SAS Functions

Streamlining data manipulation, simplifying complex logic, and improving code efficiency

Presented by: Carleigh Jo Crabtree

Email: CarleighJo.Crabtree@sas.com

LinkedIn: Carleigh Jo Crabtree | LinkedIn

Replacing IF-THEN Conditional Processing

Manipulating, Creating, and Shifting Dates

Converting Column Types

Manipulating Character Values

Eliminating Case Sensitivity on the WHERE Statement

Bonus:

Using Patterns to Manipulate Data

Modifying Character Column Length Using a Macro

Replacing IF-THEN Conditional Processing

```
IF expression THEN action;
ELSE IF expression THEN action;
ELSE action;
```

IFC(*expression, value-when-true, value-when-false, value-when-missing* **)**

Returns a **character** value based on whether an expression is true, false, or missing

IFN(*expression, value-when-true, value-when-false, value-when-missing* **)**

Returns a **numeric** value based on whether an expression is true, false, or missing

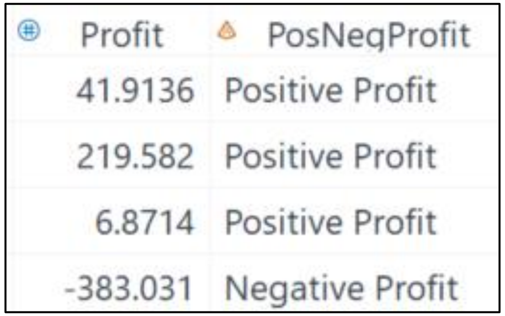
```
if Profit>0 then PosNegProfit="Positive Profit";
else if Profit<=0 then PosNegProfit="Negative Profit";
else PosNegProfit="Missing- Investigate";
```

```
PosNegProfit=
ifc(Profit>0, "Positive Profit", "Negative Profit", "Missing- Investigate");
```

Value if expression is true

Value if expression is false

Value if missing



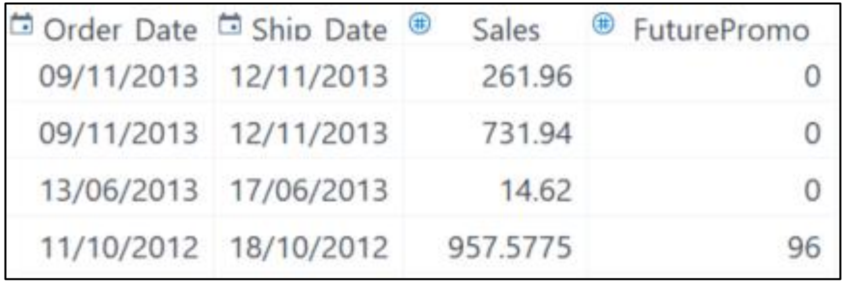
Profit	PosNegProfit
41.9136	Positive Profit
219.582	Positive Profit
6.8714	Positive Profit
-383.031	Negative Profit

```
if (Ship_Date-Order_Date)>5 and Sales>500 then FuturePromo=round(Sales*.1);
else FuturePromo=0;
```

```
FuturePromo=ifn((Ship_Date-Order_Date)>5 and Sales>500, round(Sales*.1), 0);
```

Value if expression is true

Value if expression is false



Order Date	Ship Date	Sales	FuturePromo
09/11/2013	12/11/2013	261.96	0
09/11/2013	12/11/2013	731.94	0
13/06/2013	17/06/2013	14.62	0
11/10/2012	18/10/2012	957.5775	96

Replacing IF-THEN Conditional Processing

Manipulating, Creating, and Shifting Dates

Converting Column Types

Manipulating Character Values

Eliminating Case Sensitivity on the WHERE Statement

Bonus:
Using Patterns to Manipulate Data
Modifying Character Column Length Using a Macro

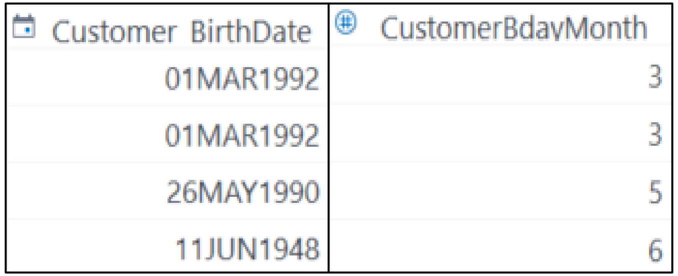
Manipulating, Creating, and Shifting Dates

Manipulating Dates	**MONTH(***dateCol***)**
	YEAR(*dateCol***)**
	DAY(*dateCol***)**
	QTR(*dateCol***)**

Creating Dates	**MDY(***month, day, year***)**

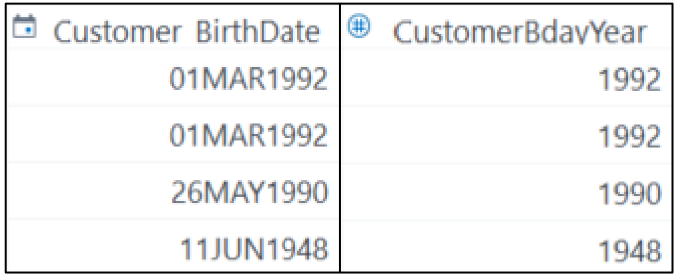
Shifting Dates	**INTNX(***interval, startDate, increment, 'alignment'***)**
	INTCK(*interval, startDate, endDate, 'method'***)**

```
CustomerBdayMonth=month(customer_birthdate);
```



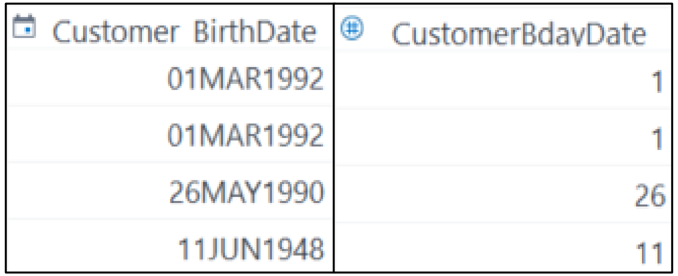
Customer BirthDate	CustomerBdayMonth
01MAR1992	3
01MAR1992	3
26MAY1990	5
11JUN1948	6

```
CustomerBdayYear=year(customer_birthdate);
```



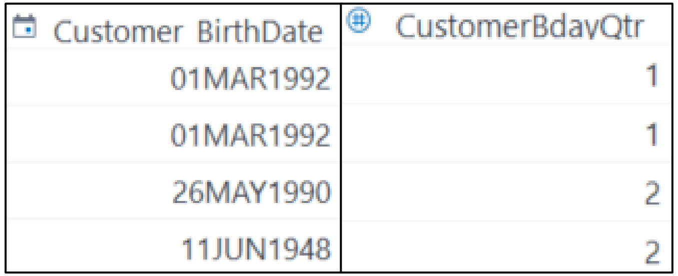
Customer BirthDate	CustomerBdayYear
01MAR1992	1992
01MAR1992	1992
26MAY1990	1990
11JUN1948	1948

```
CustomerBdayDate=day(customer_birthdate);
```

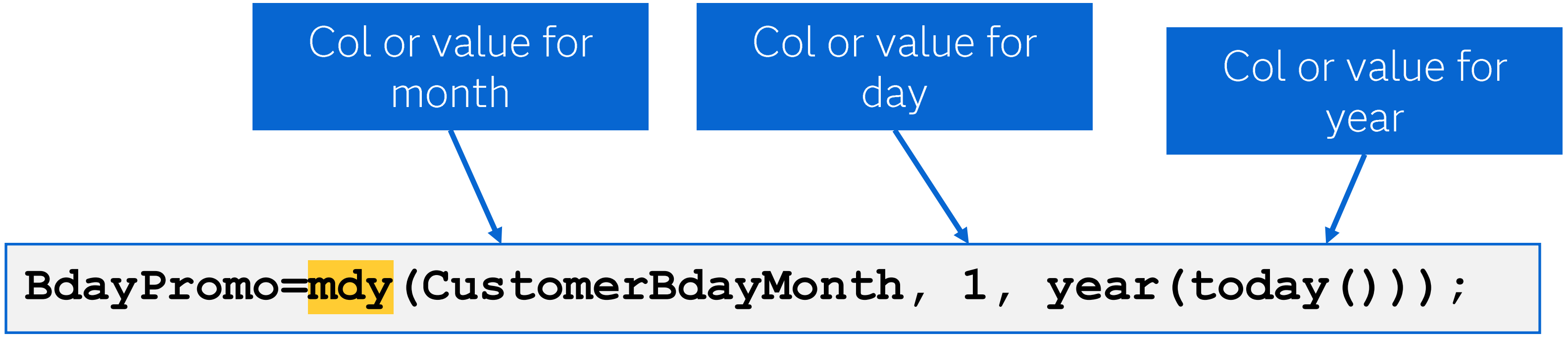


Customer BirthDate	CustomerBdayDate
01MAR1992	1
01MAR1992	1
26MAY1990	26
11JUN1948	11

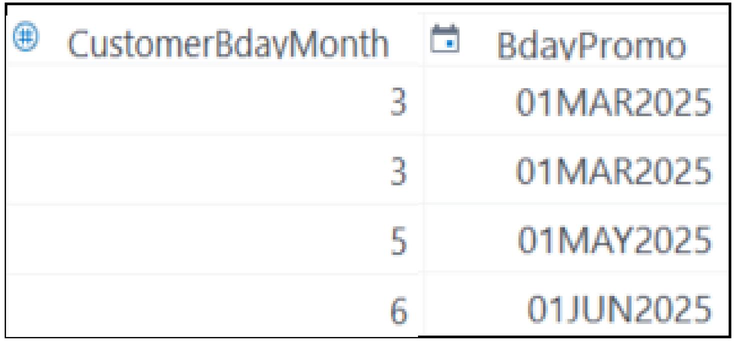
```
CustomerBdayQtr=qtr(customer_birthdate);
```



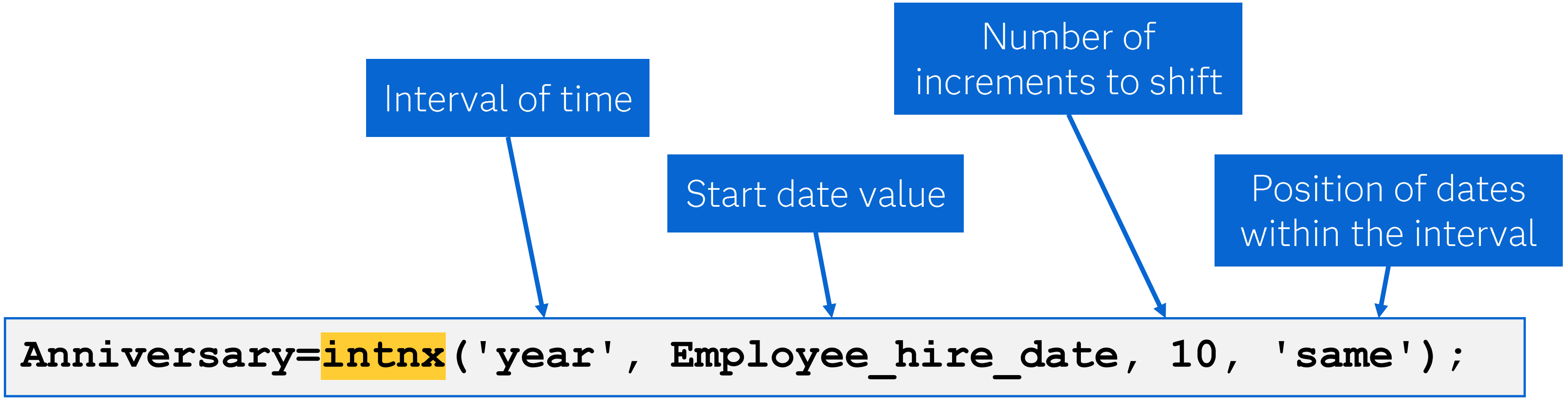
Customer_BirthDate	CustomerBdayQtr
01MAR1992	1
01MAR1992	1
26MAY1990	2
11JUN1948	2



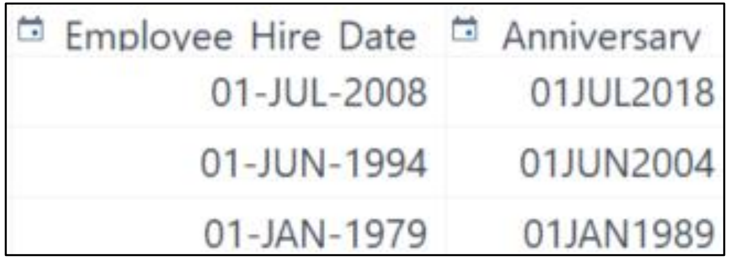
```
BdayPromo=mdy(CustomerBdayMonth, 1, year(today()));
```



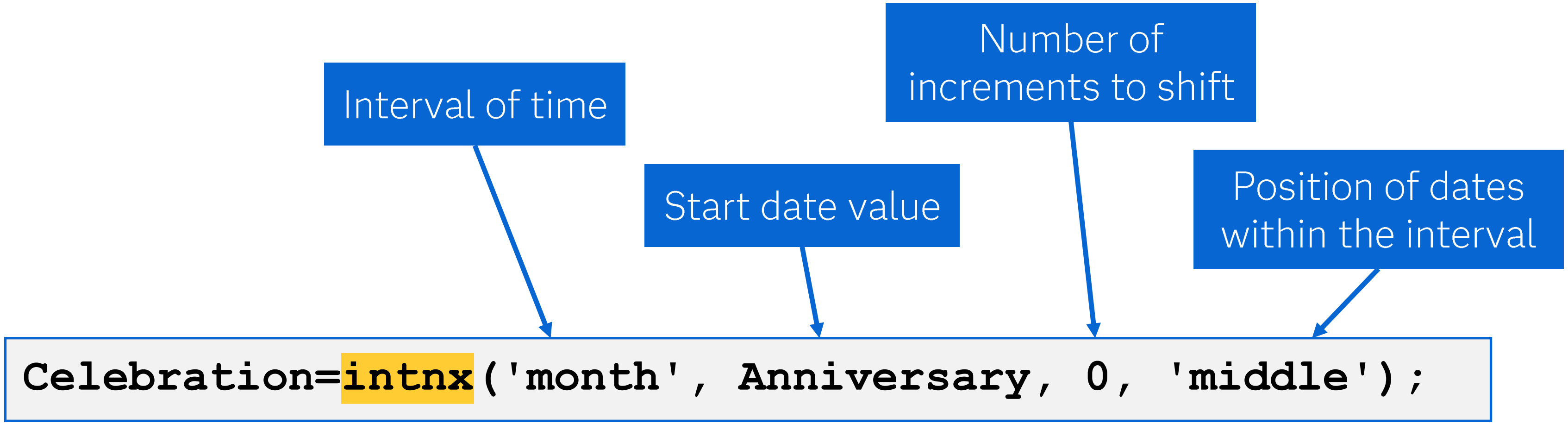
CustomerBdayMonth	BdayPromo
3	01MAR2025
3	01MAR2025
5	01MAY2025
6	01JUN2025


Interval of time
Start date value
Number of increments to shift
Position of dates within the interval

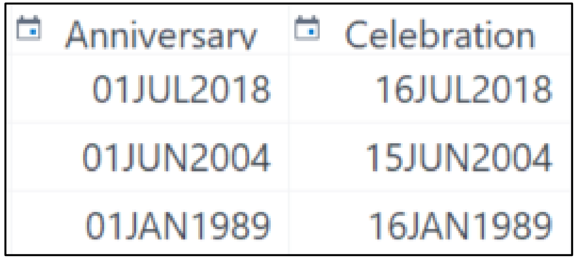
```
Anniversary=intnx('year', Employee_hire_date, 10, 'same');
```



Employee Hire Date	Anniversary
01-JUL-2008	01JUL2018
01-JUN-1994	01JUN2004
01-JAN-1979	01JAN1989


Interval of time
Start date value
Number of increments to shift
Position of dates within the interval

```
Celebration=intnx('month', Anniversary, 0, 'middle');
```



Anniversary	Celebration
01JUL2018	16JUL2018
01JUN2004	15JUN2004
01JAN1989	16JAN1989

Interval of time

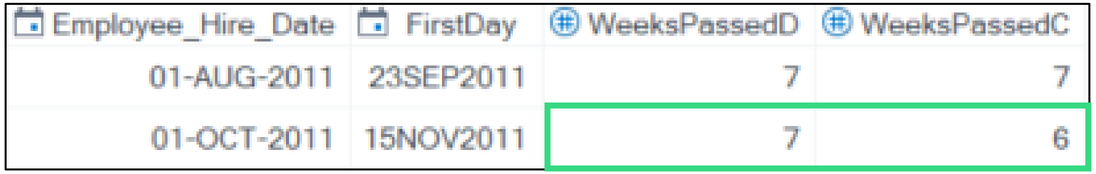
Start date value

End date value

Method to count intervals

```
WeeksPassedD=intck('week', employee_hire_date, FirstDay, 'd');
```

```
WeeksPassedC=intck('week', employee_hire_date, FirstDay, 'c');
```



Employee_Hire_Date	FirstDay	WeeksPassedD	WeeksPassedC
01-AUG-2011	23SEP2011	7	7
01-OCT-2011	15NOV2011	7	6

Discrete 'd'	**Continuous 'c'**
Counts interval boundaries Ex: end of the year, end of the week	Counts interval boundaries based on the start date

Discrete
'd'

October, 2011

Sun	Mon	Tue	Wed	Thu	Fri	Sat	
						1 - begin	1
2	3	4	5	6	7	8	2
9	10	11	12	13	14	15	3
16	17	18	19	20	21	22	4
23	24	25	26	27	28	29	5
30	31						

November, 2011

Sun	Mon	Tue	Wed	Thu	Fri	Sat	
		1	2	3	4	5	6
6	7	8	9	10	11	12	7
13	14	**15 - end**					

Continuous
'c'

October, 2011

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1 - begin
2	3	4	5	6	7	① 8
9	10	11	12	13	14	② 15
16	17	18	19	20	21	③ 22
23	24	25	26	27	28	④ 29
30	31					

November, 2011

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	⑤ 5
6	7	8	9	10	11	⑥ 12
13	14	15 - end				

Replacing IF-THEN Conditional Processing

Manipulating, Creating, and Shifting Dates

Converting Column Types

Manipulating Character Values

Eliminating Case Sensitivity on the WHERE Statement

Bonus:

Using Patterns to Manipulate Data

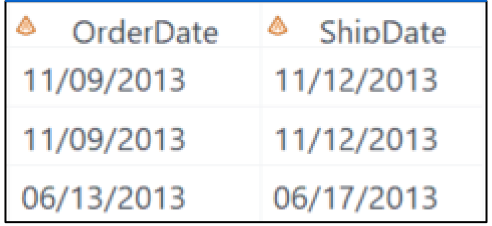
Modifying Character Column Length Using a Macro

Converting Column Types

INPUT(*source, informat.***)**	Converts character values to **numeric** values using a specified **informat**

PUT(*source, format.***)**	Converts numeric or character values to **character** values using a specified **format**

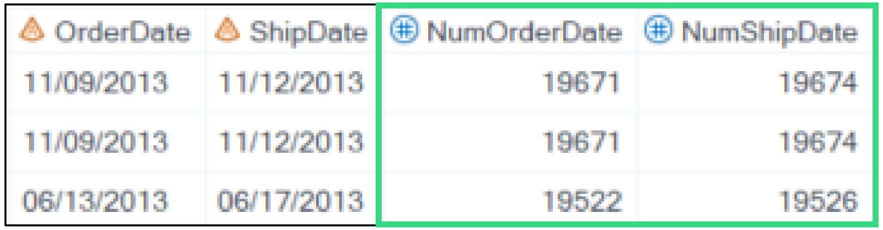
Character columns with dates



OrderDate	ShipDate
11/09/2013	11/12/2013
11/09/2013	11/12/2013
06/13/2013	06/17/2013

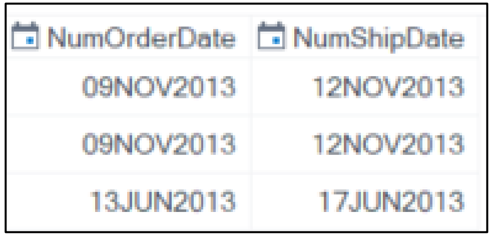
Create numeric columns- what informat can be used?

```
NumOrderDate=input(OrderDate, mmddyy10.);
NumShipDate=input(ShipDate, mmddyy10.);
```



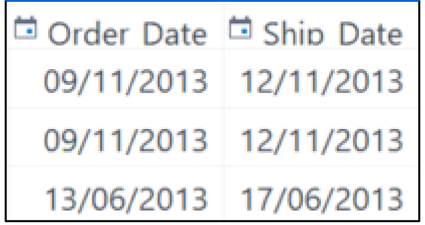
OrderDate	ShipDate	NumOrderDate	NumShipDate
11/09/2013	11/12/2013	19671	19674
11/09/2013	11/12/2013	19671	19674
06/13/2013	06/17/2013	19522	19526

```
format NumOrderDate NumShipDate date9.;
```



NumOrderDate	NumShipDate
09NOV2013	12NOV2013
09NOV2013	12NOV2013
13JUN2013	17JUN2013

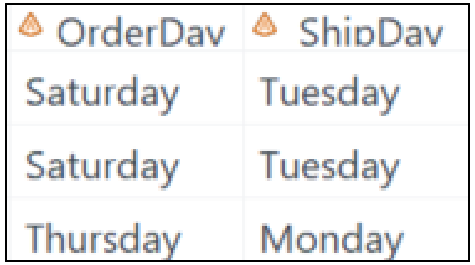
Numeric columns with dates



Order Date	Ship Date
09/11/2013	12/11/2013
09/11/2013	12/11/2013
13/06/2013	17/06/2013

Create **character** columns with the day of week

```
OrderDay=strip(put(Order_Date, downame.));
ShipDay=strip(put(Ship_Date, downame.));
```



OrderDay	ShipDay
Saturday	Tuesday
Saturday	Tuesday
Thursday	Monday

Replacing IF-THEN Conditional Processing

Manipulating, Creating, and Shifting Dates

Converting Column Types

Manipulating Character Values

Eliminating Case Sensitivity on the WHERE Statement

Bonus:

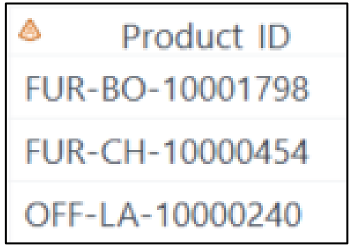
Using Patterns to Manipulate Data

Modifying Character Column Length Using a Macro

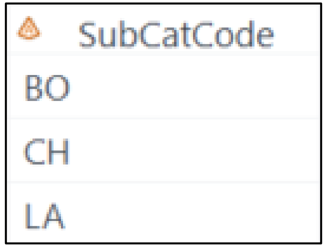
Manipulating Character Values

SCAN(*string, count, character-list, modifiers* **)**

```
SubCatCode=scan(product_id, 2, '-');
```



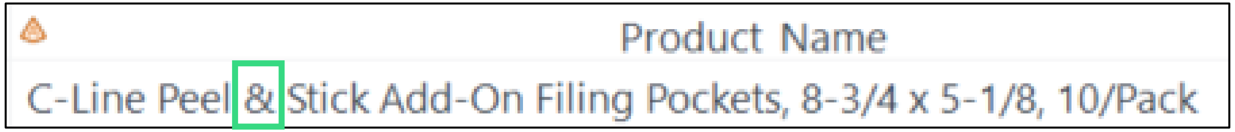
Product ID
FUR-BO-10001798
FUR-CH-10000454
OFF-LA-10000240



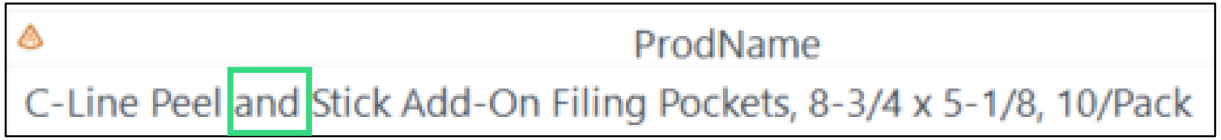
SubCatCode
BO
CH
LA

TRANWRD(*source, target, replacement* **)**

```
ProdName=tranwrd(Product_Name, "&", "and");
```



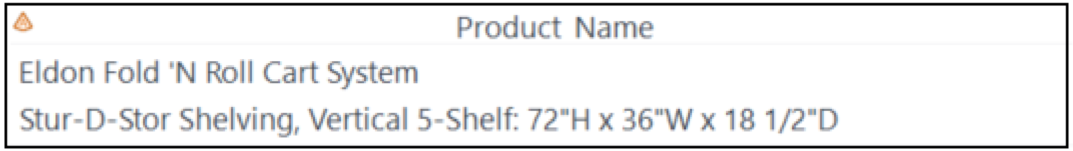
Product Name
C-Line Peel & Stick Add-On Filing Pockets, 8-3/4 x 5-1/8, 10/Pack



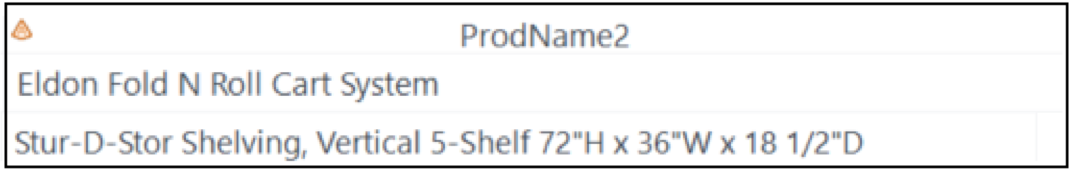
ProdName
C-Line Peel and Stick Add-On Filing Pockets, 8-3/4 x 5-1/8, 10/Pack

COMPRESS(*source, characters, modifiers* **)**

```
ProdName2=compress(ProdName, "':" );
```



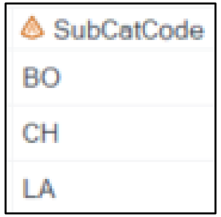
Product Name
Eldon Fold 'N Roll Cart System
Stur-D-Stor Shelving, Vertical 5-Shelf: 72"H x 36"W x 18 1/2"D



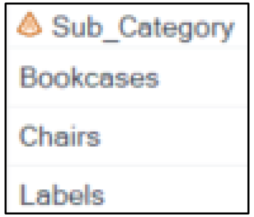
ProdName2
Eldon Fold N Roll Cart System
Stur-D-Stor Shelving, Vertical 5-Shelf 72"H x 36"W x 18 1/2"D

CATX(*delimiter, item1, itemn…* **)**

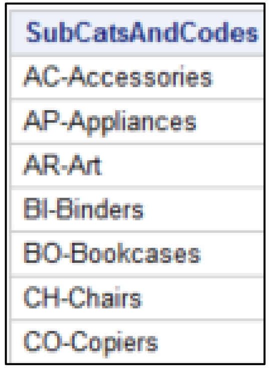
```
select distinct catx('-', SubCatCode, Sub_Category) as SubCatsAndCodes
```



SubCatCode
BO
CH
LA



Sub_Category
Bookcases
Chairs
Labels

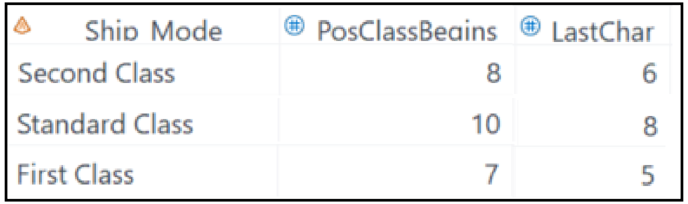


SubCatsAndCodes
AC-Accessories
AP-Appliances
AR-Art
BI-Binders
BO-Bookcases
CH-Chairs
CO-Copiers

FIND(*string, substring, startposition, modifiers***)**

```
PosClassBegins=find(ship_mode, 'Class');
```

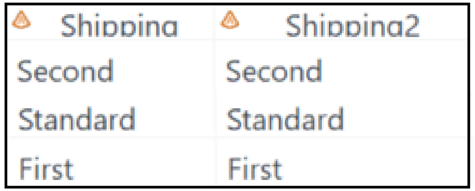
```
LastChar=PositionClassBegins-2;
```



Ship Mode	PosClassBegins	LastChar
Second Class	8	6
Standard Class	10	8
First Class	7	5

SUBSTR(*string, position, length***)**

```
Shipping=substr(ship_mode, 1, LastChar);
```



Shipping	Shipping2
Second	Second
Standard	Standard
First	First

```
Shipping2=substr(ship_mode, 1, find(ship_mode, 'Class')-2);
```

Replacing IF-THEN Conditional Processing

Manipulating, Creating, and Shifting Dates

Converting Column Types

Manipulating Character Values

Eliminating Case Sensitivity on the WHERE Statement

Bonus:

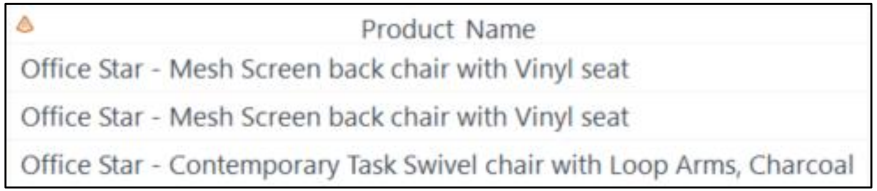
Using Patterns to Manipulate Data

Modifying Character Column Length Using a Macro

Eliminating Case Sensitivity on the WHERE Statement

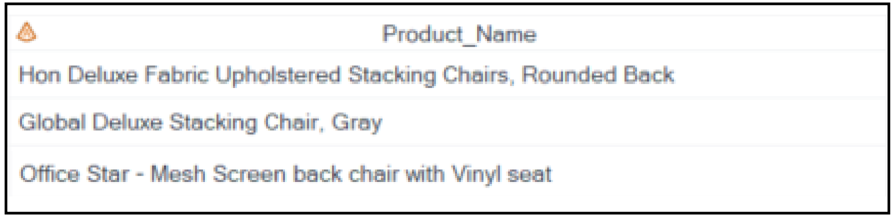
UPCASE(*argument*)

```
where Product_Name like '%chair%';
```



Product_Name
Office Star - Mesh Screen back chair with Vinyl seat
Office Star - Mesh Screen back chair with Vinyl seat
Office Star - Contemporary Task Swivel chair with Loop Arms, Charcoal

```
where upcase(Product_Name) like '%CHAIR%';
```



Product_Name
Hon Deluxe Fabric Upholstered Stacking Chairs, Rounded Back
Global Deluxe Stacking Chair, Gray
Office Star - Mesh Screen back chair with Vinyl seat

Replacing IF-THEN Conditional Processing

Manipulating, Creating, and Shifting Dates

Converting Column Types

Manipulating Character Values

Eliminating Case Sensitivity on the WHERE Statement

Bonus:

Using Patterns to Manipulate Data

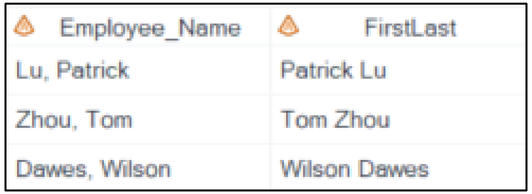
Modifying Character Column Length Using a Macro

Bonus:
Using Patterns to Manipulate Data

Pearl Regular Expressions (PRX)

Pattern matching tools used to **search**, **extract**, or **modify text** based on **patterns** rather than exact characters

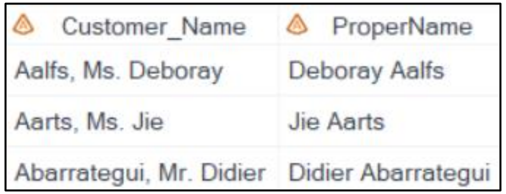
Function	What It Does	Typical Use
PRXPARSE	Compiles a regex pattern	Define reusable pattern
PRXMATCH	Finds first match position	Check if pattern exists
PRXPOSN	Retrieves captured text	Extract parts of a match
PRXNEXT	Finds next match in a loop	Iterate through multiple matches
PRXSUBSTR	Returns matched substring	Get match directly
PRXCHANGE	Substitutes text using regex	Replace or reorder text
PRXDEBUG	Turns on debugging info	Troubleshoot regex logic



Employee_Name	FirstLast
Lu, Patrick	Patrick Lu
Zhou, Tom	Tom Zhou
Dawes, Wilson	Wilson Dawes

```
FirstLast=prxchange('s/(\w+), (\w+)/$2 $1/', -1, employee_name);
```

Syntax	**Description**
s/…/…/	Substitution operator. “Find this pattern and replace it with something else.”
(\w+)	Capture group: captures one or more word characters (letters, numbers, or underscores). This first capture group represents the **last name**.
,	Matches a literal comma followed by a space.
(\w+)	Capture group: captures another word — this one is the **first name**.
$2 $1	The replacement text: inserts the **second capture group** (first name) followed by a space and the **first capture group** (last name).
-1	The second argument. Means replace all matches in the string- often added as best practice.
Employee_name	Column containing the names.



Customer_Name	ProperName
Aalfs, Ms. Deboray	Deboray Aalfs
Aarts, Ms. Jie	Jie Aarts
Abarrategui, Mr. Didier	Didier Abarrategui

```
ProperName=prxchange('s/(\w+), (?:\w+\.\s)?(\w+)/$2 $1/', -1, customer_name);
```

Syntax	Description
s/…/…/	Substitution operator. “Find this pattern and replace it with something else.”
(\w+)	Capture group: captures one or more word characters (letters, numbers, or underscores). This first capture group represents the **last name**.
,	Matches a literal comma followed by a space.
(?: …)	Specifies a non-capturing group.
(?:\w+\.\s)?	Looks for one or more word characters followed by a period and space. The final question mark specifies the entire group is optional.
(\w+)	Capture group: captures another word — this one is the **first name**.
$2 $1	The replacement text: inserts the **second capture group** (first name) followed by a space and the **first capture group** (last name).

Replacing IF-THEN Conditional Processing

Manipulating, Creating, and Shifting Dates

Converting Column Types

Manipulating Character Values

Eliminating Case Sensitivity on the WHERE Statement

Bonus:

Using Patterns to Manipulate Data

Modifying Character Column Length Using a Macro

Bonus:
Modifying Character Column Length Using a Macro

MAX(LENGTH(*argument*)

Returns the length of the longest value in a character column

Creates a data set of all character variables in a specified data set along with their defined length and the length of the longest value in the variable

%charCheck

Updates character columns in the specified data set if the defined length is larger than the length of the longest value in the variable

%charResize