\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BCS ATTESTATION ANALYSIS

Brian Kolovich - 6-20-2018

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*;

libname bcs '/cbna/cbnadm/bk31963';

options mlogic mprint symbolgen;

**proc** **printto** log='/cbna/cbnadm/bk31963/BCS-Log.txt' new;

\*%include "/cbna/cbnadm/Macros.sas";\*/

\* Convert all MS and MG nodes to character format ;

**%macro** ***convert***;

data bcs.GDW2;

set bcs.GDW\_201712\_mod ; /\*bcs.GDW; \*/

%do i=**1** %to **13** ;

%let MS\_List = L01\*L02\*L03\*L04\*L05\*L06\*L07\*L08\*L09\*L10\*L11\*L12\*L13\*L14\*L15\*L16\*L17 ;

%let iter1 = %scan(&MS\_List,&i,\*);

MS\_Node2\_&iter1 = compress(put(MS\_Node\_&iter1 ,**8.**));

rename MS\_Node2\_&iter1=MS\_Node\_&iter1 ;

%end;

%do i=**1** %to **3** ;

%let MG\_List = L01\*L02\*L03;

%let iter2 = %scan(&MG\_List,&i,\*);

MG\_Node2\_&iter2 = compress(put(MG\_Node\_&iter2 ,**8.**));

rename MG\_Node2\_&iter2=MG\_Node\_&iter2 ;

%end;

%do i=**1** %to **9** ;

%let Dir\_Mgr\_List = L12\*L13\*L14\*L15\*L16\*L17\*L18\*L19\*L20;

%let iter3 = %scan(&Dir\_Mgr\_List,&i,\*);

DIRMGR2\_&iter3 = input(DIRMGR\_&iter3 ,**20.**);

rename DIRMGR2\_&iter3=DIRMGR\_&iter3 ;

%end;

/\* COMPRESS DIRECT MANAGER NAMES TO FORM ONE STRING W/O BREAKS \*/

%do i=**1** %to **20** ;

%let DM\_List = L01\*L02\*L03\*L04\*L05\*L06\*L07\*L08\*L09\*L10\*L11\*L12\*L13\*L14\*L15\*L16\*L17\*L18\*L19\*L20;

%let iter4 = %scan(&DM\_List,&i,\*);

DIRMGR\_&iter4.\_NAME = compress(tranwrd(DIRMGR\_&iter4.\_NAME,',',' '));

%end;

rename GOC=GOC\_Code;

drop MS\_NODE\_L01 MS\_NODE\_L02 MS\_NODE\_L03 MS\_NODE\_L04 MS\_NODE\_L05 MS\_NODE\_L06 MS\_NODE\_L07 MS\_NODE\_L08 MS\_NODE\_L09

MS\_NODE\_L10 MS\_NODE\_L11 MS\_NODE\_L12 MS\_NODE\_L13 MG\_NODE\_L01 MG\_NODE\_L02 MG\_NODE\_L03 DIRMGR\_L12 DIRMGR\_L13 DIRMGR\_L14

DIRMGR\_L15 DIRMGR\_L16 DIRMGR\_L17 DIRMGR\_L18 DIRMGR\_L19 DIRMGR\_L20;

run;

**%mend**;

%***convert***;

\*Remove digits from non-code text to ensure codes do not get modified;

**data** bcs.supervisory2;

set bcs.supervisory;

mgd\_seg\_geo\_temp1 = tranwrd(mgd\_seg\_geo,'Delta 1','Delta');

mgd\_seg\_geo\_temp2 = tranwrd(mgd\_seg\_geo\_temp1,'PFDelta1','PFDelta');

mgd\_seg\_geo\_temp3 = tranwrd(mgd\_seg\_geo\_temp2,'G10 FX','GFX');

mgd\_seg\_geo\_adj = tranwrd(mgd\_seg\_geo\_temp3,'G10 Markets','GMarkets');

drop mgd\_seg\_geo\_temp1 mgd\_seg\_geo\_temp2 mgd\_seg\_geo\_temp3;

**run**;

\* Separate Inclusions from Exclusions ;

**data** bcs.supervisory\_inc\_exc;

set bcs.supervisory2;

\* Delete n/a from dataset ;

if substr(mgd\_seg\_geo\_adj,**1**,**3**) = 'n/a' then delete;

\* MANAGED SEGMENT, WORK CITY, SOEID, GOC AND DIRECT MANAGER FIELDS ARE COMMINGLED ;

\* Extract inclusion portion of the string. ;

MS\_WC\_ID\_DM\_Inclusion\_Prep = scan(mgd\_seg\_geo\_adj, **1**, ';');

\* Pull out the inclusion codes. The period delimeter in the 'exc.' portion will remove exclusions ;

\* Temp variable created as 'exclude' text needs to be removed ;

if find(MS\_WC\_ID\_DM\_Inclusion\_Prep,'.') then do;

MS\_WC\_ID\_DM\_temp = strip(scan(MS\_WC\_ID\_DM\_Inclusion\_Prep, **1**, '.'));

MS\_WC\_ID\_DM\_Inclusion = tranwrd(MS\_WC\_ID\_DM\_temp,"exc","");

end;

else do;

MS\_WC\_ID\_DM\_Inclusion = MS\_WC\_ID\_DM\_Inclusion\_Prep;

end;

\* Extract exclusions ;

if find(mgd\_seg\_geo\_adj,'.') then do;

MS\_WC\_ID\_DM\_Exclusion\_temp = strip(scan(mgd\_seg\_geo\_adj, -**1**, '.'));

MS\_WC\_ID\_DM\_Exclusion = strip(scan(MS\_WC\_ID\_DM\_Exclusion\_temp, **1**, ';'));

end;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF MS, WC, SOEID, GOC AND DM EXTRACTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ;

\* MANAGED GEOGRAPHY IS EXTRACTED SEPARATELY ;

\* Extract exclusion portion of the string ;

\* Pull out the MG portion of the string. Everything after semicolon ;

MG\_All = strip(scan(mgd\_seg\_geo\_adj, -**1**, ';'));

\* Pull out the MG inclusions from MG All. ;

\* Temp variable created as 'exclude' text needs to be removed ;

if find(MG\_All,'.') then do;

MG\_Inclusion\_temp = strip(scan(MG\_All, **1**, '.'));

MG\_Inclusion = tranwrd(MG\_Inclusion\_temp,"exc","");

end;

else do;

MG\_Inclusion = MG\_All ;

end;

if find(MG\_Inclusion,'1 - Total Citi') then do;

MG\_Inclusion = '1000 - Total Citi [L1]' ;

end;

\* Extract exclusions ;

if find(MG\_All,'.') then do;

MG\_Exclusion = strip(scan(MG\_All, -**1**, '.'));

end;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END OF MANAGED GEOGRAPHY EXTRACTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ;

drop MS\_WC\_ID\_DM\_Inclusion\_Prep MS\_WC\_ID\_DM\_temp MG\_Inclusion\_temp MG\_All MS\_WC\_ID\_DM\_Exclusion\_temp ;

**run**;

\* Remove nodes for number extraction/compression ;

**%macro** ***remove\_nodes***;

data bcs.supervisory\_rmv\_nodes;

set bcs.supervisory\_inc\_exc;

%do i=**1** %to **17** ;

%let Node\_List = '[L1]'\*'[L2]'\*'[L3]'\*'[L4]'\*'[L5]'\*'[L6]'\*'[L7]'\*'[L8]'\*'[L9]'\*'[L10]'\*'[L11]'\*'[L12]'\*'[L13]'

\*'[L14]'\*'[L15]'\*'[L16]'\*'[L17]';

%let iter = %scan(&Node\_List,&i,\*);

MG\_Exclusion = tranwrd(MG\_Exclusion,&iter,"");

MG\_Inclusion = tranwrd(MG\_Inclusion,&iter,"");

MS\_WC\_ID\_DM\_Exclusion = tranwrd(MS\_WC\_ID\_DM\_Exclusion,&iter,"");

MS\_WC\_ID\_DM\_Inclusion = tranwrd(MS\_WC\_ID\_DM\_Inclusion,&iter,"");

%end;

run;

**%mend**;

%***remove\_nodes***;

\* Replace 'and' with & delimiter and compress ;

**data** bcs.bcs\_tokenize\_prep;

set bcs.supervisory\_rmv\_nodes;

\*Replace and with & for delimiter identifer ;

MS\_WC\_ID\_DM\_Inclusion\_temp = tranwrd(MS\_WC\_ID\_DM\_Inclusion,'and','&');

MS\_WC\_ID\_DM\_Inclusion\_Final = compress(MS\_WC\_ID\_DM\_Inclusion\_temp);

MS\_WC\_ID\_DM\_Exclusion\_temp = tranwrd(MS\_WC\_ID\_DM\_Exclusion,'and','&');

MS\_WC\_ID\_DM\_Exclusion\_Final = compress(MS\_WC\_ID\_DM\_Exclusion\_temp);

MG\_Inclusion\_temp = tranwrd(MG\_Inclusion,'and','&');

MG\_Inclusion\_Final = compress(MG\_Inclusion\_temp);

MG\_Exclusion\_temp = tranwrd(MG\_Exclusion,'and','&');

MG\_Exclusion\_Final = compress(MG\_Exclusion\_temp);

drop MS\_WC\_ID\_DM\_Exclusion\_temp MG\_Inclusion\_temp MS\_WC\_ID\_DM\_Inclusion\_temp

MG\_Exclusion\_temp MS\_WC\_ID\_DM\_Inclusion MS\_WC\_ID\_DM\_Exclusion MG\_Inclusion MG\_Exclusion ;

**run**;

\* Tokenize text into multiple columns based on & delimeter ;

**data** bcs.bcs\_tokenize2;

set bcs.bcs\_tokenize\_prep;

\*MS ;

length MS\_Inc\_1-MS\_Inc\_10 $**100**;

length MS\_Exc\_1-MS\_Exc\_10 $**100**;

array MS\_Inc\_(**10**) $;

array MS\_Exc\_(**10**) $;

\* MG ;

length MG\_Inc\_1-MG\_Inc\_10 $**100**;

length MG\_Exc\_1-MG\_Exc\_10 $**100**;

array MG\_Inc\_(**10**) $;

array MG\_Exc\_(**10**) $;

\* GOC ;

length GOC\_Tmp\_Inc\_1-GOC\_Tmp\_Inc\_10 $**100**;

length GOC\_Tmp\_Exc\_1-GOC\_Tmp\_Exc\_10 $**100**;

length GOC\_Inc\_1-GOC\_Inc\_10 $**100**;

length GOC\_Exc\_1-GOC\_Exc\_10 $**100**;

array GOC\_Tmp\_Inc\_(**10**) $;

array GOC\_Tmp\_Exc\_(**10**) $;

array GOC\_Inc\_(**10**) $;

array GOC\_Exc\_(**10**) $;

\* Work City ;

length WC\_Tmp\_Inc\_1-WC\_Tmp\_Inc\_10 $**100**;

length WC\_Tmp\_Exc\_1-WC\_Tmp\_Exc\_10 $**100**;

length WC\_Inc\_1-WC\_Inc\_10 $**100**;

length WC\_Exc\_1-WC\_Exc\_10 $**100**;

array WC\_Tmp\_Inc\_(**10**) $;

array WC\_Tmp\_Exc\_(**10**) $;

array WC\_Inc\_(**10**) $;

array WC\_Exc\_(**10**) $;

\* Job Function ;

length JF\_Tmp\_Inc\_1-JF\_Tmp\_Inc\_10 $**100**;

length JF\_Tmp\_Exc\_1-JF\_Tmp\_Exc\_10 $**100**;

length JF\_Inc\_1-JF\_Inc\_10 $**100**;

length JF\_Exc\_1-JF\_Exc\_10 $**100**;

array JF\_Tmp\_Inc\_(**10**) $;

array JF\_Tmp\_Exc\_(**10**) $;

array JF\_Inc\_(**10**) $;

array JF\_Exc\_(**10**) $;

\* SOEID ;

length SOEID\_Tmp\_Inc\_1-SOEID\_Tmp\_Inc\_10 $**100**;

length SOEID\_Tmp\_Exc\_1-SOEID\_Tmp\_Exc\_10 $**100**;

length SOEID\_Tmp2\_Inc\_1-SOEID\_Tmp2\_Inc\_10 $**100**;

length SOEID\_Tmp2\_Exc\_1-SOEID\_Tmp2\_Exc\_10 $**100**;

length SOEID\_Inc\_1-SOEID\_Inc\_10 $**100**;

length SOEID\_Exc\_1-SOEID\_Exc\_10 $**100**;

array SOEID\_Tmp\_Inc\_(**10**) $;

array SOEID\_Tmp\_Exc\_(**10**) $;

array SOEID\_Tmp2\_Inc\_(**10**) $;

array SOEID\_Tmp2\_Exc\_(**10**) $;

array SOEID\_Inc\_(**10**) $;

array SOEID\_Exc\_(**10**) $;

\* Direct Manager ;

length DM\_Tmp\_Inc\_1-DM\_Tmp\_Inc\_10 $**100**;

length DM\_Tmp\_Exc\_1-DM\_Tmp\_Exc\_10 $**100**;

length DM\_Inc\_1-DM\_Inc\_10 $**100**;

length DM\_Exc\_1-DM\_Exc\_10 $**100**;

array DM\_Tmp\_Inc\_(**10**) $;

array DM\_Tmp\_Exc\_(**10**) $;

array DM\_Inc\_(**10**) $;

array DM\_Exc\_(**10**) $;

\* MS Inclusions ;

do \_i = **1** to dim(MS\_Inc\_);

MS\_Inc\_[\_i] = compress(scan(MS\_WC\_ID\_DM\_Inclusion\_Final,\_i,'&'),,'kd'); \* KD modifier removes non-digits ;

end;

\* MS Exclusions ;

do \_i = **1** to dim(MS\_Exc\_);

MS\_Exc\_[\_i] = compress(scan(MS\_WC\_ID\_DM\_Exclusion\_Final,\_i,'&'),,'kd'); \* KD modifier removes non-digits ;

end;

\* MG Inclusions - Need additional criteria for numeric vs. alpha(country) codes ;

do \_i = **1** to dim(MG\_Inc\_);

if substr(MG\_Inclusion\_Final,**1**,**1**) >=**0** then

MG\_Inc\_[\_i] = compress(scan(MG\_Inclusion\_Final,\_i,'&'),,'kd'); \* KD modifier removes non-digits ;

else if substr(MG\_Inclusion\_Final,**1**,**1**) not=**0** then MG\_Inc\_[\_i] = substr(scan(MG\_Inclusion\_Final,\_i,'&'),**1**,**2**);

end;

\* MG Exclusions - Need additional criteria for numeric vs. alpha(country) codes ;

do \_i = **1** to dim(MG\_Exc\_);

if substr(MG\_Exclusion\_Final,**1**,**1**) >=**0** then

MG\_Exc\_[\_i] = compress(scan(MG\_Exclusion\_Final,\_i,'&'),,'kd'); \* KD modifier removes non-digits ;

else if substr(MG\_Exclusion\_Final,**1**,**1**) not=**0** then MG\_Exc\_[\_i] = substr(scan(MG\_Exclusion\_Final,\_i,'&'),**1**,**2**);

end;

\* GOC Inclusions ;

do \_i = **1** to dim(GOC\_Tmp\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'GOC') then

GOC\_Tmp\_Inc\_[\_i] = strip(scan(MS\_WC\_ID\_DM\_Inclusion\_Final,\_i,'&'));

end;

do \_i = **1** to dim(GOC\_Inc\_);

GOC\_Inc\_[\_i] = tranwrd(strip(substr(GOC\_Tmp\_Inc\_[\_i],**4**,**10**)),'-','');

end;

\* GOC Exclusions ;

do \_i = **1** to dim(GOC\_Tmp\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'GOC') then

GOC\_Tmp\_Exc\_[\_i] = strip(scan(MS\_WC\_ID\_DM\_Exclusion\_Final,\_i,'&'));

end;

do \_i = **1** to dim(GOC\_Exc\_);

GOC\_Exc\_[\_i] = tranwrd(strip(substr(GOC\_Tmp\_Exc\_[\_i],**4**,**10**)),'-','');

end;

\* Work City Inclusions ;

do \_i = **1** to dim(WC\_Tmp\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'City') then

WC\_Tmp\_Inc\_[\_i] = scan(MS\_WC\_ID\_DM\_Inclusion\_Final,\_i,'&');

end;

do \_i = **1** to dim(WC\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'City') then

WC\_Inc\_[\_i] = tranwrd(strip(substr(WC\_Tmp\_Inc\_[\_i],**10**,**20**)),"'",'');

end;

\* Work City Exclusions ;

do \_i = **1** to dim(WC\_Tmp\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'City') then

WC\_Tmp\_Exc\_[\_i] = scan(MS\_WC\_ID\_DM\_Exclusion\_Final,\_i,'&');

end;

do \_i = **1** to dim(WC\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'City') then

WC\_Exc\_[\_i] = tranwrd(strip(substr(WC\_Tmp\_Exc\_[\_i],**10**,**20**)),"'",'');

end;

\* Job Function Inclusions ;

do \_i = **1** to dim(JF\_Tmp\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'JobFunctionDesc') then

JF\_Tmp\_Inc\_[\_i] = scan(MS\_WC\_ID\_DM\_Inclusion\_Final,\_i,'&');

end;

do \_i = **1** to dim(JF\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'JobFunctionDesc') then

JF\_Inc\_[\_i] = tranwrd(strip(substr(JF\_Tmp\_Inc\_[\_i],**17**,**20**)),"'",'');

end;

\* Job Function Exclusions ;

do \_i = **1** to dim(JF\_Tmp\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'JobFunctionDesc') then

JF\_Tmp\_Exc\_[\_i] = scan(MS\_WC\_ID\_DM\_Exclusion\_Final,\_i,'&');

end;

do \_i = **1** to dim(JF\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'JobFunctionDesc') then

JF\_Exc\_[\_i] = tranwrd(strip(substr(JF\_Tmp\_Exc\_[\_i],**17**,**20**)),"'",'');

end;

\* SOEID Inclusions ;

do \_i = **1** to dim(SOEID\_Tmp\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'SOEID') then

SOEID\_Tmp\_Inc\_[\_i] = scan(MS\_WC\_ID\_DM\_Inclusion\_Final,\_i,'&');

end;

do \_i = **1** to dim(SOEID\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'SOEID') then

SOEID\_Tmp2\_Inc\_[\_i] = index(SOEID\_Tmp\_Inc\_[\_i],'SOEID') + **5**;

end;

do \_i = **1** to dim(SOEID\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'SOEID') then

SOEID\_Inc\_[\_i] = strip(substr(SOEID\_Tmp\_Inc\_[\_i],SOEID\_Tmp2\_Inc\_[\_i],**7**));

end;

\* SOEID Exclusions ;

do \_i = **1** to dim(SOEID\_Tmp\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'SOEID') then

SOEID\_Tmp\_Exc\_[\_i] = scan(MS\_WC\_ID\_DM\_Exclusion\_Final,\_i,'&');

end;

do \_i = **1** to dim(SOEID\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'SOEID') then

SOEID\_Tmp2\_Exc\_[\_i] = index(SOEID\_Tmp\_Exc\_[\_i],'SOEID') + **5**;

end;

do \_i = **1** to dim(SOEID\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'SOEID') then

SOEID\_Exc\_[\_i] = strip(substr(SOEID\_Tmp\_Exc\_[\_i],SOEID\_Tmp2\_Exc\_[\_i],**7**));

end;

\* Direct Manager Inclusions ;

do \_i = **1** to dim(DM\_Tmp\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'DM') then

DM\_Tmp\_Inc\_[\_i] = scan(MS\_WC\_ID\_DM\_Inclusion\_Final,\_i,'&');

end;

do \_i = **1** to dim(DM\_Inc\_);

if find(MS\_WC\_ID\_DM\_Inclusion\_Final,'DM') then

DM\_Inc\_[\_i] = compress(tranwrd(strip(substr(DM\_Tmp\_Inc\_[\_i],**5**,**20**)),"'",''),',');

end;

\* Direct Manager Exclusions ;

do \_i = **1** to dim(DM\_Tmp\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'DM') then

DM\_Tmp\_Exc\_[\_i] = scan(MS\_WC\_ID\_DM\_Exclusion\_Final,\_i,'&');

end;

do \_i = **1** to dim(DM\_Exc\_);

if find(MS\_WC\_ID\_DM\_Exclusion\_Final,'DM') then

DM\_Exc\_[\_i] = tranwrd(strip(substr(DM\_Tmp\_Exc\_[\_i],**5**,**20**)),"'",'');

end;

\* Make MS '1' values blank from DM1 value - everything in GDW is 1 ;

if MS\_Inc\_1 = '1' then MS\_Inc\_1= '';

if MS\_Inc\_2 = '1' then MS\_Inc\_2= '';

if MS\_Inc\_3 = '1' then MS\_Inc\_3= '';

if MS\_Inc\_4 = '1' then MS\_Inc\_4= '';

if MS\_Inc\_5 = '1' then MS\_Inc\_5= '';

if MS\_Inc\_6 = '1' then MS\_Inc\_6= '';

if MS\_Inc\_7 = '1' then MS\_Inc\_7= '';

if MS\_Inc\_8 = '1' then MS\_Inc\_8= '';

if MS\_Inc\_9 = '1' then MS\_Inc\_9= '';

if MS\_Inc\_10 = '1' then MS\_Inc\_10= '';

drop \_i GOC\_Tmp\_Inc\_1-GOC\_Tmp\_Inc\_10 GOC\_Tmp\_Exc\_1-GOC\_Tmp\_Exc\_10 WC\_Tmp\_Inc\_1-WC\_Tmp\_Inc\_10 WC\_Tmp\_Exc\_1-WC\_Tmp\_Exc\_10

JF\_Tmp\_Inc\_1-JF\_Tmp\_Inc\_10 JF\_Tmp\_Exc\_1-JF\_Tmp\_Exc\_10 SOEID\_Tmp\_Inc\_1-SOEID\_Tmp\_Inc\_10 SOEID\_Tmp\_Exc\_1-SOEID\_Tmp\_Exc\_10

SOEID\_Tmp2\_Inc\_1-SOEID\_Tmp2\_Inc\_10 SOEID\_Tmp2\_Exc\_1-SOEID\_Tmp2\_Exc\_10 DM\_Tmp\_Inc\_1-DM\_Tmp\_Inc\_10 DM\_Tmp\_Exc\_1-DM\_Tmp\_Exc\_10 ;

**run**;

\* Initiate Macro to remove empty columns ;

%***findmiss***(bcs\_tokenize2,dropcol);

**data** bcs.bcs\_tokenize\_empty ;

set bcs.bcs\_tokenize2(drop=&dropcol);

**run**;

\* Now Lookup Codes, Names, Cities, etc. in GDW to blank out non-matches ;

**proc** **sql**;

create table bcs.bcs\_remove\_non\_matches as select distinct

A.Control\_Number,A.Supervisor\_Business,A.Supervisor\_Sub\_Business,A.Supervisor\_Region,A.Supervisor\_Country,A.Supervisor\_SOEID,

A.Supervisor\_GEID,A.EMP\_Status,A.Supervisor\_Name,A.Supervisor\_Title,A.Supervisor\_Desks,A.Mgd\_Seg\_Geo\_Headcount,A.Direct\_Reports,

A.All\_Reports\_DM1\_only,A.Mgd\_Seg\_Geo,

/\* Macros \*/

%***MS\_Inc\_GDW\_No\_Match***

%***MS\_Exc\_GDW\_No\_Match***

%***MG\_Inc\_GDW\_No\_Match***

%***MG\_Exc\_GDW\_No\_Match***

%***GOC\_Inc\_GDW\_No\_Match***

%***WC\_Inc\_GDW\_No\_Match***

%***JF\_Inc\_GDW\_No\_Match***

%***SOEID\_Inc\_GDW\_No\_Match***

%***DM\_Inc\_GDW\_No\_Match***

from bcs.bcs\_tokenize\_empty as A, bcs.gdw2 as B

group by A.Control\_Number,A.Supervisor\_Business,A.Supervisor\_Sub\_Business,A.Supervisor\_Region,A.Supervisor\_Country,A.Supervisor\_SOEID,

A.Supervisor\_GEID,A.EMP\_Status,A.Supervisor\_Name,A.Supervisor\_Title,A.Supervisor\_Desks,A.Mgd\_Seg\_Geo\_Headcount,A.Direct\_Reports,

A.All\_Reports\_DM1\_only,A.Mgd\_Seg\_Geo;

**quit**;

\* Replace SOEID with GEID for Direct Manager Lookup ;

**proc** **sql**;

create table bcs.bcs\_match\_geid as select distinct

A.Control\_Number, A.Supervisor\_Business, A.Supervisor\_Sub\_Business, A.Supervisor\_Region,

A.Supervisor\_Country, A.Supervisor\_SOEID, A.Supervisor\_GEID, A.EMP\_Status, A.Supervisor\_Name,

A.Supervisor\_Title, A.Supervisor\_Desks, A.Mgd\_Seg\_Geo\_Headcount, A.Direct\_Reports,

A.All\_Reports\_DM1\_only, A.Mgd\_Seg\_Geo, A.MS\_Inc\_1, A.MS\_Inc\_2, A.MS\_Inc\_3, A.MS\_Exc\_1, A.MS\_Exc\_2,

case when A.MG\_Inc\_1 is missing and not find(Mgd\_Seg\_Geo,';') then '1000' else A.MG\_Inc\_1 end as MG\_Inc\_1,

A.MG\_Inc\_2, A.MG\_Exc\_1, A.GOC\_Inc\_1, A.GOC\_Inc\_2, A.WC\_Inc\_1, A.WC\_Inc\_2, A.JF\_Inc\_1,

A.JF\_Inc\_3, A.SOEID\_Inc\_1, A.SOEID\_Inc\_2, A.DM\_Inc\_1, A.DM\_Inc\_2,

case when A.SOEID\_Inc\_1 not is missing and A.SOEID\_Inc\_1 = B.SOEID then B.GEID end as GEID\_Inc\_1,

case when A.SOEID\_Inc\_2 not is missing and A.SOEID\_Inc\_2 = B.SOEID then B.GEID end as GEID\_Inc\_2

from bcs.bcs\_remove\_non\_matches as A left join bcs.gdw2 as B on (A.Supervisor\_SOEID=B.SOEID);

**quit**;

\* Compile final table ;

**proc** **sql**;

create table bcs.bcs\_final\_1 as select distinct

A.Control\_number,

A.Supervisor\_SOEID,

A.SUPERVISOR\_GEID,

A.Supervisor\_Business,

A.Supervisor\_Sub\_Business,

A.Supervisor\_Region,

A.Supervisor\_Country,

A.Supervisor\_Desks,

A.Mgd\_Seg\_Geo,

B.SOEID as MGD\_SOEID,

B.GEID as MGD\_GEID

from bcs.bcs\_match\_geid as A, bcs.gdw2 as B

where

%***Filter***

order by A.Control\_number;

**quit**;

**data** bcs.bcs\_final\_1\_rmv;

set bcs.bcs\_final\_1;

where Control\_number not = **77**;

**run**;

**proc** **sql**;

create table bcs.bcs\_final\_2 as select distinct

A.Control\_number,

A.Supervisor\_SOEID,

A.SUPERVISOR\_GEID,

A.Supervisor\_Business,

A.Supervisor\_Sub\_Business,

A.Supervisor\_Region,

A.Supervisor\_Country,

A.Supervisor\_Desks,

A.Mgd\_Seg\_Geo,

B.SOEID as MGD\_SOEID,

B.GEID as MGD\_GEID

from bcs.bcs\_match\_geid as A, bcs.gdw2 as B

/\* Outlier Manual Override for Control\_Number 77 - one time only \*/

where

(case when control\_number = **77** and B.MS\_NODE\_L07 in ('1355','1383') and B.MG\_NODE\_L02 = '1129' then **1**

when control\_number = **77** and B.MS\_NODE\_L07 = '24683' and B.MS\_NODE\_L08 not= '17641' and B.MG\_NODE\_L02 = '1129' then **1**

when control\_number = **77** and B.MS\_NODE\_L07 = '24683' and B.MS\_NODE\_L08 not= '17641' and B.MG\_NODE\_L03 = '1018' then **1** end) = **1**

order by A.Control\_number;

**quit**;

**data** bcs.bcs\_final;

set bcs.bcs\_final\_1\_rmv bcs.bcs\_final\_2;

**run**;

**proc** **sort** data=bcs.bcs\_final; by control\_number;

/\*%include "/cbna/cbnadm/Control Testing.sas";\*/

**proc** **printto** log=log;

**run**;