CLASS-26 (27-05-2018)
****

PROC TRANSPOSE

The TRANSPOSE procedure creates an output data set by restructuring the values in a SAS data set, transposing selected variables into observations.
The TRANSPOSE procedure can often eliminate the need to write a lengthy DATA step to achieve the same result.
Further, the output data set can be used in subsequent DATA or PROC steps for analysis, reporting, or further data manipulation.
PROC TRANSPOSE does not produce printed output. To print the output data set from the PROC TRANSPOSE step, use PROC PRINT, PROC REPORT, or another SAS reporting tool.
To create **transposed variable**, the procedure transposes the values of an observation in the input data set into values of a variable in the output data set.
Or in simple words: Proc Transpose changes multiple values in rows into columns, and can also change multiple columns'values into multiple rows values for a single column.
Or: Changing Tall data to flat data or flat data to tall data.

Types of Transpositions PROC TRANSPOSE can Perform:

1. Simple Transposition: a) with one numeric variable.

CODE:

```
data pets;
input po$ pet $ pop;
cards;
amit dog 2
neha cat 3
arnab fish 2
ram bird 2
;
run;

proc transpose data=pets out=pets_t;
Run;
```

Input data set

	po	pet	pop
1	amit	dog	2
2	neha	cat	3
3	arnab	fish	2
4	ram	bird	2

Transposed data set

	NAME	COL1	COL2	COL3	COL4
1	pop	2	3	2	2

Note: value of _NAME_ is the name of a variable in the input data set that tells which variable has transposed.
COL1,COL2...COL4 are the default names of the transposed columns.
Only numeric variables are transposed in simple transposition unless you tell it otherwise.

---

b) with two numeric variable:

CODE:

```
data pets;
input po$ pet $ pop cost;
cards;
amit dog 2 1200
neha cat 3 2500
arnab fish 2 2300
ram bird 2 500
;
run;

proc transpose data=pets out=pets_t;
run;
```

	NAME	COL1	COL2	COL3	COL4
1	pop	2	3	2	2

2	cost	1200	2500	2300	500

Note: If we do not define anything then in simple transpose the numeric variables are transposed whether we have 1,2 or multiple numeric variables all will get transposed.

---

2. PREFIX option: Prefix option provides a prefix to the transposed column names instead of COL1, COL2, etc

CODE:

```
data pets;
input po$ pet $ pop ;
cards;
amit dog 2
neha cat 3
arnab fish 2
ram bird 2
;
run;

proc transpose data=pets out=pets_t  prefix=pet;
Run;
```

	NAME	pet1	pet2	pet3	pet4
1	pop	2	3	2	2

Now, col1,col2….col4 has changed to pet1,pet2...pet4 by using prefix.

---

3. NAME option: Name option provides the name for an output file column which tells which input variables were transposed.

CODE:

```
data pets;
input po$ pet $ pop ;
cards;
amit dog 2
ravi dog 2
neha cat 3
arnab fish 2
ram bird 2
;
```

```
run;

proc transpose data=pets  out=pets_t prefix=pet  name=col_transposed;
Run;
```

	col_transposed	pet1	pet2	pet3	pet4
1	pop	2	3	2	2

Here _NAME_ from the table has changed to col_transposed by using name option.

---

4. ID statement : ID statement names the column in the input file whose row values provide the column names in the output file. It will transpose the variable written with it.There should only be one variable in an ID statement. Also, the column used for the ID statement **cannot have** any **duplicate values**.
For eg: Look at the code below.

CODE: with duplicate values

```
data pets;
input po$ pet $ pop ;
cards;
amit dog 2
ravi dog 2
neha cat 3
arnab fish 2
ram bird 2
;
run;

proc transpose data=pets  out=pets_t   name=col_transposed;
id pet;
Run;
```

ERROR: The ID value "dog" occurs twice in the input data set i.e. with amit and ravi and we cannot have two variables with same name in one data set. So, this code will not process because of these errors.

CODE: with no duplicate values

```
data pets;
input po$ pet $ pop ;
cards;
amit dog 2
```

```
ravi doggy 2
neha cat 3
arnab fish 2
ram bird 2
;
run;

proc transpose data=pets  out=pets_t  name=col_transposed;
id pet;
Run;
```

	col_transposed	dog	doggy	cat	fish	bird
1	pop	2	2	3	2	2

Here, duplicate values are removed and ‘id’ is used with pet. So, it will names of pet i.e. dog,doggy….bird will printed instead pet1,pet2...pet5

Exercise : Route wise count from sasuser.cargorev and then transposing the data.

CODE:

```
Proc sql;
Create table route as select route,count(*) as count from sasuser.cargorev group by route;
Quit;
```

	Route	Count
1	Route1	17
2	Route2	4
3	Route3	16
4	Route4	5
5	Route5	6
6	Route6	1
7	Route7	1

```
proc transpose data=route out=route_t  name=col_transposed;
```

```
id route;
Run;
```

	col_transposed	Route1	Route2	Route3	Route4	Route5	Route6	Route7
1	Count	17	4	16	5	6	1	1

---

5. BY statement: BY statement names row-identification variable(s) whose values are not transposed; it requires a preliminary Proc Sort. It is used to handle duplicates.

CODE:

```
data pets;
input po$ pet $ pop ;
cards;
amit dog 92
amit cat 3
amit cow 3
ravi dog 52
neha cat 3
arnab fish 22
ram bird 12
;
Run;

proc sort data=pets;
by po;
Run;                        // by sorting with po, 5 bygroups will be formed i.e. amit,arnab,neha,ram,ravi so
duplicate values wil not clash //

proc transpose data=pets out=pets_t name=col_transposed;
id pet;
by po;
Run;
```

	po	col_transposed	dog	cat	cow	fish	bird
1	amit	pop	92	3	**3**	.	.
2	arnab	pop	.	.	.	22	.
3	neha	pop	.	3	.	.	.

4	ram	pop	.	.	.	.	12
5	ravi	pop	52	.	.	.	.

Note: Null values will show that the particular data is not present with some variables but it will get transposed.

---

6. VAR statement: VAR statement specifies which variables' values are to be transposed; can be character and/or numeric variables; if VAR is omitted, Transpose transposes all numeric vars. VAR controls what we have to transpose.
Here, VAR is used with cost , so only cost will transpose .

CODE:

```
data pets;
input po$ pet $ pop cost;
cards;
amit dog 92 200
amit cat 3 500
arnab fish 22 300
ram bird 12 300
;
run;

proc sort data=pets;
by po;
run;

proc transpose data=pets out=pets_t  name=col_transposed;
id pet;
var cost;
Run;
```

	col_transposed	dog	cat	fish	bird
1	cost	200	500	300	300

---

7. VAR and ID statements:

CODE:

```
data pets;
input po$ pet $ pop cost;
cards;
amit dog 92 200
amit cat 3 500
arnab fish 22 300
ram bird 12 300
;
run;

proc sort data=pets;
by po;
run;

proc transpose data=pets out=pets_t name=col_transposed;
id pet;
var cost;
by po;
Run;
```

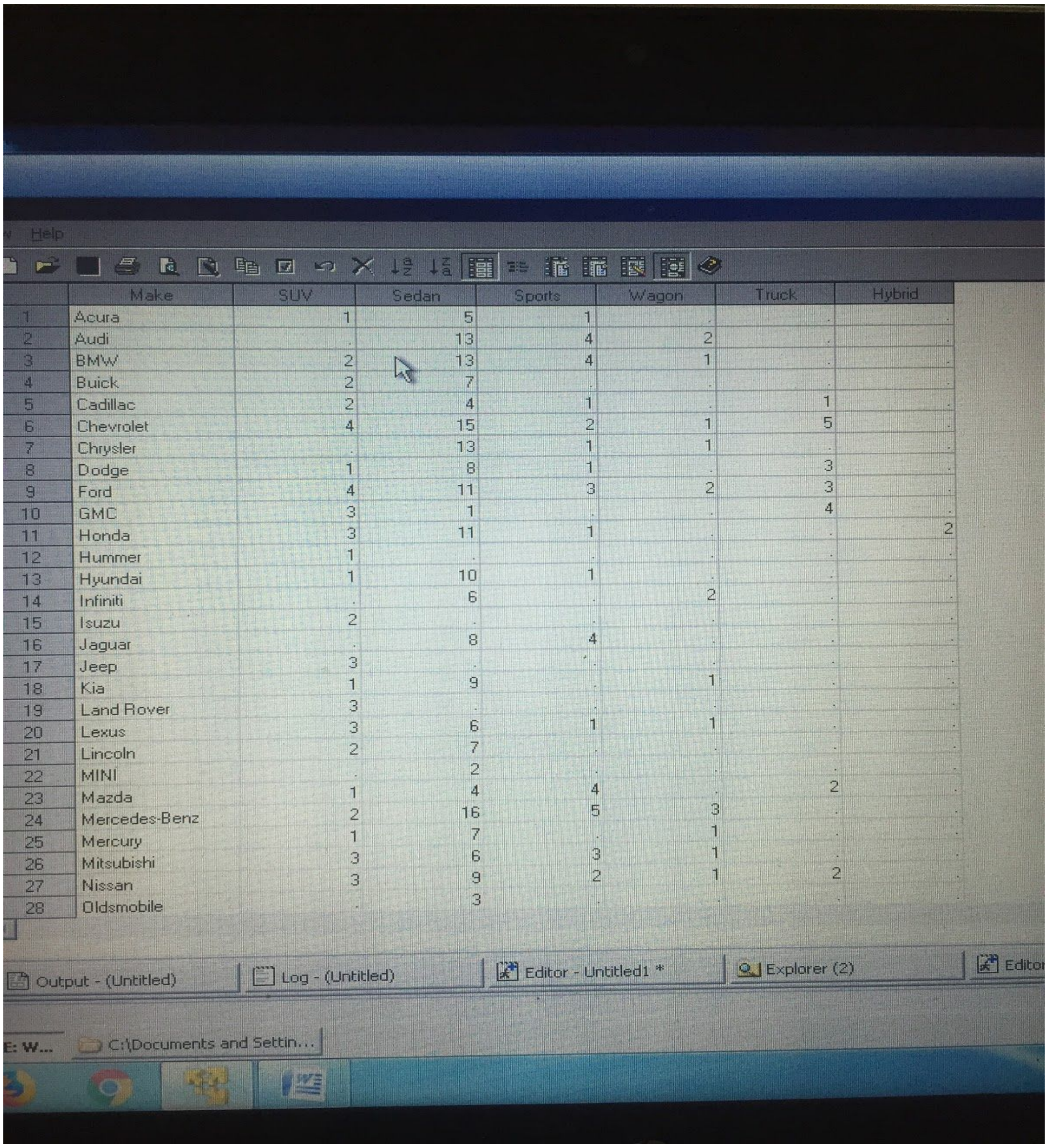
	po	**col_transposed**	**dog**	**cat**	**fish**	**bird**
1	amit	cost	200	500	**.**	**.**
2	arnab	cost	**.**	**.**	300	**.**
3	ram	cost	**.**	**.**	**.**	300

Exercise: make and type wise count

CODE:

```
Proc sql;
Create table car as select make,type,count(*) as count from sashelp.cars group by make,type;
Run;

proc transpose data=car  out=car_t(drop=_:);  // drop=_: will delete all the variables starting with
“_” //
id type;
By make;
Run;
```



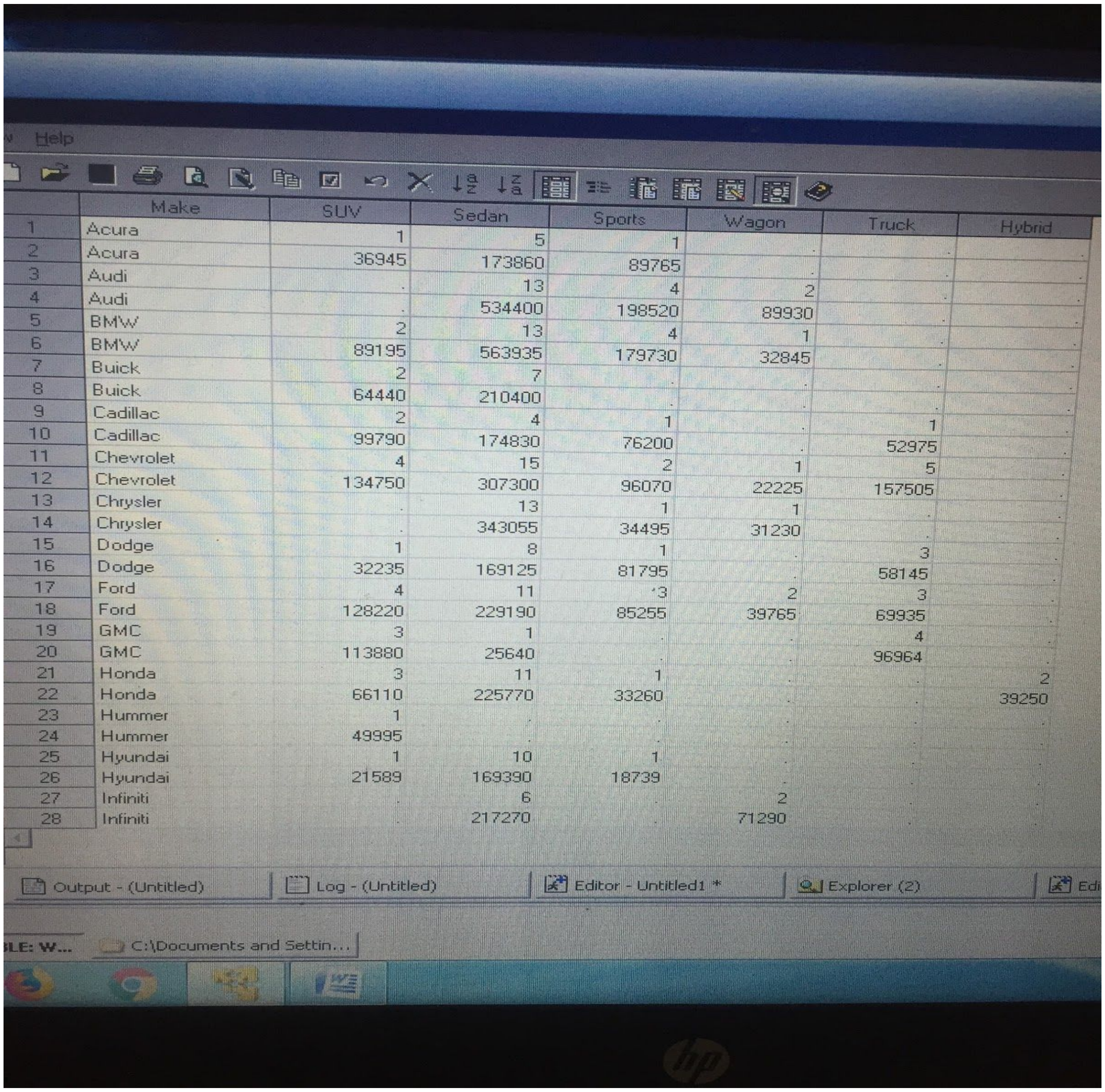
	Make	SUV	Sedan	Sports	Wagon	Truck	Hybrid
1	Acura	1	5	1	.	.	.
2	Audi	.	13	4	2	.	.
3	BMW	2	13	4	1	.	.
4	Buick	2	7	.	.	.	.
5	Cadillac	2	4	1	.	1	.
6	Chevrolet	4	15	2	1	5	.
7	Chrysler	.	13	1	1	.	.
8	Dodge	1	8	1	.	3	.
9	Ford	4	11	3	2	3	.
10	GMC	3	1	.	.	4	.
11	Honda	3	11	1	.	.	2
12	Hummer	1	.	.	.	.	.
13	Hyundai	1	10	1	.	.	.
14	Infiniti	.	6	.	2	.	.
15	Isuzu	2	.	.	.	.	.
16	Jaguar	.	8	4	.	.	.
17	Jeep	3	.	.	.	.	.
18	Kia	1	9	.	1	.	.
19	Land Rover	3	.	.	.	.	.
20	Lexus	3	6	1	1	.	.
21	Lincoln	2	7	.	.	.	.
22	MINI	.	2	.	.	.	.
23	Mazda	1	4	4	.	2	.
24	Mercedes-Benz	2	16	5	3	.	.
25	Mercury	1	7	.	1	.	.
26	Mitsubishi	3	6	3	1	.	.
27	Nissan	3	9	2	1	2	.
28	Oldsmobile	.	3	.	.	.	.

Exercise:From the above example find the sum of maxumum selling retail price(msrp).

CODE:

```
Proc sql;
Create table car as select make,type,count(*) as count, sum(msrp) as rev from sashelp.cars
group by make,type;
Run;
```

```
proc transpose data=car  out=car_t(drop=_:);
id type;
By make;
Run;
```



	Make	SUV	Sedan	Sports	Wagon	Truck	Hybrid
1	Acura	1	5	1	.	.	.
2	Acura	36945	173860	89765	.	.	.
3	Audi	.	13	4	2	.	.
4	Audi	.	534400	198520	89930	.	.
5	BMW	2	13	4	1	.	.
6	BMW	89195	563935	179730	32845	.	.
7	Buick	2	7	.	.	.	.
8	Buick	64440	210400	.	.	.	.
9	Cadillac	2	4	1	.	1	.
10	Cadillac	99790	174830	76200	.	52975	.
11	Chevrolet	4	15	2	1	5	.
12	Chevrolet	134750	307300	96070	22225	157505	.
13	Chrysler	.	13	1	1	.	.
14	Chrysler	.	343055	34495	31230	.	.
15	Dodge	1	8	1	.	3	.
16	Dodge	32235	169125	81795	.	58145	.
17	Ford	4	11	3	2	3	.
18	Ford	128220	229190	85255	39765	69935	.
19	GMC	3	1	.	.	4	.
20	GMC	113880	25640	.	.	96964	.
21	Honda	3	11	1	.	.	2
22	Honda	66110	225770	33260	.	.	39250
23	Hummer	1	.	.	.	.	.
24	Hummer	49995	.	.	.	.	.
25	Hyundai	1	10	1	.	.	.
26	Hyundai	21589	169390	18739	.	.	.
27	Infiniti	.	6	.	2	.	.
28	Infiniti	.	217270	.	71290	.	.

Here , two level information is generated. From every brand , count is shown and the revenue generated from that segment is also shown.

REVERSE TRANSPOSE: Changes Flat data to tall data

CODE: Taking the data from sasuser.cargorev

```
Proc transpose data=route_t  out=route_rev_tran(rename=(col1=count))  name=route;
Run;
```

Transposed data

	col_transposed	Route1	Route2	Route3	Route4	Route5	Route6	Route7
1	Count	17	4	16	5	6	1	1

Reverse Transposed data

	Route	**Count**
1	Route1	17
2	Route2	4
3	Route3	16
4	Route4	5
5	Route5	6
6	Route6	1
7	Route7	1