway	Weight	Wheelbase	Length
1	Acura	MDX	SUV	Asia	All	$36,945	$33,337	3.5	6	265	17	23	4451	106	189
2	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2	4	200	24	31	2778	101	172
3	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	200	22	29	3230	105	183
4	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	270	20	28	3575	108	186
5	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	225	18	24	3880	115	197
6	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100	$41,100	3.5	6	225	18	24	3893	115	197
7	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6	290	17	24	3153	100	174
8	Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940	$23,508	1.8	4	170	22	31	3252	104	179
9	Audi	A41.8T convertible 2dr	Sedan	Europe	Front	$35,940	$32,506	1.8	4	170	23	30	3638	105	180
10	Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840	$28,846	3	6	220	20	28	3462	104	179
11	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	$33,430	$30,366	3	6	220	17	26	3583	104	179
12	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	$34,480	$31,388	3	6	220	18	25	3627	104	179
13	Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640	$33,129	3	6	220	20	27	3561	109	192
14	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	$39,640	$35,992	3	6	220	18	25	3880	109	192
15	Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	$42,490	$38,325	3	6	220	20	27	3814	105	180
16	Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	All	$44,240	$40,075	3	6	220	18	25	4013	105	180
17	Audi	A6 2.7 Turbo Quattro 4dr	Sedan	Europe	All	$42,840	$38,840	2.7	6	250	18	25	3836	109	192
18	Audi	A6 4.2 Quattro 4dr	Sedan	Europe	All	$49,690	$44,936	4.2	8	300	17	24	4024	109	193
19	Audi	A8 L Quattro 4dr	Sedan	Europe	All	$69,190	$64,740	4.2	8	330	17	24	4399	121	204
20	Audi	S4 Quattro 4dr	Sedan	Europe	All	$48,040	$43,556	4.2	8	340	14	20	3825	104	179
21	Audi	RS 6 4dr	Sports	Europe	Front	$84,600	$76,417	4.2	8	450	15	22	4024	109	191
22	Audi	TT 1.8 convertible 2dr (coupe)	Sports	Europe	Front	$35,940	$32,512	1.8	4	180	20	28	3131	95	159
23	Audi	TT 1.8 Quattro 2dr (convertibl	Sports	Europe	All	$37,390	$33,891	1.8	4	225	20	28	2921	96	159
24	Audi	TT 3.2 coupe 2dr (convertible)	Sports	Europe	All	$40,590	$36,739	3.2	6	250	21	29	3351	96	159
25	Audi	A6 3.0 Avant Quattro	Wagon	Europe	All	$40,840	$37,060	3	6	220	18	25	4035	109	192
26	Audi	S4 Avant Quattro	Wagon	Europe	All	$49,090	$44,446	4.2	8	340	15	21	3936	104	179
27	BMW	X3 3.0i	SUV	Europe	All	$37,000	$33,873	3	6	225	16	23	4023	110	180
28	BMW	X5 4.4i	SUV	Europe	All	$52,195	$47,720	4.4	8	325	16	22	4824	111	184

Figure 2.2

Now consider what happens if there are no column name definition like this:



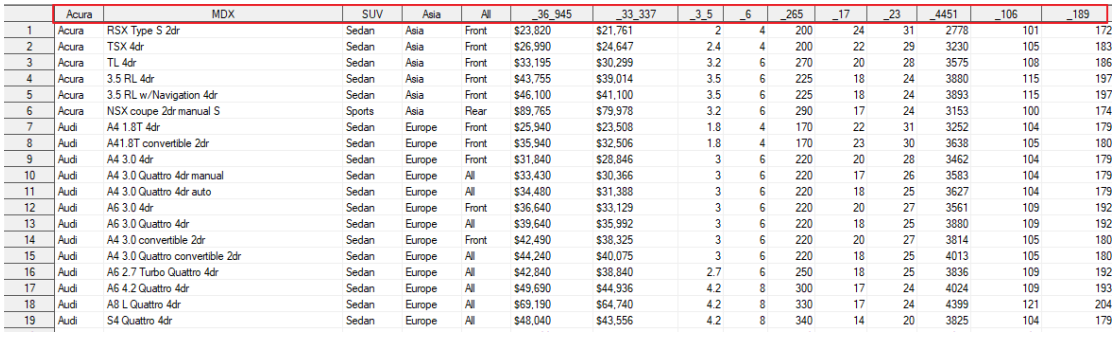
CARS.csv

```
Acura,MDX,SUV,Asia,All,"$36,945","$33,337",3.5,6,265,17,23,4451,106,189
Acura,RSX Type S 2dr,Sedan,Asia,Front,"$23,820","$21,761",2,4,200,24,31,2778,101,172
Acura,TSX 4dr,Sedan,Asia,Front,"$26,990","$24,647",2.4,4,200,22,29,3230,105,183
Acura,TL 4dr,Sedan,Asia,Front,"$33,195","$30,299",3.2,6,270,20,28,3575,108,186
Acura,3.5 RL 4dr,Sedan,Asia,Front,"$43,755","$39,014",3.5,6,225,18,24,3880,115,197
Acura,3.5 RL w/Navigation 4dr,Sedan,Asia,Front,"$46,100","$41,100",3.5,6,225,18,24,3893,115,197
Acura,NSX coupe 2dr manual S,Sports,Asia,Rear,"$89,765","$79,978",3.2,6,290,17,24,3153,100,174
Audi,A4 1.8T 4dr,Sedan,Europe,Front,"$25,940","$23,508",1.8,4,170,22,31,3252,104,179
Audi,A41.8T convertible 2dr,Sedan,Europe,Front,"$35,940","$32,506",1.8,4,170,23,30,3638,105,180
Audi,A4 3.0 4dr,Sedan,Europe,Front,"$31,840","$28,846",3,6,220,20,28,3462,104,179
Audi,A4 3.0 Quattro 4dr manual,Sedan,Europe,All,"$33,430","$30,366",3,6,220,17,26,3583,104,179
Audi,A4 3.0 Quattro 4dr auto,Sedan,Europe,All,"$34,480","$31,388",3,6,220,18,25,3627,104,179
Audi,A6 3.0 4dr,Sedan,Europe,Front,"$36,640","$33,129",3,6,220,20,27,3561,109,192
Audi,A6 3.0 Quattro 4dr,Sedan,Europe,All,"$39,640","$35,992",3,6,220,18,25,3880,109,192
Audi,A4 3.0 convertible 2dr,Sedan,Europe,Front,"$42,490","$38,325",3,6,220,20,27,3814,105,180
Audi,A4 3.0 Quattro convertible 2dr,Sedan,Europe,All,"$44,240","$40,075",3,6,220,18,25,4013,105,180
Audi,A6 2.7 Turbo Quattro 4dr,Sedan,Europe,All,"$42,840","$38,840",2.7,6,250,18,25,3836,109,192
Audi,A6 4.2 Quattro 4dr,Sedan,Europe,All,"$49,690","$44,936",4.2,8,300,17,24,4024,109,193
Audi,A8 L Quattro 4dr,Sedan,Europe,All,"$69,190","$64,740",4.2,8,330,17,24,4399,121,204
Audi,S4 Quattro 4dr,Sedan,Europe,All,"$48,040","$43,556",4.2,8,340,14,20,3825,104,179
Audi,RS 6 4dr,Sports,Europe,Front,"$84,600","$76,417",4.2,8,450,15,22,4024,109,191
```

only data with no column name

Figure 2.3

If we don't change the option of GETNAMES, the first row is treated as column name instead of a "real" data row.



	Acura	MDX	SUV	Asia	All	_36_945	_33_337	_3_5	_6	_265	_17	_23	_4451	_106	_189
1	Acura	RSX Type S 2dr	Sedan	Asia	Front	$23,820	$21,761	2	4	200	24	31	2778	101	172
2	Acura	TSX 4dr	Sedan	Asia	Front	$26,990	$24,647	2.4	4	200	22	29	3230	105	183
3	Acura	TL 4dr	Sedan	Asia	Front	$33,195	$30,299	3.2	6	270	20	28	3575	108	186
4	Acura	3.5 RL 4dr	Sedan	Asia	Front	$43,755	$39,014	3.5	6	225	18	24	3880	115	197
5	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	$46,100	$41,100	3.5	6	225	18	24	3893	115	197
6	Acura	NSX coupe 2dr manual S	Sports	Asia	Rear	$89,765	$79,978	3.2	6	290	17	24	3153	100	174
7	Audi	A4 1.8T 4dr	Sedan	Europe	Front	$25,940	$23,508	1.8	4	170	22	31	3252	104	179
8	Audi	A41.8T convertible 2dr	Sedan	Europe	Front	$35,940	$32,506	1.8	4	170	23	30	3638	105	180
9	Audi	A4 3.0 4dr	Sedan	Europe	Front	$31,840	$28,846	3	6	220	20	28	3462	104	179
10	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	$33,430	$30,366	3	6	220	17	26	3583	104	179
11	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	$34,480	$31,388	3	6	220	18	25	3627	104	179
12	Audi	A6 3.0 4dr	Sedan	Europe	Front	$36,640	$33,129	3	6	220	20	27	3561	109	192
13	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	$39,640	$35,992	3	6	220	18	25	3880	109	192
14	Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	$42,490	$38,325	3	6	220	20	27	3814	105	180
15	Audi	A4 3.0 Quattro convertible 2dr	Sedan	Europe	All	$44,240	$40,075	3	6	220	18	25	4013	105	180
16	Audi	A6 2.7 Turbo Quattro 4dr	Sedan	Europe	All	$42,840	$38,840	2.7	6	250	18	25	3836	109	192
17	Audi	A6 4.2 Quattro 4dr	Sedan	Europe	All	$49,690	$44,936	4.2	8	300	17	24	4024	109	193
18	Audi	A8 L Quattro 4dr	Sedan	Europe	All	$69,190	$64,740	4.2	8	330	17	24	4399	121	204
19	Audi	S4 Quattro 4dr	Sedan	Europe	All	$48,040	$43,556	4.2	8	340	14	20	3825	104	179

Figure 2.4

To resolve this issue, we can change the code:

```
PROC IMPORT OUT=WORK.cars_csv FILE="C:\Ping\CARS.csv" DBMS=csv REPLACE;
   GETNAMES=NO;
RUN;
```

Furthermore, we can adjust the **DATAROW** value to fit our requirement.

```
PROC IMPORT OUT=XXX FILE="C:\Ping\CARS.csv" DBMS=csv REPLACE;
   GETNAMES=YES|NO;
   DATAROW=10;
RUN;
```

Now let's check the option definition of **DATAROW** and **GETNAMES.**

GETNAMES=YES | NO (specifies whether the IMPORT procedure is to generate SAS variable names from the data values in the first row of the import file.**)**

In short, if GETNAMES=YES is set, the first row of data in the range is used for the column names, and the data starts from the second row in the range. If GETNAMES=NO is set, the data starts from the first row and column names are generated by the IMPORT procedure.

DATAROW=*n* (specifies the row number where the IMPORT procedure starts reading data. When GETNAMES=NO: DATAROW=1; when GETNAMES=YES: DATAROW=2**)**

DBMS identifier	Statement Options	Support PROC IMPORT	Support PROC EXPORT	Valid Value	Default Value
CSV and Tab	DATAROW	Yes	No	1 to 2147483647	Depends on GETNAMES= option value
	GETNAMES	Yes	No	Yes | No	Yes
DLM	DATAROW	Yes	No	1 to 2147483647	2
	GETNAMES	Yes	No	Yes | No	Yes

Table 2.1

Manage type and length when import

Have you ever met column type mismatch or data truncation issue? It's quite common when column definition does not accord with the real data. Do you know how to avoid such issues when importing data? Let's consider a scenario: we need to import a CSV file which has N rows. We want all columns' type and length could be recognized automatically. Let's check the columns in the example below:

Sale.txt — tab delimited data

```
COUNTRY STATE    COUNTY  ACTUAL  PREDICT PRODTYPE    PRODUCT YEAR    QUARTER MONTH   MONYR
U.S.A.  California       $987.36 $692.24 FURNITURE   SOFA    1995    1   Jan JAN95
U.S.A.  California       $1,782.96   $568.48 FURNITURE   SOFA    1995    1   Feb FEB95
U.S.A.  California       $32.64  $16.32  FURNITURE   SOFA    1995    1   Mar MAR95
U.S.A.  California       $1,825.12   $756.16 FURNITURE   SOFA    1995    2   Apr APR95
U.S.A.  California       $750.72 $723.52 FURNITURE   SOFA    1995    2   May MAY95
U.S.A.  California       $2,426.24   $2,428.96   FURNITURE   SOFA    1995    2   Jun JUN95
U.S.A.  California       $1,791.12   $2,250.80   FURNITURE   SOFA    1995    3   Jul JUL95
U.S.A.  California       $2,282.08   $350.88 FURNITURE   SOFA    1995    3   Aug AUG95
U.S.A.  California       $2,518.72   $1,736.72   FURNITURE   SOFA    1995    3   Sep SEP95
U.S.A.  California       $1,436.16   $2,167.84   FURNITURE   SOFA    1995    4   Oct OCT95
U.S.A.  California       $2,314.72   $62.56  FURNITURE   SOFA    1995    4   Nov NOV95
U.S.A.  California       $1,410.32   $1,670.08   FURNITURE   SOFA    1995    4   Dec DEC95
U.S.A.  California       $369.92 $1,365.44   FURNITURE   BED 1995    1   Jan JAN95
U.S.A.  California       $2,014.16   $2,358.24   FURNITURE   BED 1995    1   Feb FEB95
U.S.A.  California       $85.68  $2,594.88   FURNITURE   BED 1995    1   Mar MAR95
U.S.A.  California       $2,694.16   $1,403.52   FURNITURE   BED 1995    2   Apr APR95
U.S.A.  California       $2,014.16   $707.20 FURNITURE   BED 1995    2   May MAY95
U.S.A.  California       $1,500.08   $2,461.60   FURNITURE   BED 1995    2   Jun JUN95
U.S.A.  California       $2,133.84   $2,486.08   FURNITURE   BED 1995    3   Jul JUL95
U.S.A.  California       $2,133.84   $2,486.08   FURNITURE   CHEST OF DRAWERS    1995    3   Jul JUL95
U.S.A.  California       $1,834.64   $2,884.56   FURNITURE   BED 1995    3   Aug AUG95
U.S.A.  California       $2,687.36   $1,224.00   FURNITURE   BED 1995    3   Sep SEP95
U.S.A.  California       $646.00 $950.64 FURNITURE   BED 1995    4   Oct OCT95
U.S.A.  California       $2,392.24   $2,356.88   FURNITURE   BED 1995    4   Nov NOV95
U.S.A.  California       $913.92 $1,538.16   FURNITURE   BED 1995    4   Dec DEC95
U.S.A.  California       $1,090.72   $375.36 OFFICE  CHAIR   1995    1   Jan JAN95
U.S.A.  California       $2,151.52   $1,025.44   OFFICE  CHAIR   1995    1   Feb FEB95
U.S.A.  California       $1,221.28   $265.20 OFFICE  CHAIR   1995    1   Mar MAR95
U.S.A.  California       $1,370.88   $2,283.44   OFFICE  CHAIR   1995    2   Apr APR95
```

Figure 3.1

Now we try to import:

```
PROC IMPORT OUT=WORK.sale FILE="C:\Ping\sale.txt" DBMS=tab REPLACE;
RUN;
```

Note the value in PRODUCT column is truncated:

VIEWTABLE: Work.Sale

	COUNTRY	STATE	COUNTY	ACTUAL	PREDICT	PRODTYPE	PRODUCT	YEAR	QUARTER	MONTH	MONYR
1	U.S.A.	California		987	692	FURNITURE	SOFA	1995	1	Jan	JAN1995
2	U.S.A.	California		1,783	568	FURNITURE	SOFA	1995	1	Feb	FEB1995
3	U.S.A.	California		33	16	FURNITURE	SOFA	1995	1	Mar	MAR1995
4	U.S.A.	California		1,825	756	FURNITURE	SOFA	1995	2	Apr	APR1995
5	U.S.A.	California		751	724	FURNITURE	SOFA	1995	2	May	MAY1995
6	U.S.A.	California		2,426	2,429	FURNITURE	SOFA	1995	2	Jun	JUN1995
7	U.S.A.	California		1,791	2,251	FURNITURE	SOFA	1995	3	Jul	JUL1995
8	U.S.A.	California		2,282	351	FURNITURE	SOFA	1995	3	Aug	AUG1995
9	U.S.A.	California		2,519	1,737	FURNITURE	SOFA	1995	3	Sep	SEP1995
10	U.S.A.	California		1,436	2,168	FURNITURE	SOFA	1995	4	Oct	OCT1995
11	U.S.A.	California		2,315	63	FURNITURE	SOFA	1995	4	Nov	NOV1995
12	U.S.A.	California		1,410	1,670	FURNITURE	SOFA	1995	4	Dec	DEC1995
13	U.S.A.	California		370	1,365	FURNITURE	BED	1995	1	Jan	JAN1995
14	U.S.A.	California		2,014	2,358	FURNITURE	BED	1995	1	Feb	FEB1995
15	U.S.A.	California		86	2,595	FURNITURE	BED	1995	1	Mar	MAR1995
16	U.S.A.	California		2,694	1,404	FURNITURE	BED	1995	2	Apr	APR1995
17	U.S.A.	California		2,014	707	FURNITURE	BED	1995	2	May	MAY1995
18	U.S.A.	California		1,500	2,462	FURNITURE	BED	1995	2	Jun	JUN1995
19	U.S.A.	California		2,134	2,486	FURNITURE	BED	1995	3	Jul	JUL1995
20	U.S.A.	California		2,134	2,486	FURNITURE	CHES	1995	3	Jul	JUL1995
21	U.S.A.	California		1,835	2,885	FURNITURE	BED	1995	3	Aug	AUG1995
22	U.S.A.	California		2,687	1,224	FURNITURE	BED	1995	3	Sep	SEP1995
23	U.S.A.	California		646	951	FURNITURE	BED	1995	4	Oct	OCT1995
24	U.S.A.	California		2,392	2,357	FURNITURE	BED	1995	4	Nov	NOV1995
25	U.S.A.	California		914	1,538	FURNITURE	BED	1995	4	Dec	DEC1995
26	U.S.A.	California		1,091	375	OFFICE	CHAI	1995	1	Jan	JAN1995
27	U.S.A.	California		2,152	1,025	OFFICE	CHAI	1995	1	Feb	FEB1995
28	U.S.A.	California		1,221	265	OFFICE	CHAI	1995	1	Mar	MAR1995
29	U.S.A.	California		1,371	2,283	OFFICE	CHAI	1995	2	Apr	APR1995
30	U.S.A.	California		2,547	1,926	OFFICE	CHAI	1995	2	May	MAY1995

truncated

Figure 3.2

The data definition is:

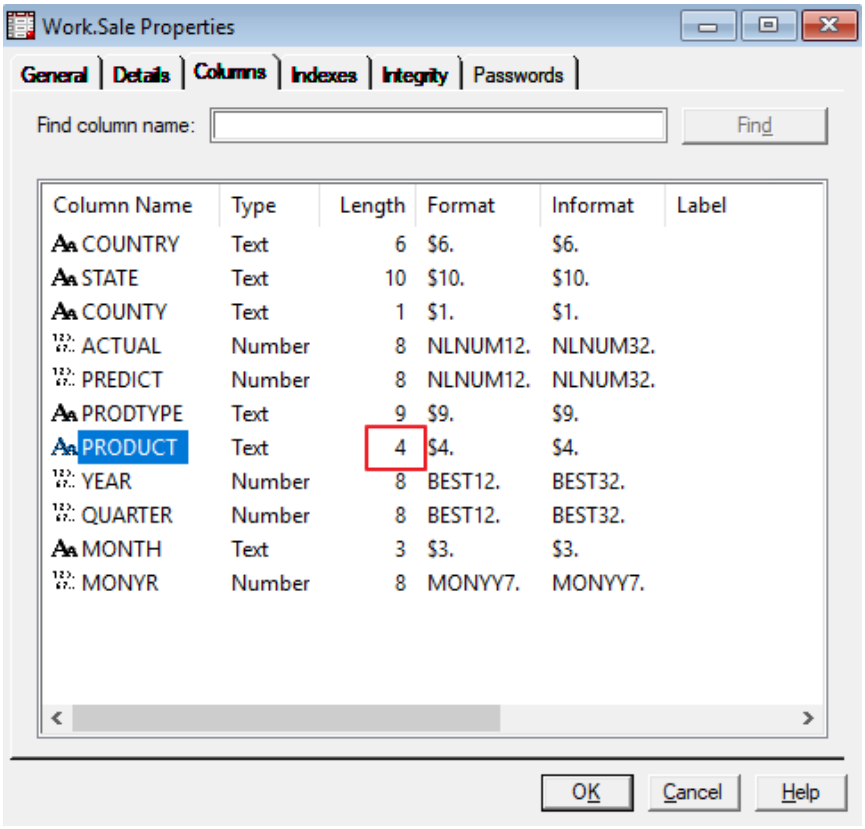


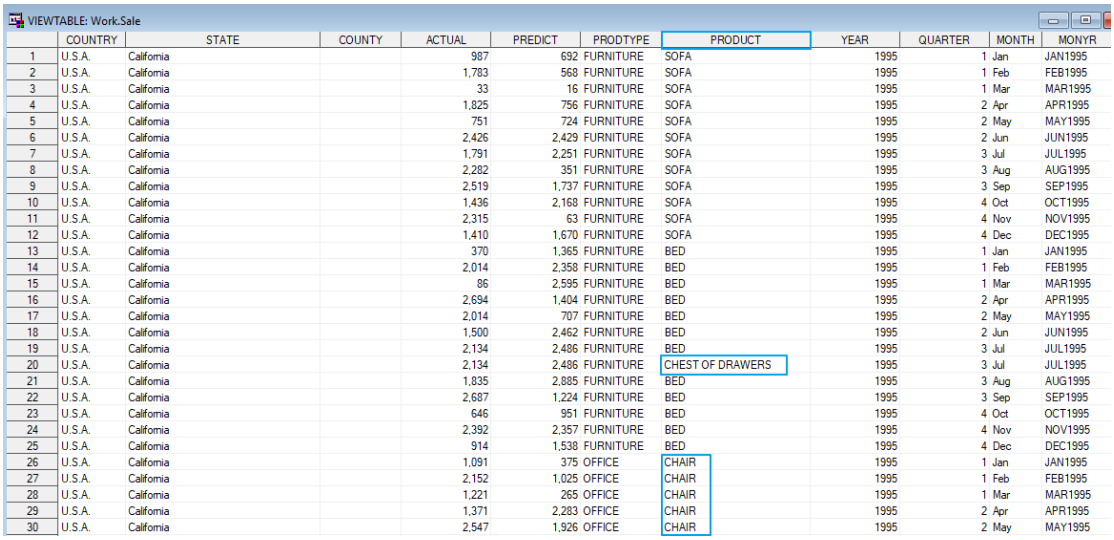
Figure 3.3

We can see the column type is text, while the length definition of PRODUCT is 4, not the maximum length of our current data. It behaves like this because PROC IMPORT scans the first **20** rows (including the first column name row) by default to determine the correct type, length and format for each column.

To avoid mismatch and truncation issue, we can change the scan count to a bigger number by introducing a useful option. Here is the updated code:

```
PROC IMPORT OUT=WORK.sale FILE="C:\Ping\sale.txt" DBMS=tab REPLACE;
GUESSINGROWS=n|<the maximum row count>;
RUN;
```

Result data:



VIEWTABLE: Work.Sale

	COUNTRY	STATE	COUNTY	ACTUAL	PREDICT	PRODTYPE	PRODUCT	YEAR	QUARTER	MONTH	MONYR
1	U.S.A.	California		987	692	FURNITURE	SOFA	1995	1	Jan	JAN1995
2	U.S.A.	California		1,783	568	FURNITURE	SOFA	1995	1	Feb	FEB1995
3	U.S.A.	California		33	16	FURNITURE	SOFA	1995	1	Mar	MAR1995
4	U.S.A.	California		1,825	756	FURNITURE	SOFA	1995	2	Apr	APR1995
5	U.S.A.	California		751	724	FURNITURE	SOFA	1995	2	May	MAY1995
6	U.S.A.	California		2,426	2,429	FURNITURE	SOFA	1995	2	Jun	JUN1995
7	U.S.A.	California		1,791	2,251	FURNITURE	SOFA	1995	3	Jul	JUL1995
8	U.S.A.	California		2,282	351	FURNITURE	SOFA	1995	3	Aug	AUG1995
9	U.S.A.	California		2,519	1,737	FURNITURE	SOFA	1995	3	Sep	SEP1995
10	U.S.A.	California		1,436	2,168	FURNITURE	SOFA	1995	4	Oct	OCT1995
11	U.S.A.	California		2,315	63	FURNITURE	SOFA	1995	4	Nov	NOV1995
12	U.S.A.	California		1,410	1,670	FURNITURE	SOFA	1995	4	Dec	DEC1995
13	U.S.A.	California		370	1,365	FURNITURE	BED	1995	1	Jan	JAN1995
14	U.S.A.	California		2,014	2,358	FURNITURE	BED	1995	1	Feb	FEB1995
15	U.S.A.	California		86	2,595	FURNITURE	BED	1995	1	Mar	MAR1995
16	U.S.A.	California		2,694	1,404	FURNITURE	BED	1995	2	Apr	APR1995
17	U.S.A.	California		2,014	707	FURNITURE	BED	1995	2	May	MAY1995
18	U.S.A.	California		1,500	2,462	FURNITURE	BED	1995	2	Jun	JUN1995
19	U.S.A.	California		2,134	2,486	FURNITURE	BED	1995	3	Jul	JUL1995
20	U.S.A.	California		2,134	2,486	FURNITURE	CHEST OF DRAWERS	1995	3	Jul	JUL1995
21	U.S.A.	California		1,835	2,885	FURNITURE	BED	1995	3	Aug	AUG1995
22	U.S.A.	California		2,687	1,224	FURNITURE	BED	1995	3	Sep	SEP1995
23	U.S.A.	California		646	951	FURNITURE	BED	1995	4	Oct	OCT1995
24	U.S.A.	California		2,392	2,357	FURNITURE	BED	1995	4	Nov	NOV1995
25	U.S.A.	California		914	1,538	FURNITURE	BED	1995	4	Dec	DEC1995
26	U.S.A.	California		1,091	375	OFFICE	CHAIR	1995	1	Jan	JAN1995
27	U.S.A.	California		2,152	1,025	OFFICE	CHAIR	1995	1	Feb	FEB1995
28	U.S.A.	California		1,221	265	OFFICE	CHAIR	1995	1	Mar	MAR1995
29	U.S.A.	California		1,371	2,283	OFFICE	CHAIR	1995	2	Apr	APR1995
30	U.S.A.	California		2,547	1,926	OFFICE	CHAIR	1995	2	May	MAY1995

Figure 3.4

Now we should have a basic understanding about the option usage, but keep in mind the possible value is up to **2147483647** (the maximum value). Below is the usage from reference: **GUESSINGROWS=*n* | MAX** specifies the number of rows that the IMPORT procedure is to scan to determine the appropriate data type for the columns. The scan process scans from row 1 to the row number that is specified by GUESSINGROWS= option. MAX can be specified instead of the maximum number, 2147483647. Keep this in mind – You could define this number as big as possible, but specifying the maximal value could adversely affect performance.

Special characters trouble

Imagine a scenario, we have some tables and columns which including special characters like "@,#,$,*" that are not permitted in SAS naming convention. We want these characters recognized well in SAS system instead of being garbled or leading to error. In this circumstance, we must use one of these techniques: **validvarname=any** or **dquote=ansi**.

validvarname=any:

Specify the global system option VALIDVARNAME=ANY and use name literals in the SAS language. Here is an example and its running log.

Consider we have an excel file which including special characters as below:

containing special chars

	A	B	C	D	E	F
1	Make#$	Model	Type	Origin	DriveTrain	MSRP
2	Acura	MDX	SUV	Asia	All	36945
3	Acura	RSX Type S 2dr	Sedan	Asia	Front	23820
4	Acura	TSX 4dr	Sedan	Asia	Front	26990
5	Acura	TL 4dr	Sedan	Asia	Front	33195
6	Acura	3.5 RL 4dr	Sedan	Asia	Front	43755
7	Acura	3.5 RL w/Navigation 4dr	Sedan	Asia	Front	46100
8	Acura	NSX coupe 2dr manual S	Sport	Asia	Rear	89765
9	Audi	A4 1.8T 4dr	Sedan	Europe	Front	25940
10	Audi	A41.8T convertible 2dr	Sedan	Europe	Front	35940
11	Audi	A4 3.0 4dr	Sedan	Europe	Front	31840
12	Audi	A4 3.0 Quattro 4dr manual	Sedan	Europe	All	33430
13	Audi	A4 3.0 Quattro 4dr auto	Sedan	Europe	All	34480
14	Audi	A6 3.0 4dr	Sedan	Europe	Front	36640
15	Audi	A6 3.0 Quattro 4dr	Sedan	Europe	All	39640
16	Audi	A4 3.0 convertible 2dr	Sedan	Europe	Front	42490

Figure 4.1

At first, we import this data as usual:

```
libname cars XLSX "C:\ping\cars.xlsx";
proc sql;
   select "Make#$"n, model, type from cars.sheet;
quit;
```

And we'll get such message since SAS can't process special chars in column name by default:

```
1     libname cars XLSX "C:\ping\cars.xlsx";
NOTE: Libref CARS was successfully assigned as follows:
      Engine:        XLSX
      Physical Name: C:\ping\cars.xlsx
2     proc sql;
3     select "Make#$"n, model,type from cars.sheet;
NOTE:     Variable Name Change.  Make#$ -> Make__
ERROR: The following columns were not found in the contributing tables: Make#$.
4     quit;
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.09 seconds
      cpu time            0.06 seconds
```

After adding options:

```
options validvarname=any;
libname cars XLSX "C:\ping\cars.xlsx";
proc sql;
   select "Make#$"n, model, type from cars.sheet;
quit;
```

Result log & data:

```
10   options validvarname=any;
11   libname cars XLSX "C:\ping\cars.xlsx";
NOTE: Libref CARS was successfully assigned as follows:
      Engine:        XLSX
      Physical Name: C:\ping\cars.xlsx
12   proc sql;
13   select "Make#$"n, model,type from cars.sheet;
NOTE: Writing HTML Body file: sashtml.htm
NOTE: The import data set has 428 observations and 15 variables.
14   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.72 seconds
      cpu time            0.53 seconds
```

The SAS System

Make#$	Model	Type
Acura	MDX	SUV
Acura	RSX Type S 2dr	Sedan
Acura	TSX 4dr	Sedan
Acura	TL 4dr	Sedan
Acura	3.5 RL 4dr	Sedan
Acura	3.5 RL w/Navigation 4dr	Sedan
Acura	NSX coupe 2dr manual S	Sports
Audi	A4 1.8T 4dr	Sedan
Audi	A41.8T convertible 2dr	Sedan
Audi	A4 3.0 4dr	Sedan
Audi	A4 3.0 Quattro 4dr manual	Sedan
Audi	A4 3.0 Quattro 4dr auto	Sedan

Figure 4.2

dquote=ansi:

dquote is a PROC SQL option. This option specifies whether PROC SQL treats values within **double** quotation marks as a character string or as a column name or table name. When you specify **dquote=ansi**, your SAS code can refer to DBMS names that contain characters and spaces that are not allowed by SAS naming conventions. This option allows you to use reserved words, and other names that do not meet SAS specifications. The value ANSI allows SAS to treat values enclosed in double quotation marks as a variable name. The default value is 'SAS' And values in double quotation marks are treated as character strings. dquote option overrides the validvarname option if present.

Let's take an example:

In general, value in quotation marks are recognized as characters, not a variable. Therefore, column **COLUMN1#$** is identified as a character string returned in result list. Check the source

and result:

```
/* dquote=SAS by default */
proc sql;
   select "COLUMN1#$" from ora.length_name;
quit;
```

Result data & log:

```
153  proc sql;
154  select "COLUMN1#$" from ora.length_name;
NOTE: Writing HTML Body file: sashtml2.htm
155  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time               0.93 seconds
      cpu time                0.65 seconds
```

The SAS System

COLUMN1#$
COLUMN1#$

What if we have a column with special chars? How can it be identified correctly? It's easy – just adding the option **dquote** and set the value to ansi. Check the source and result below:

```
/* dquote=ansi */
proc sql dquote=ansi;
   select "COLUMN1#$" from ora.length_name;
quit;
```

Result log & data:

```
159  proc sql dquote=ansi;
160  select "COLUMN1#$" from ora.length_name;
161  quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time               0.06 seconds
      cpu time                0.06 seconds
```

The SAS System

COLUMN1#$
1
2

Gap in length definition between Oracle and SAS

When we manipulate third-party data in SAS, we definitely should pay attention to its column length. Generally, we define it much shorter than the length definition to avoid data truncation. This is safe though. What if the data actual length is not what we expected, and it exceeds the definition? For instance, we can insert such data which exceeds the column length even if the data are truncated in English SAS session. While we can't achieve the same goal because this will lead to an error by default in non-English SAS session.

Let me give you an example to illustrate. There is an Oracle table definition as follows. Take a look at the column definition of JOB1 and keep it in mind.

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	LEVEL2	VARCHAR2(52 BYTE)	Yes	(null)	1	(null)
2	LEVEL1	VARCHAR2(64 BYTE)	Yes	(null)	2	(null)
3	LEVEL5	VARCHAR2(120 BYTE)	Yes	(null)	3	(null)
4	DEPTHEAD	VARCHAR2(60 BYTE)	Yes	(null)	4	(null)
5	LEVEL3	VARCHAR2(80 BYTE)	Yes	(null)	5	(null)
6	LEVEL4	VARCHAR2(120 BYTE)	Yes	(null)	6	(null)
7	JOB1	VARCHAR2(60 BYTE)	Yes	(null)	7	(null)
8	N	NUMBER	Yes	(null)	8	(null)

Figure 5.1

When we want to add new rows into an existing SAS table in **English** SAS session, typically code like this: (actual length of **JOB1** column is 101 bytes)

```
libname ora oracle user=<user> pw=<password>;
proc sql;
  insert into ora.company values
  ('BEIJING','International','Ping Lu', '1', 'Marketing', 'Sale', 'Marketing Specialist; Marketing Manager; Marketing Director; Graphic Designer; Marketing Research Analyst', 1);
quit;
```

It runs successfully even if there is a truncation warning like below.

```
11   libname ora oracle user=system pw=XXXXXXXX;
NOTE: Libref ORA was successfully assigned as follows:
      Engine:        ORACLE
      Physical Name:
12   proc sql;
13   insert into ora.company values
14   ('BEIJING','International','Ping Lu', '1', 'Marketing', 'Sale', 'Marketing Specialist
14 ! ;Marketing Manager;Marketing Director;Graphic Designer;Marketing Research Analyst',
14 ! 1);
WARNING: Character expression will be truncated when assigned to character column JOB1.
NOTE: 1 row was inserted into ORA.COMPANY.

15   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.04 seconds
      cpu time            0.03 seconds
```

Column JOB1 length is now truncated to 60 bytes which just fits the definition:

VIEWTABLE: Ora.Company

	LEVEL2	LEVEL1	LEVEL5	DEPTHEAD	LEVEL3	LEVEL4	JOB1	N
38	NEW YORK	International Ai	Elaine Dumas	2	SALES/MARKETING	SALES	INDUSTRY	1
39	NEW YORK	International Ai	Alan Picard	2	SALES/MARKETING	SALES	RESPONS. TERTIA	1
40	NEW YORK	International Ai	Jean Francois Dumas	2	SALES/MARKETING	SALES	PUBLIC	1
41	NEW YORK	International Ai	Peter Caillon	2	SALES/MARKETING	SALES	AGENCE TERTIARE	1
42	NEW YORK	International Ai	Alan Bentz	2	SALES/MARKETING	SALES	ASSISTANT	1
43	NEW YORK	International Ai	Richard G. Roach	1	TECHN. SERVICES	MIS	MANAGER	1
44	NEW YORK	International Ai	Danielle Biabaut	2	TECHN. SERVICES	MIS	TECH.-CONS.	1
45	NEW YORK	International Ai	George H. Ruth	2	TECHN. SERVICES	MIS	TECH.-CONS	1
46	NEW YORK	International Ai	Michael Garris	2	TECHN. SERVICES	MIS	ASSISTANT	1
47	NEW YORK	International Ai	Claire Voyant	2	TECHN. SERVICES	MIS	TRANSLATOR	1
48	NEW YORK	International Ai	Roy Hobbs	2	TECHN. SERVICES	MIS	TECH. CONS.	1
49	BEIJING	International	Ping Lu	1	Marketing	Sale	Marketing Specialist;Marketing Manager;Marketing Director;Gr	1

60 bytes

Figure 5.2

This is quite common for large data size. We might not be able to guarantee the actual size of whole data is always shorter than length definition since actual data probably are not what we imagined. In this circumstance, we would rather choose to accept truncation than receive rollback error for some columns data when these columns are not key measure.

Now let's run the same code in **non-English** SAS session, see what happens? This row isn't permitted due to the large value length. A rollback is triggered:

```
6    libname ora oracle user=system pw=XXXXXXXX;
NOTE: Libref ORA was successfully assigned as follows:
      Engine:        ORACLE
      Physical Name:
7
8    proc sql;
9    insert into ora.company values ('BEIJING','International','Ping Lu', '1', 'Marketing', 'Sale', 'Marketing
9  ! Specialist;Marketing Manager;Marketing Director;Graphic Designer;Marketing Research Analyst', 1);
ERROR: ERROR: ERROR: ORACLE execute error: ORA-12899: value too large for column "SYSTEM"."COMPANY"."JOB1" (actual:
       101, maximum: 60).        With the occurrence of the above ERROR, the error limit of 1 set by the ERRLIMIT=
       option has been reached. ROLLBACK has been issued(Any Rows processed after the last COMMIT are lost).

      Total rows processed: 1
      Rows failed         : 1

ERROR: ROLLBACK issued due to errors for data set ORA.COMPANY.DATA.
10   quit;
NOTE: The SAS System stopped processing this step because of errors.
```

How can we let the row be accepted like in English SAS session? Let's introduce an option - **DBSERVER_ENCODING_FIXED**. It is an option for libname statement and supported since 9.4M1. SAS add this option for helping produce the same result as English session when data length exceeds the definition length in DB.

Below is the adjusted libname statement:

```
libname ora_yes oracle user=<user> pw=<password>DBSERVER_ENCODING_FIXED=YES;
proc sql;
  insert into ora_yes.company values
  ('BEIJING','International','Ping Lu', '1', 'Marketing', 'Sale', 'Marketing Specialist;Marketing
Manager;Marketing Director;Graphic Designer;Marketing Research Analyst', 1);
quit;
```

Now let's execute this adjusted code again and check the result. It produces a warning but get the result correctly inserted.

```
17   libname ora_yes oracle user=system pw=XXXXXXXX DBSERVER_ENCODING_FIXED=YES;

NOTE: Libref ORA_YES was successfully assigned as follows:
      Engine:        ORACLE
      Physical Name:
18
19   proc sql;
20   insert into ora_yes.company values ('BEIJING','International','Ping Lu', '1', 'Marketing', 'Sale', 'Marketing
20 ! Specialist;Marketing Manager;Marketing Director;Graphic Designer;Marketing Research Analyst', 1);
WARNING: Character expression will be truncated when assigned to character column JOB1.
NOTE: 1 row was inserted into ORA_YES.COMPANY.

21   quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.06 seconds
      cpu time            0.03 seconds
```

The result is:

VIEWTABLE: Ora_yes.Company

	LEVEL2	LEVEL1	LEVEL5	DEPTHEAD	LEVEL3	LEVEL4	JOB1	N
38	NEW YORK	International Ai	Elaine Dumas	2	SALES/MARKETING	SALES	INDUSTRY	1
39	NEW YORK	International Ai	Alan Picard	2	SALES/MARKETING	SALES	RESPONS. TERTIA	1
40	NEW YORK	International Ai	Jean Francois Dumas	2	SALES/MARKETING	SALES	PUBLIC	1
41	NEW YORK	International Ai	Peter Caillon	2	SALES/MARKETING	SALES	AGENCE TERTIARE	1
42	NEW YORK	International Ai	Alan Bentz	2	SALES/MARKETING	SALES	ASSISTANT	1
43	NEW YORK	International Ai	Richard G. Roach	1	TECHN. SERVICES	MIS	MANAGER	1
44	NEW YORK	International Ai	Danielle Biabaut	2	TECHN. SERVICES	MIS	TECH.-CONS.	1
45	NEW YORK	International Ai	George H. Ruth	2	TECHN. SERVICES	MIS	TECH.-CONS	1
46	NEW YORK	International Ai	Michael Garris	2	TECHN. SERVICES	MIS	ASSISTANT	1
47	NEW YORK	International Ai	Claire Voyant	2	TECHN. SERVICES	MIS	TRANSLATOR	1
48	NEW YORK	International Ai	Roy Hobbs	2	TECHN. SERVICES	MIS	TECH. CONS.	1
49	BEIJING	International	Ping Lu	1	Marketing	Sale	Marketing Specialist;Marketing Manager;Marketing Director;Gr	1

60 bytes

Figure 5.3

DBSERVER_ENCODING_FIXED specify whether Oracle database encoding is a fixed width. It has 2 possible values – YES or NO. YES indicates that database table column lengths in characters are a fixed width. Use this setting to adjust byte lengths within SAS for any database column lengths that are specified in bytes. The number of characters is calculated by dividing the byte length by the value in DBSERVER_MAX_BYTES=. We can adopt this option when we need to manipulate data when length exceeds the length defined. Just keep in mind the

truncation issue it might lead to. It is just a temporary solution for manipulating large-length data instead of a best approach. For these data, the best approach is to set the length as long as possible at definition stage.

Joining 2 tables from different RDBMS: dbmaster

Imaging a scenario which might be quite common in daily work. You have table1 from Oracle database, and table2 from DB2 database. You want to join these 2 tables, but you also worry about the performance issue this might cause.

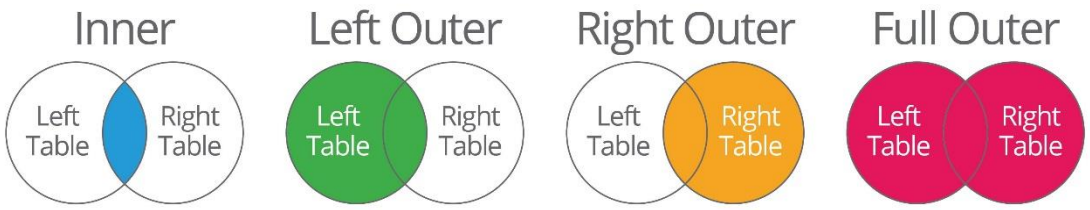


Figure 6.1

This time you can finish the join by virtue of dbmaster option – use this option to specify which table reference in a join is the larger table. This can improve performance by eliminating the processing that is normally performed to determine this information. However, this option is ignored when outer joins are processed.

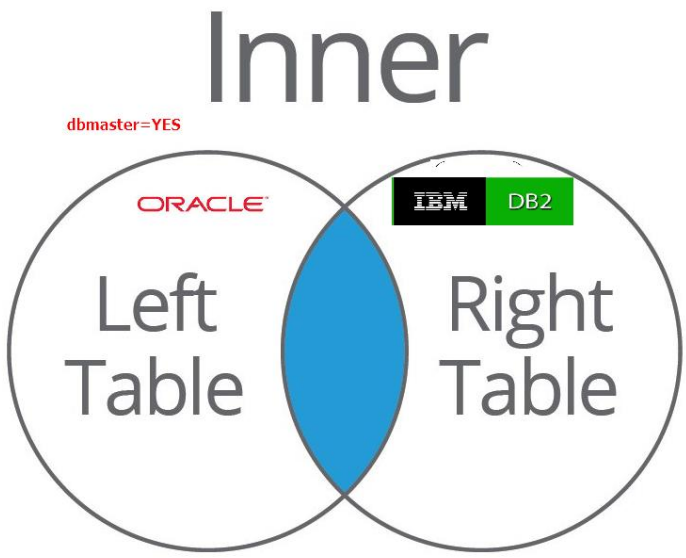


Figure 6.2

```
libname mydblib1 oracle user=<user> pw=<password> path='myorapath';    /*database 1 */
libname mydblib2 db2 user=<user> pw=<password> path='mydb2path';       /*database 2 */
```

```
proc sql;
   select * from mydblib1.bigtab(dbmaster=yes), mydblib2.smalltab
   where bigtab.key=smalltab.key;
quit;
```

Order! Order!

Sometimes it is more efficient to extract or copy DBMS data to a SAS data file than to repeatedly read the data from DBMS. SAS data files are organized to provide optimal performance with PROC and DATA steps. Programs that use SAS data files often outperform SAS programs that read DBMS data.
Consider a scenario, we need to sort the dataset after extract from DBMS. But we know the sort rules for SAS and for DBMS might be different (this is always true). We can specify the sort rule by using option: SORTPGM. Below table explains the option possible value and their meaning:

Value	Purpose
BEST	sorts data according to the DBMS sort rules, the host sort rules, and the SAS sort rules. (Sorting uses the first available and pertinent sorting algorithm in this list.)
HOST	sorts data according to host rules and then SAS rules. (Sorting uses the first available and pertinent sorting algorithm in this list, which might be the most efficient for large tables that contain many rows of data.)
SAS	sort data to use the SAS rules.

Table 7.1

Audit the world

We always have the requirement that we want to check the detail of code processing especially when error or warning happens. SASTRACE and SASTRACELOC options are specific to SAS/ACCESS software. This series of option is a very powerful tool to use when we want to see the commands that SAS/ACCESS sent to our DBMS. It is DBMS-specific. However, most SAS/ACCESS engines show you statements like SELECT or COMMIT as the DBMS processes them for the SAS application.
This is the option definition:

SASTRACE= ',,,d' | ' ,,d,' | ' d,' | 'd,,,' | ',,,db' | ' ,,,s' | ' ,,,sa' | ',,t,'
SASTRACELOC=stdout | SASLOG | FILE 'path-and-filename';
Let's see the option usage for most common value:
',,,d' specifies that all SQL statements that are sent to the DBMS are sent to the log. Here are the applicable statements:

SELECT	DELETE
CREATE	SYSTEM CATALOG
DROP	COMMIT
INSERT	ROLLBACK
UPDATE	

For engines that do not generate SQL statements, API calls and all parameters are sent to the log.

Following is a simple example:

```
data work.winter_birthdays;
    input empid birthdat date9. lastname $18.;
    format birthdat date9.;
datalines;
678999 28DEC1966 PAVEO              JULIANA          3451
456788 12JAN1977 SHIPTON            TIFFANY          3468
890123 20FEB1973 THORSTAD           EDVARD           3329
;
run;

options sastrace=',,,d' sastraceloc=saslog nostsuffix;
libname ora oracle user=<user> password=<password>;
data ora.snow_birthdays;
  set work.winter_birthdays;
run;

libname ora clear;
```

We can see the log info with SASTRACE option enabled:

```
18   options sastrace=',,,d' sastraceloc=saslog nostsuffix;
19   libname ora oracle user=system password=XXXXXXXX;
NOTE: Libref ORA was successfully assigned as follows:
      Engine:        ORACLE
      Physical Name:
20
21   data ora.snow_birthdays;
22      set work.winter_birthdays;
23   run;

ORACLE_1: Prepared: on connection 1
SELECT * FROM SNOW_BIRTHDAYS

NOTE: SAS variable labels, formats, and lengths are not written to DBMS tables.

ORACLE_2: Executed: on connection 2
CREATE TABLE SNOW_BIRTHDAYS(empid NUMBER ,birthdat DATE,lastname VARCHAR2 (72))


ORACLE_3: Prepared: on connection 2
INSERT  INTO SNOW_BIRTHDAYS (empid,birthdat,lastname) VALUES
(:empid,TO_DATE(:birthdat,'DDMONYYYY','NLS_DATE_LANGUAGE=American'),:lastname)

NOTE: There were 3 observations read from the data set WORK.WINTER_BIRTHDAYS.

ORACLE_4: Executed: on connection 2
INSERT statement  ORACLE_3

ORACLE:  *-*-*-*-*-*-* COMMIT *-*-*-*-*-*-*
NOTE: The data set ORA.SNOW_BIRTHDAYS has 3 observations and 3 variables.
ORACLE:  *-*-*-*-*-*-* COMMIT *-*-*-*-*-*-*
NOTE: DATA statement used (Total process time):
      real time           0.17 seconds
      cpu time            0.06 seconds

24
25   libname ora clear;
NOTE: Libref ORA has been deassigned.
```

Conclusion

When dealing with issue related to external data interaction, make sure you use the right SAS statement and options. Although SAS/ACCESS has made it very easy for programmers to accomplish tasks that access external data, there are still some quite useful advanced options deserving your attention. And these options really can make a difference, even simple ones like adopting a sort rule or importing files with correct column names. So, remember to flip thru the documentation specific to your version of SAS.

References

SAS/ACCESS® 9.4 Interface to PC Files: Reference, Fourth Edition
SAS/ACCESS® 9.4 for Relational Databases: Reference, Ninth Edition
SAS® 9.4 SQL Procedure User's Guide, Fourth Edition

Acknowledgments

I would like to express my gratitude to all those who review this paper and give valuable comments.
Specially, I would like to extend my sincere gratitude to my Manager, Han Liu and Director, Jungle Cheng, for their constant encouragement and valuable guidance on my thesis. I am deeply grateful of their help in the completion of this thesis.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Name: Ping Lu
Enterprise: SAS Research and Development (Beijing) Co., Ltd.
Address: Motorola Plaza, No. 1 Wang Jing East Road
City, State ZIP: Beijing, 100102
Work Phone: (8610) 83193355-3812
Fax: (8610) 6310-9130
E-mail: Ping.Lu@sas.com
Web: www.sas.com