Working with Dates, Creating Summary Datasets and Some SAS Functions

STAT 3505

Week 5 (February 15, 2024)

Gunes Fleming Ph.D.

TEMPLE
UNIVERSITY

Remember from Previous Class

- Format/ Informat,
- Proc Format,
- Subsetting data using IF and WHERE statements,
- DROP, KEEP, RENAME statements.

Question:

What is the role of “IF” in the example below? Is it used to subset the data?

```
Data newdata;
  Set  olddata;
  If AGE>35 then AGE_new = “Adult”;
Run;
```

Question:

What is the role of “IF” in the example below? Is it used to subset the data?

```
Data newdata;
  Set  olddata;
  If AGE>35 then AGE_new = “Adult”;
Run;
```

Answer: No! The role of IF here is to set a condition based on which a new variable (named AGE_new) is created.

Modify/Manipulate Data in Data Step: **LABELS**

- In order to add variable labels, LABEL statement is used.
- Syntax:

```
LABEL   variablename_1 = ‘description’
        variablename_2 = ‘description’
                  …
        variablename_n = ‘description’;
```

LABELS

- Labels can be up to 40 characters. Each blank counts as a character.
- Single or double quotes can be used for description (do not mix single and double quotes).
- LABEL statement can be placed anywhere in DATA step.

Variable LABELs:

Example:

```
Data newdata;
   Set  olddata;
   Keep AGE WGTBL HGTBL BMI;
    Label AGE     = “Age at Baseline”
          WGTBL = “Weight at Baseline”
          HGTBL  = “Height at Baseline”
          BMI      = “Body Mass Index”;
Run;
```

About SAS Functions

- The syntax for a function is:

```
NEWVARIABLE = FUNCTIONNAME(argument1, argument2.., argumentk);
```

where arguments can be constants, variables, expressions. or other functions.

SAS DATE AND TIME FUNCTIONS

DATE and TIME Functions

- SAS date and time variables are stored as integers and indicate the number of days since January 1, 1960.
- A positive number indicates a date after January 1, 1960, and a negative number indicates a date before January 1, 1960.
- Date values that contain both date and time are stored in SAS as the number of seconds since midnight on January 1, 1960

DATE and TIME Functions

Two ways to designate variables as a date value:

1. Read a value as a date in an INPUT statement.

For example:

```
INPUT BDATE MMDDYY8.;
```

2. Assign a date value to a fixed date.

A “d” at the end of the value tells SAS to interpret this quoted string as a date.

For example:

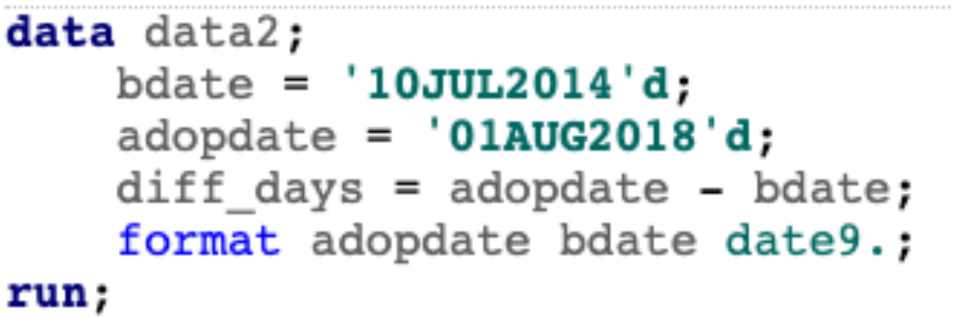
```
BDATE = '12DEC2010’d;
BEGINDATE=“1jan2011”D;
EXAMDATA=‘13-APR-2014’d;
```

NOTE: D at end of string mean date.

Difference in Dates

- Since SAS stores dates in number of days from January 1, 1960, the difference between two dates would be calculated in **days.**

Ex:



```
data data2;
    bdate = '10JUL2014'd;
    adopdate = '01AUG2018'd;
    diff_days = adopdate - bdate;
    format adopdate bdate date9.;
run;
```

Difference in Dates

- DATDIF Function returns the number of days between two dates.
- Syntax: DATDIF(start-date, end-date, < basis>)
- Examples to basis values:
  - ‘ACT/ACT': Actual/actual method. This method calculates the exact number of days between two dates, taking leap years and month lengths into account.
  - ‘ACT/360': This method uses the actual number of calendar days in a particular month, and 360 days as the number of days in a year, regardless of the actual number of days in a year.
  - For more info, check SAS documentation.

Difference in Dates

- Examples for DATDIF:

  - DAYS=DATDIF(‘07JUL1976’d,’01JAN2013’d, ‘30/360’);

  Returns the value (number of days) as 13134. Basis ’30/360’ specifies a 30-day month and 360-day year regardless of the actual number of calendar days in a month or year.

  - DAYS=DATDIF(‘07JUL1976’d,’01JAN2013’d, ’ACT/ACT’);

  Returns the value (number of days) as 13327.

(Different BASIS are used in different disciplines.)

Difference in Dates

- To get the difference between two dates in **years**:
  - One way is to use arithmetic difference

    Year_diff = (Datevar2 – Datevar1)/365.25

  - Another way is to use YRDIF function:

    Year_diff = YRDIF(Datevar1, Datevar2, “ACTUAL”)

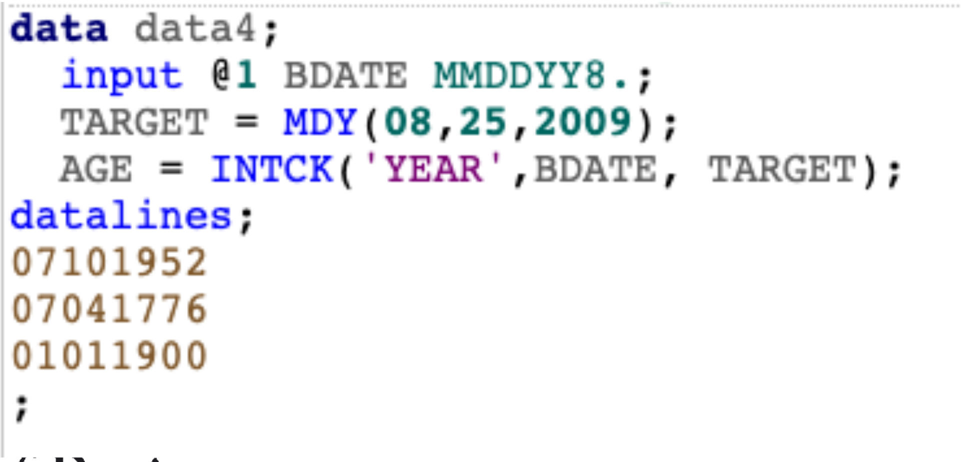
Difference in Dates

- YRDIF function returns the difference in years between two dates, taking into account fractional parts of a year if required.
- Syntax: YRDIF(start-date, end-date, < basis>)
- Examples to basis values:
  - ‘ACT/ACT': Actual/actual method. This method calculates the exact number of years between two dates, taking leap years into account.
  - 'AGE': Age method. This method calculates the difference in years based on a 365-day year, ignoring leap years. This method is often used in financial calculations.
  - For more info, check SAS documentation.

Some Other SAS Date/Time Functions:

- MDY(month,day,year): Creates and returns a SAS date from month, day, year.
- INTCK('interval',from, to): Returns the number of time intervals in a given time span, where interval can be DAY, WEEKDAY, YEAR etc.

Example:



```
data data4;
   input @1 BDATE MMDDYY8.;
   TARGET = MDY(08,25,2009);
   AGE = INTCK('YEAR',BDATE, TARGET);
datalines;
07101952
07041776
01011900
;
```

Some Other SAS Date/Time Functions:

- Rerun the same example with some changes:

  - Change

    TARGET = MDY(**08,25,2009**); to

    TARGET = ’**25-OCT-2009**’d;

  - Change

    AGE = INTCK('YEAR',BDATE,TARGET); to

    WEEKS = INTCK(‘WEEK',BDATE,TARGET);

Some Other SAS Date/Time Functions:

- MONTH(datevar):
  - Extracts the month component from a SAS date value.
  - Returns an integer between 1 and 12 representing the month of the specified date.

- WEEKDAY(datevar)
  - Produces an integer that represents the day of the week from 1 and 7, where 1 = Sunday, 2 = Monday.. 7= Saturday.

- DAY(datevar)
  - Produces an integer from 1 to 31 that represents the day of the month.

Some Other Tricks:

```
*Get today;
TODAY=TODAY() ;


* Get the last day of previous month ;
END =TODAY-DAY(TODAY) ;


* Get the first day of previous month ;
START=END-DAY(END)+1;
```

Example: Calculate Time to Surgery

```
 * REQD DATE AND TIME OF ER ARRIVAL AND SURGERY;
data ER;
input @1 DATE_ARRIVE DATE9. @11 TIME_ARRIVE  time5. @16 DATETIME_SURGERY datetime15.;
format DATE_ARRIVE monyy7.
        DATETIME_SURGERY DATETIME_ARRIVE datetime15.
                                        TIME_arrive time11.;
* How long until surgery?;
  DATETIME_ARRIVE=(DATE_ARRIVE*  24 * 60 * 60) + TIME_ARRIVE;
  MINUTES_TO_SURGERY=(DATETIME_SURGERY-DATETIME_ARRIVE)/ 60;
  datalines;
  12jan2014 7:10 12jan2014/10:33
  12Jan2014 19:11 13jan2014/1:01
  ;
  run;
  proc print;
  run;
```

Convert date into seconds

Calculate difference in seconds, convert to minutes

Example: Calculate Time to Surgery

Results – Time to Surgery:

Obs	DATE_ARRIVE	TIME_ARRIVE	DATETIME_SURGERY	DATETIME_ARRIVE	MINUTES_TO_SURGERY
1	JAN2014	7:10:00	12JAN14:10:33	12JAN14:07:10	203
2	JAN2014	19:11:00	13JAN14:01:01	12JAN14:19:11	350

Interesting Fact about SAS Date Cutoff:

- What happens if the year in date variable is in two-digit years?
- How does SAS read 10/21/04?
- Would this be October 21, 1904 or October 21, 2004?

Interesting Fact about SAS Date Cutoff:

- What happens if the year in date variable is in two-digit years?
- How does SAS read 10/21/04?
- Would this be October 21, 1904 or October 21, 2004?
- SAS has an extremely useful system option:

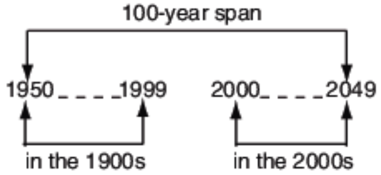
```
YEARCUTOFF = value;
```

The “value” marks the beginning of a 100-year window, and it should be a four-digit year

Interesting Fact about SAS Date Cutoff:

- If the default value of nnnn (1940) is in effect, the 100-year span begins with 1940 and ends with 2039.
- Therefore, any informat or function that uses a two-digit year value that ranges from 40 to 99 assumes a prefix of 19. For example, the value 92 refers to the year 1992.
- If you specify for example, YEARCUTOFF=1950, any two-digit value between 50 and 99 inclusive refers to the first half of the 100-year span, which is in the 1900s.

A 100-Year Span with Values in Two Centuries



***To get current cutoff at any time, run the following code:

```
%put Current Year Cutoff: %sysfunc(getoption(yearcutoff));
```

Some More SAS Functions:

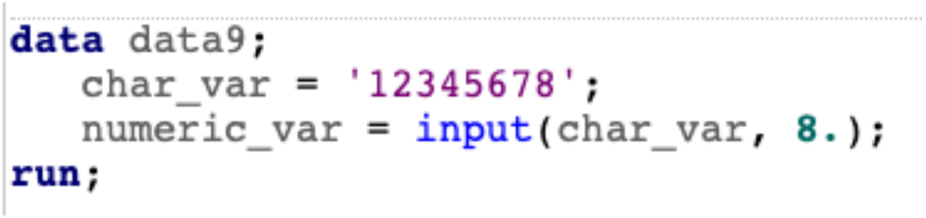
- **UPCASE**(expression): copies a character expression, converts all lowercase letters to uppercase letters, and returns the altered value as a result.

- **LOWCASE**(expression): copies a character expression, converts all uppercase letters to lowercase letters, and returns the altered value as a result.

- **PROPCASE**(expression): copies a character argument and converts all uppercase letters to lowercase letters. It then converts to uppercase the first character of a word that is preceded by a blank, forward slash, hyphen, open parenthesis, period, or tab.

Special Use Functions

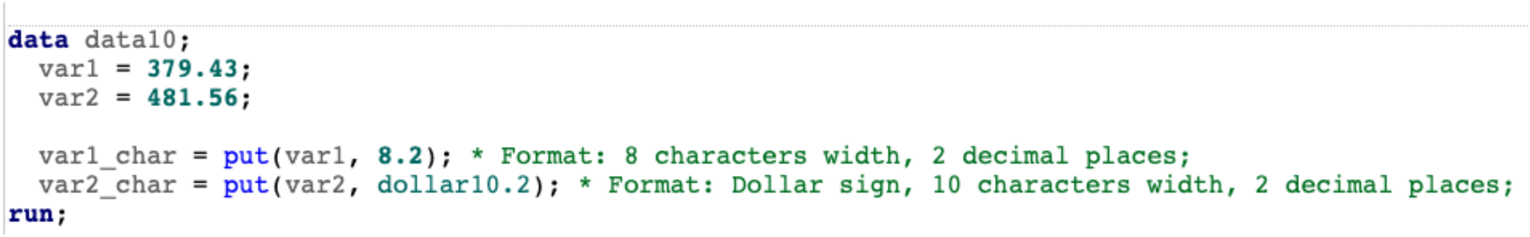
- **INPUT Function**:
  - Syntax: input(*original_variable*, *informat.*);
  - Converts a character expression to character or numeric using a specified informat.
  - The informat tells SAS how to interpret the data in the original character variable.
  - Example:



```
data data9;
   char_var = '12345678';
   numeric_var = input(char_var, 8.);
run;
```

Special Use Functions

- **PUT Function**：
  - Syntax: put(*original_variable*, *format.*);
  - Converts a numeric (or character) expression to character using a specified format.
  - The informat contains the SAS format that you want applied to the value that is specified in the original variable.
  - Example:

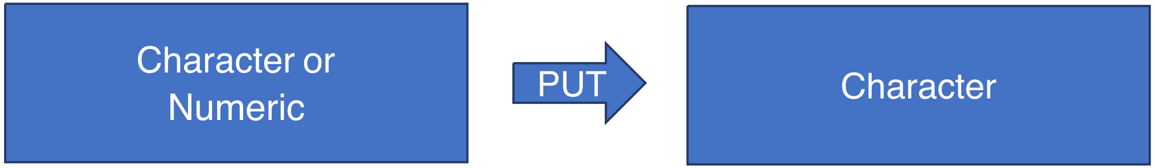


```
data data10;
  var1 = 379.43;
  var2 = 481.56;

  var1_char = put(var1, 8.2); * Format: 8 characters width, 2 decimal places;
  var2_char = put(var2, dollar10.2); * Format: Dollar sign, 10 characters width, 2 decimal places;
run;
```

In Short, PUT and INPUT Functions:

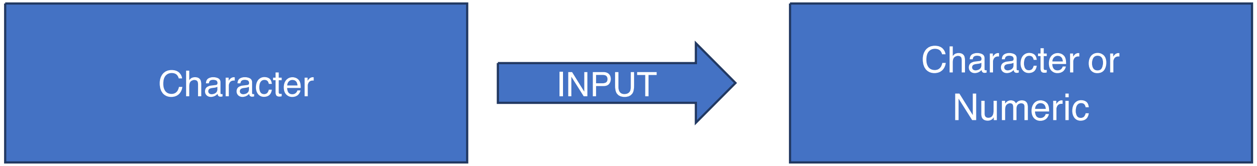
- **PUT** – Converts from Character or Numeric to Character


Character or Numeric

PUT

Character

- **INPUT** – Converts from Character to Character or Numeric


Character

INPUT

Character or Numeric

If original value is NUMERIC – you must use the PUT function.
If you want result to be NUMERIC -- you must use the INPUT function.

Note:

- Please note that there are also other versions of INPUT and PUT:
- INPUTN, INPUTC, PUTN, PUTC
- These functions give you additional control over the formats used in a conversion.
- We will not cover them further in class – feel free to look into those!

CREATING SUMMARY DATA SETS

CREATING SUMMARY DATASET WITH PROC MEANS

- We need to include OUTPUT statement in PROC MEANS to create a summary dataset.

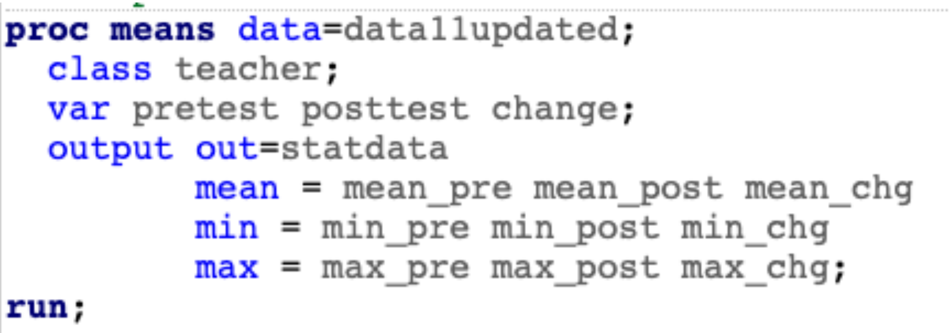
- Syntax:

```
proc means data=your_dataset;
    var var1 var2 ... vark;   *Numeric variables to calculate summary statistics of;
    output out = output_dataset_name /* Specify the output dataset and the desired statistics to be saved */
            mean = mean_var1 mean_vari2 ...;
run;
```

- Note/recall that you can specify additional options for the desired statistics such as mean, sum, min, max, median, etc. If no statistics are specified, PROC MEANS calculates the default statistics: N, Mean, Std Dev, Minimum, and Maximum.

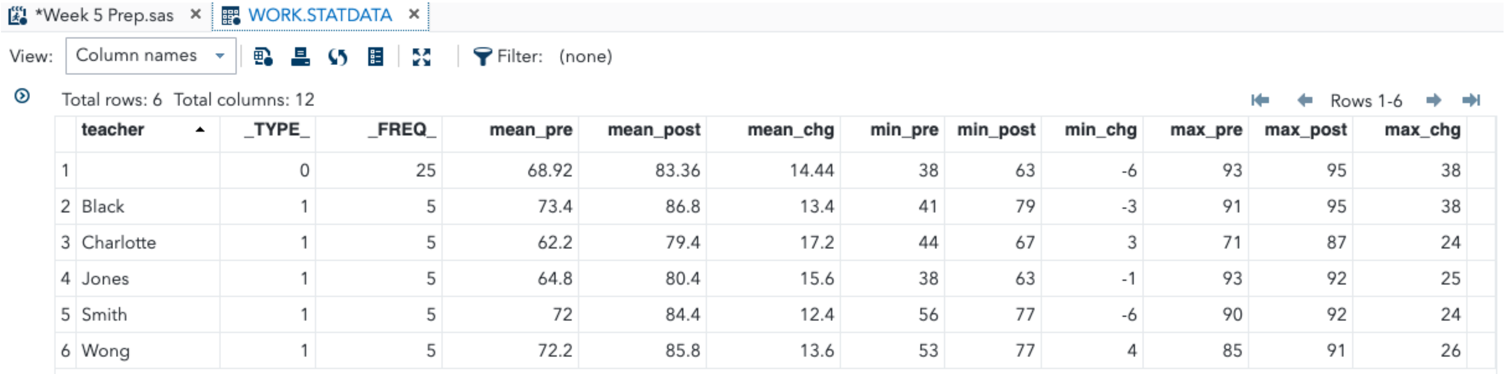
CREATING SUMMARY DATASET WITH PROC MEANS

- Example:



```
proc means data=data11updated;
  class teacher;
  var pretest posttest change;
  output out=statdata
         mean = mean_pre mean_post mean_chg
         min = min_pre min_post min_chg
         max = max_pre max_post max_chg;
run;
```

- Output:

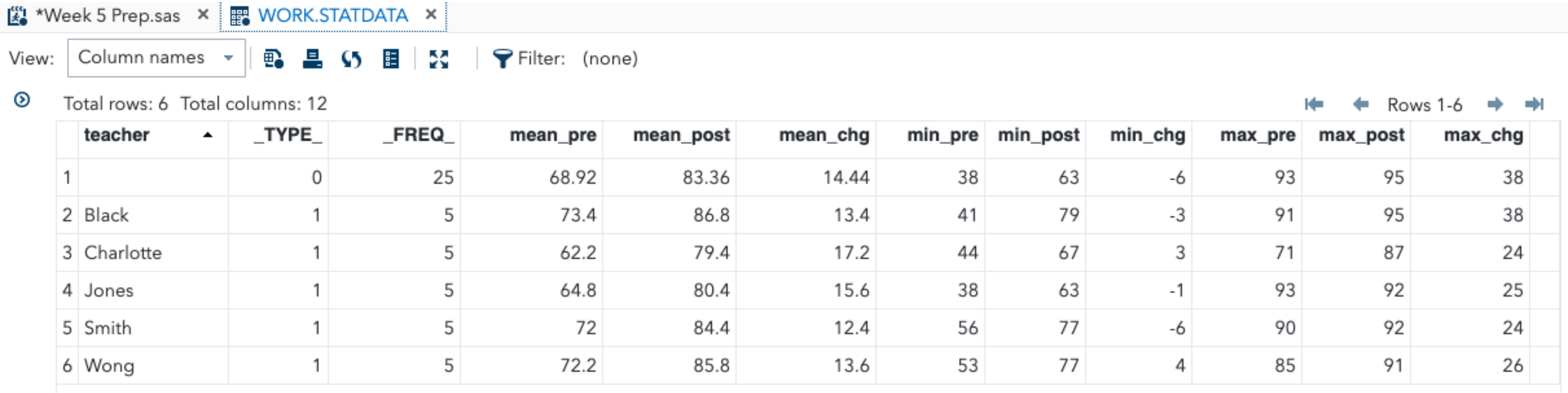


Total rows: 6 Total columns: 12

	teacher	_TYPE_	_FREQ_	mean_pre	mean_post	mean_chg	min_pre	min_post	min_chg	max_pre	max_post	max_chg
1		0	25	68.92	83.36	14.44	38	63	-6	93	95	38
2	Black	1	5	73.4	86.8	13.4	41	79	-3	91	95	38
3	Charlotte	1	5	62.2	79.4	17.2	44	67	3	71	87	24
4	Jones	1	5	64.8	80.4	15.6	38	63	-1	93	92	25
5	Smith	1	5	72	84.4	12.4	56	77	-6	90	92	24
6	Wong	1	5	72.2	85.8	13.6	53	77	4	85	91	26

CREATING SUMMARY DATASET WITH PROC MEANS

- Output:



*Week 5 Prep.sas × | WORK.STATDATA ×

View: Column names | Filter: (none)

Total rows: 6 Total columns: 12 | Rows 1-6

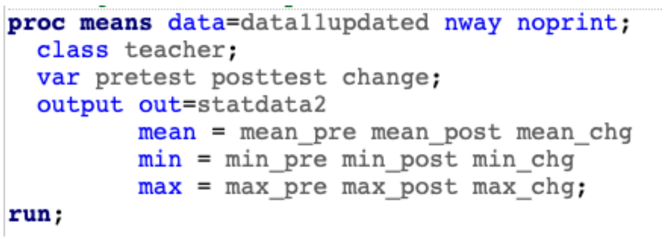
	teacher	_TYPE_	_FREQ_	mean_pre	mean_post	mean_chg	min_pre	min_post	min_chg	max_pre	max_post	max_chg
1		0	25	68.92	83.36	14.44	38	63	-6	93	95	38
2	Black	1	5	73.4	86.8	13.4	41	79	-3	91	95	38
3	Charlotte	1	5	62.2	79.4	17.2	44	67	3	71	87	24
4	Jones	1	5	64.8	80.4	15.6	38	63	-1	93	92	25
5	Smith	1	5	72	84.4	12.4	56	77	-6	90	92	24
6	Wong	1	5	72.2	85.8	13.6	53	77	4	85	91	26

- Note that
  - _FREQ_ gives number of observations (missing or non-missing)
  - _TYPE_ : the first observation with 0 is mean of all non-missing

    the observations with 1 are means broken down by the class variable Teacher.

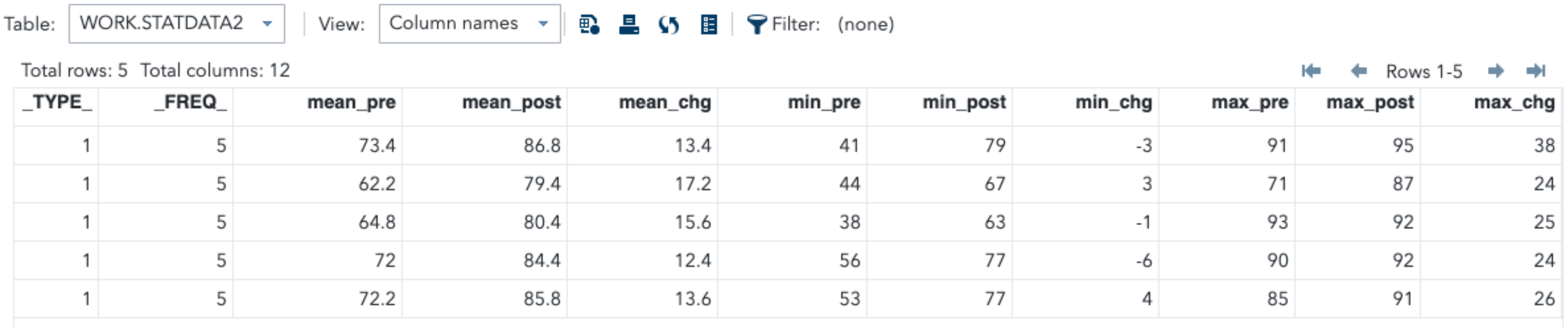
CREATING SUMMARY DATASET WITH PROC MEANS

- To get only the stats for each Teacher category, use NWAY option:



```
proc means data=datallupdated nway noprint;
  class teacher;
  var pretest posttest change;
  output out=statdata2
         mean = mean_pre mean_post mean_chg
         min = min_pre min_post min_chg
         max = max_pre max_post max_chg;
run;
```

- Output:



Total rows: 5 Total columns: 12

TYPE	_FREQ_	mean_pre	mean_post	mean_chg	min_pre	min_post	min_chg	max_pre	max_post	max_chg
1	5	73.4	86.8	13.4	41	79	-3	91	95	38
1	5	62.2	79.4	17.2	44	67	3	71	87	24
1	5	64.8	80.4	15.6	38	63	-1	93	92	25
1	5	72	84.4	12.4	56	77	-6	90	92	24
1	5	72.2	85.8	13.6	53	77	4	85	91	26