/\* Project Title: IHA Target File 3 2023

Requesting Department: Internal

Requestor: XXXXXXXXXX

Origination Date: 12/6/2023

Requested Completion Date: 12/11/2023

Date received: 12/6/2023

Assigned Priority: High

Assigned Delivery Date: 12/11/2023

RDA\_Project\_number: XXXXXXXXXX

Assigned Analyst: Meagan Windler

Assigned Days: 3

Support Analyst:

Perc\_Support: 0

Assigned Completion Date: 12/11/2023

Remarks:

Use member list from Warren Clinic. Confirm members are still active (remove any that are not).

Create Emcara chase list and gaps file.

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Define Macro Variables\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\*month we are running for;

%Let DataMonth=202310; /\* (Format:yyyymm) \*/

/\* Clearing Work Library \*/

proc datasets nolist nodetails lib=Work kill; quit;

/\* Setting Valid Variable Names to Change Blanks to Underscore when Importing \*/

options validvarname=V7;

/\* Searching for custom formats \*/

options fmtsearch=(work formats) fmterr;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*User Variables\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Measurement Year (Current Calendar Year)\*/

%Let MeasurementYear=2023;

/\* Member List \*/

\*Using a member list generated by Warren Clinic;

%Let MemberFile=November 2023 Community Care Patients not seen;

/\* Change Extract \*/

\*This is going to be the end of the previous month;

%Let LimitationPartC="30NOV2023"d;

/\* Part D Date - Check that tables in location are updated, should be in the

last couple of weeks\*/

\*check the "final\_rates\_diab table located here:

\\XXXXXXXXXX\Production\W\_Stars Reporting - Medication Adherence\Data

for the most up to date linked end date, that will be the value you use here;

%Let LimitationPartD="04DEC2023"d;

/\* Cut Point File for Measure Names \*/

\*This is an excel sheet at location \\XXXXXXXXXX\HDA\Production\M\_Medicare HEDIS Rates\Source Data

and it should be the most up to date version. Looks like it changes once a year;

%Let CutPointFile=STARS 2024 Cut Points;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Setup\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Setting standard date variables for today \*/

%Let rundate=%sysfunc(today(),mmddyy10.); /\* Format: mm/dd/yyyy \*/

%Let logdate=%sysfunc(today(),yymmddn8.); /\* Format: yyyymmdd \*/

%Let analysisdate=%sysfunc(inputn(&rundate,mmddyy10.),date9.); /\* Format: ddmmmyyyy \*/

/\* Defining User and Display Macro Variables \*/

%Let analyst=MW;

%Let rda=XXXXXXXXXX;

/\* Server Address \*/

%Let server=XXXXXXXXXX;

/\* Dynamically establishing path based on location of sas file being run \*/

data \_null\_;

/\* If you open the program via the Server \*/

if substr(compress(&\_SASPROGRAMFILE,"'"),1,1)='N' then

call symput("File\_Path",cat("&server\DAR\",substr(compress(&\_SASPROGRAMFILE,"'"),4)));

/\* If you open the program via the File Explorer \*/

if substr(compress(&\_SASPROGRAMFILE,"'"),1,1)='G' then

call symput("File\_Path",cat("&server\",substr(compress(&\_SASPROGRAMFILE,"'"),4)));

/\* If the SAS Program file macro is correct... \*/

if substr(compress(&\_SASPROGRAMFILE,"'"),1,11)='\\twr-sas-1' then

call symput("File\_Path",&\_SASPROGRAMFILE);

run;

%Let File\_Name=%scan(&File\_Path,-1,'\');

%Let Code\_Folder=%scan(&File\_Path,-2,'\');

%Let Code\_Path=%substr(%Sysfunc(Tranwrd(&File\_Path,&File\_Name,)),1,%length(%Sysfunc(Tranwrd(&File\_Path,&File\_Name,)))-1);

%Let Path=%substr(%Sysfunc(Tranwrd(&Code\_Path,&Code\_Folder,)),1,%length(%Sysfunc(Tranwrd(&Code\_Path,&Code\_Folder,)))-1);

%Put &Path;

/\* Locations \*/

%let ExtractLocation=XXXXXXXXXX\DAR\HDA\Regulatory Reporting\HEDIS\Compliance Extracts\Standard;

%Let change\_sasdata=XXXXXXXXXX\DAR\HDA\Regulatory Reporting\HEDIS\HEDIS MY&MeasurementYear\Change Healthcare;

/\* Libraries \*/

libname npi "XXXXXXXXXX\Regulatory Reporting\NPI DATA\SAS DATA";

libname partd "&server\XXXXXXXXXX\W\_Stars Reporting - Medication Adherence\Data"; /\* Part D Data, Managed by Michael\*/

libname hedis "&change\_sasdata\SAS DATA"; /\* Member Crosswalk & Claims \*/

libname extract "\\XXXXXXXXXXRegulatory Reporting\HEDIS\Compliance Extracts\SAS Tables"; \*Hedics claims Extract;

/\* Loading format for Change Crosswalk \*/

proc format cntlin=hedis.xwalk; run;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Loading Members\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Loading members to report on \*/

proc import datafile="&Path\Data\&MemberFile..xlsx"

dbms=excel

out=PopHealth\_Members

replace;

sheet="CCOK Patients";

run;

/\* Applying Crosswalk \*/

data PopHealth\_Members;

set PopHealth\_Members;

MemberAddno=substr(Member\_ID,1,9);

MemberXwalk=put(Member\_ID,$xwalk.);

run;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Medicare Members\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Looking for Medicare Members active during the year \*/

data member\_spans;

set memb.membspan;

where input(ymdeff,? yymmdd8.)<=today()

and input(ymdend,? yymmdd8.)>="01Jan&MeasurementYear"d

and void=" "

and prog\_ in ("XXXXXXXXXX","XXXXXXXXXX","XXXXXXXXXX");

Member9=substr(Member\_,1,9);

Original\_Member=Member\_;

Member\_=put(Member\_,$xwalk.);

Prov\_Number=scan(aff\_,1,"");

/\* Removing 1 Day Spans (Usually establsihing Medicare Member) \*/

date\_start=input(ymdeff,?? yymmdd8.);

date\_end=input(ymdend,?? yymmdd8.);

if date\_start~=date\_end;

format date\_: mmddyy10.;

run;

/\* Sorting to find most recent span \*/

proc sort data=member\_spans; by Member\_ date\_end; run;

/\* Finding latest span \*/

data member\_spans\_recent;

set member\_spans;

by Member\_ date\_end;

if last.Member\_=1;

/\* Need date to use AttachAff Info Macro \*/

Today=today();

/\* Flagging Active Records \*/

if date\_start<=today<=date\_end then Flag\_Active=1;

strtdate=min(date\_end,Today);

format today strtdate mmddyy10.;

run;

/\* Flagging All Active Members \*/

%Flagem(member\_spans\_recent,Original\_Member,$All\_OriginalMembers);

%Flagem(member\_spans\_recent,member\_,$All\_Members);

%Flagem(member\_spans\_recent,Member9,$Find\_Members\_Addno);

/\* Attaching IRS Info \*/

%AttachAffInfo(member\_spans\_recent,Spec1 IRS\_);

/\* Creating formats using Formatme stored macro \*/

%Formatme(work,member\_spans\_recent\_withinfo,Member\_,Prog\_,$Member\_to\_Program);

%Formatme(work,member\_spans\_recent\_withinfo,Member\_,Aff\_,$Member\_to\_Affiliation);

%Formatme(work,member\_spans\_recent\_withinfo,Member\_,Prov\_Number,$Member\_to\_PCP);

%Formatme(work,member\_spans\_recent\_withinfo,Member\_,Spec1,$Member\_to\_Specialty);

%Formatme(work,member\_spans\_recent\_withinfo,Member\_,IRS\_,$Member\_to\_IRS);

%Formatme(work,member\_spans\_recent\_withinfo,Member\_,Date\_End,$Member\_to\_Term);

%Formatme(work,member\_spans\_recent\_withinfo,Member\_,Flag\_Active,$ActiveMembers);

/\* Flagging Provider Number for NPI Search \*/

%Flagem(member\_spans\_recent\_withinfo,Prov\_Number,$Find\_Provider);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Member DOB & MBI \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Finding DOB, Name & Gender, & MBI \*/

data membership\_details;

set memb.memb;

where put(Member\_,$All\_OriginalMembers.)="\*";

MemberID=put(member\_,$xwalk.);

DOB=input(ymdbirth,yymmdd8.);

format DOB mmddyy10.;

run;

/\* Creating formats \*/

%Formatme(work,membership\_details,Member\_,Name\_First,$Member\_FirstName);

%Formatme(work,membership\_details,Member\_,Name\_Last,$Member\_LastName);

%Formatme(work,membership\_details,Member\_,DOB,$Member\_DOB);

%Formatme(work,membership\_details,Member\_,Sex,$Member\_Sex);

%Formatme(work,membership\_details,Member\_,Alt\_Key,$MBI);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Launguage \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Going to initially use the memb table for language (but this is not populated

very well so will use the data incorporated by HEDIS as well) \*/

data hedis\_langauge;

set hedis.membership;

where put(Mem\_Unique\_ID,$All\_Members.)="\*";

/\* Key for what each value represents \*/

/\* Update mapping of language codes 01312023 \*/

if Spoken\_Language="E" then ISOLanguage="EN"; /\* English \*/

else if Spoken\_Language="S" then ISOLanguage="ES"; /\* Spanish \*/

else if Spoken\_Language="C" then ISOLanguage="ZH"; /\* Chinese \*/

else if Spoken\_Language="N" then ISOLanguage="EN"; /\* English \*/

else if Spoken\_Language="D" then ISOLanguage="EN"; /\* English \*/

else if Spoken\_Language="U" then ISOLanguage="EN"; /\* English \*/

if ISOLanguage~="" then ISOLanguage="EN";

keep Mem\_Unique\_ID Spoken\_Language ISOLanguage;

run;

/\* Creating format for language \*/

%Formatme(work,hedis\_langauge,Mem\_Unique\_ID,ISOLanguage,$Spoken\_Language);

/\* Creating final table \*/

data membership\_language;

set memb.memb;

where put(Member\_,$All\_OriginalMembers.)="\*";

MemberID=put(member\_,$xwalk.);

/\* Need to convert to ISO Codes \*/

if language~="" then do;

if language="SP" then language="ES";

end;

/\* Otherwise use HEDIS \*/

if language="" then do;

language=put(MemberID,$Spoken\_Language.);

end;

run;

/\* Observing Frequencies \*/

proc freq data=membership\_language noprint;

tables language/out=check\_language;

run;

/\* Creating format for language \*/

%Formatme(work,membership\_language,Member\_,language,$Language);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Finding PCP Address \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Finding address and phone number for each PCP \*/

data pcp\_address;

set memb.address;

where put(addno,$Find\_Provider.)="\*"

and input(ymdend,? yymmdd8.)>today()

/\* There may be an address listed for the future so I need to

add a restriction on effective date aswell \*/

and input(ymdeff,? yymmdd8.)<=today()

and whotype="P"

/\* Just need Default Address for Provider \*/

and AddType in: ("D")

and void=" ";

run;

/\* Sorting to find most recent address \*/

proc sort data=pcp\_address; BY addno addtype YmdEnd YmdEff; run;

/\* Finding last known address \*/

data pcp\_address\_last (keep=addno Address address1 address2 city state zip

final\_myphone email phone3);

length final\_myphone $12.;

set pcp\_address;

by addno;

if last.addno;

Address=catx(" ",address1, address2, city, state, zip);

/\* Looking for meaningful phone number \*/

if phone1 not in ("","0000000000") then myphone=phone1;

else if phone2 not in ("","0000000000") then myphone=phone2;

else if phone3 not in ("","0000000000") then myphone=phone3;

else myphone='N/A';

temp\_myphone=compress(myphone,'() -.');

/\* Putting Phone Number in the desired Format \*/

if myphone='N/A' then final\_myphone=" ";

else final\_myphone=compress(catx("",substr(temp\_myphone,1,3),

substr(temp\_myphone,4,3),

substr(temp\_myphone,7,4)));

run;

/\* Creating format for address and phone number \*/

%formatme(work,pcp\_address\_last,addno,address1,$PCP\_to\_Address\_One);

%formatme(work,pcp\_address\_last,addno,address2,$PCP\_to\_Address\_Two);

%formatme(work,pcp\_address\_last,addno,city,$PCP\_to\_City);

%formatme(work,pcp\_address\_last,addno,state,$PCP\_to\_State);

%formatme(work,pcp\_address\_last,addno,zip,$PCP\_to\_Zip);

%formatme(work,pcp\_address\_last,addno,final\_myphone,$PCP\_to\_Phone);

/\* Looking at the structure of the provider addresses it looks to be

that Phone 3 always contains the Fax Number indicated by the info

being "FX" \*/

/\* Format for Provider Fax \*/

%formatme(work,pcp\_address\_last,addno,phone3,$Provider\_Fax);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Finding NPI for PCP \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Pulling all provider records for NPI \*/

data provider\_npi;

set provider.provider;

where put(prov\_,$Find\_Provider.)="\*";

run;

/\* Creating format for NPI & Credential (if missing from NPI file) \*/

%Formatme(work,provider\_npi,Prov\_,NPI,$Provider\_NPI);

%Formatme(work,provider\_npi,Prov\_,Title,$Provider\_Credential\_Backup);

/\* Flagging NPI's for further information search \*/

%Flagem(provider\_npi,NPI,$Find\_NPI);

/\* Creating format for first and last name \*/

%Formatme(work,provider\_npi,Prov\_,FirstName,$Provider\_FirstName);

%Formatme(work,provider\_npi,Prov\_,LastName,$Provider\_LastName);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Finding Taxonomy for PCP \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Searching for desired NPI records \*/

data npi\_data;

set npi.npi\_data;

where put(NPI,$Find\_NPI.)="\*";

/\* Providers can have taxonomy codes for any state. I need the code

that is relevant to Oklahoma which may not be the first one listed \*/

if Prvdr\_LicenseNbr\_State\_1="OK" then First\_Taxonomy\_Code=Prvdr\_Taxonomy\_Code\_1;

else if Prvdr\_LicenseNbr\_State\_2="OK" then First\_Taxonomy\_Code=Prvdr\_Taxonomy\_Code\_2;

else if Prvdr\_LicenseNbr\_State\_3="OK" then First\_Taxonomy\_Code=Prvdr\_Taxonomy\_Code\_3;

else if Prvdr\_LicenseNbr\_State\_4="OK" then First\_Taxonomy\_Code=Prvdr\_Taxonomy\_Code\_4;

else if Prvdr\_LicenseNbr\_State\_5="OK" then First\_Taxonomy\_Code=Prvdr\_Taxonomy\_Code\_5;

/\* There were a few providers that did not have a taxonomy code for OK so I

am going to go with the first Code listed \*/

else First\_Taxonomy\_Code=Prvdr\_Taxonomy\_Code\_1;

run;

/\* Creating formats for Credential & Taxonomy \*/

%Formatme(work,npi\_data,NPI,First\_Taxonomy\_Code,$Provider\_TaxonomyCode);

%Formatme(work,npi\_data,NPI,Prvdr\_Credential,$Provider\_Credential);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Member Address \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Finding address and phone number for each member \*/

data member\_address;

set memb.address;

where put(addno,$Find\_Members\_Addno.)="\*"

and input(ymdend,yymmdd8.)>=today()

and input(ymdeff,yymmdd8.)<=today()

and whotype="S"

/\* Medicare Members want Mail sent to M, but need D if no M \*/

and AddType in: ("D","M")

and void=" ";

run;

/\* Sorting to find most recent address \*/

proc sort data=member\_address; BY addno addtype YmdEff YmdEnd; run;

/\* Finding last known address \*/

data member\_address\_last (keep=addno addtype address: city state zip);

length Address $100.;

set member\_address;

by addno;

if last.addno;

address=catx(" ",address1, address2);

/\* For whatever reason there was a city record ending with a

comma that misaligns the output, correcting here \*/

if city="TULSA," then city="TULSA";

run;

/\* Creating formats \*/

%Formatme(work,member\_address\_last,Addno,address,$Member\_to\_Address);

%Formatme(work,member\_address\_last,Addno,City,$Member\_to\_City);

%Formatme(work,member\_address\_last,Addno,State,$Member\_to\_State);

%Formatme(work,member\_address\_last,Addno,ZIP,$Member\_to\_ZIP);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Member Phone Numbers \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Need to grab primary member phone - limiting to default address records \*/

data member\_phone;

length final\_myphone $120.;

set memb.address;

where put(Addno,$Find\_Members\_Addno.)="\*"

and input(ymdeff,? yymmdd8.)<=today()<=input(ymdend,? yymmdd8.)

and whotype="S"

and AddType in: ("D")

and void="";

/\* Looking for meaningful phone number and putting into

meaningful format \*/

if phone1 not in ("","0000000000") then do;

Priority=1;

temp\_myphone=compress(phone1,'() -.');

final\_myphone=compress(catx("",substr(temp\_myphone,1,3),

substr(temp\_myphone,4,3),

substr(temp\_myphone,7,4)));

temp\_storedphone1=final\_myphone;

output member\_phone;

end;

if phone2 not in ("","0000000000") then do;

Priority=2;

temp\_myphone=compress(phone2,'() -.');

final\_myphone=compress(catx("",substr(temp\_myphone,1,3),

substr(temp\_myphone,4,3),

substr(temp\_myphone,7,4)));

temp\_storedphone2=final\_myphone;

if temp\_storedphone2~=temp\_storedphone1 then output member\_phone;

end;

if phone3 not in ("","0000000000") then do;

Priority=3;

temp\_myphone=compress(phone3,'() -.');

final\_myphone=compress(catx("",substr(temp\_myphone,1,3),

substr(temp\_myphone,4,3),

substr(temp\_myphone,7,4)));

temp\_storedphone3=final\_myphone;

if (temp\_storedphone3~=temp\_storedphone2

and temp\_storedphone3~=temp\_storedphone1) then output member\_phone;

end;

run;

/\* Need Primary Phone so prioritizing Default over Mailing \*/

proc sort data=member\_phone nodupkey; by Addno AddType Priority final\_myphone; run;

/\* Keeping default phone1 where available \*/

data member\_phone\_primary member\_phone\_remaining;

set member\_phone;

by Addno AddType Priority;

if first.Addno then output member\_phone\_primary;

else output member\_phone\_remaining;

run;

/\* Creating format for phone \*/

%Formatme(work,member\_phone\_primary,Addno,final\_myphone,$MemberPrimaryPhone);

/\* Need Secondary Phone so re-sorting \*/

proc sort data=member\_phone\_remaining; by Addno AddType Priority; run;

/\* Keeping default phone2 or 3 where available \*/

data member\_phone\_secondary;

set member\_phone\_remaining;

by Addno AddType Priority;

if first.Addno;

run;

/\* Creating format for phone \*/

%Formatme(work,member\_phone\_secondary,Addno,final\_myphone,$MemberSecondaryPhone);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Formats for Star Measures \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* This table will need to updated each year as new measures are included or existing

ones are removed \*/

/\* Importing CMS Star Table for formats \*/

proc import datafile="\\twr-sas-1\DAR\HDA\Production\M\_Medicare HEDIS Rates\Source Data\&CutPointFile..xlsx"

dbms=excel

out=CMS\_Stars

replace;

run;

/\* Creating additional key for filtering \*/

data CMS\_Stars;

set CMS\_Stars;

FilterKey=catx("|",ChangeHealthcare\_Measure,ChangeHealthcare\_SubMeasure);

run;

/\* Flagging SubMeasures to Keep \*/

%Flagem(CMS\_Stars,ChangeHealthcare\_Measure,$CMS\_Measures);

%Flagem(CMS\_Stars,ChangeHealthcare\_SubMeasure,$CMS\_SubMeasures);

%Flagem(CMS\_Stars,FilterKey,$CMS\_MeasureSubmeasureKey);

/\* Creating format for Measure Abreviations \*/

%Formatme(work,CMS\_Stars,ChangeHealthcare\_SubMeasure,Measure\_Abreviation,$CMS\_ChangeSubMeasure);

%Formatme(work,CMS\_Stars,Measure\_Abreviation,MyMeasure,$CMS\_Measure\_Title);

%Formatme(work,CMS\_Stars,MyMeasure,Measure\_Abreviation,$CMS\_Measure\_Abreviation);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* HEDIS Extract \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Loading Extract that has already been imported and saved \*/

data ChangeHealthcare\_Extract\_Raw;

set extract.HEDIS\_Extract\_&DataMonth;

run;

/\* Checking format of variables that commonly cause issues \*/

proc contents data=ChangeHealthcare\_Extract\_Raw

out=Check\_Extract\_Variables(keep=Name Type Length) noprint;

run;

/\* Step for any filters if needed \*/

data ChangeHealthcare\_Extract;

set ChangeHealthcare\_Extract\_Raw;

/\* Applying Crosswalk \*/

MemberXwalk=put(MemberID,$xwalk.);

/\* Renaming ECDS SubMeasures because they are indentical to the standard

measures - Prevents format errors \*/

if substr(Measure,length(Measure)-1,2)="-E" then SubMeasure=catx("",SubMeasure,"(ECDS)");

/\* For measures with multiple sub-measures I need to filter down to the

relevant indicator(s) & ensure the measure is named correctly \*/

if put(catx("|",Measure,SubMeasure),$CMS\_MeasureSubmeasureKey.)="\*";

Final\_Measure\_Abreviation=put(SubMeasure,$CMS\_ChangeSubMeasure.);

Final\_Measure=put(Final\_Measure\_Abreviation,$CMS\_Measure\_Title.);

/\* Limiting OMW Fractures to those that can be intervened with \*/

if Final\_Measure\_Abreviation="OMW" then do;

if AcceptableDateRangeStart<=today()<=AcceptableDateRangeEnd;

end;

/\* Limiting to relevant measures \*/

else do;

/\* Either not star related or highly time sensitive \*/

if Final\_Measure\_Abreviation not in: ("","PCR","TRC","FMC","CDC Neph");

end;

%Runquit;

/\* If there are any errors I need this to stop so that they can be

corrected before moving forwards \*/

/\* Inversing CDC Blood Sugar Controlled (Hit defined to be when Poor Control is not a hit) \*/

data Final\_Extract\_Table Final\_Extract\_Table\_Exclusions

/\*Final\_Extract\_Table\_ECDS\*/;

set ChangeHealthcare\_Extract;

if Final\_Measure\_Abreviation="CDC A1c" then do;

if Admin\_Numerator=1 then Inverse\_Admin\_Numerator=0;

else if Admin\_Numerator=0 then Inverse\_Admin\_Numerator=1;

Final\_Admin\_Numerator=Inverse\_Admin\_Numerator;

end;

else Final\_Admin\_Numerator=Admin\_Numerator;

/\* Change is providing records for exclusions, we need to remove these to

prevent reporting gaps that don't count \*/

if Admin\_Denominator=0 then output Final\_Extract\_Table\_Exclusions;

/\* else if substr(Measure,length(Measure)-1,2)="-E" then output Final\_Extract\_Table;\*/

else output Final\_Extract\_Table;

drop Admin\_Numerator;

run;

/\* Noticing some duplicate records for BP that I am going to handle

- Not ideal but have no other option (records look identical except for

comments listing last BP differently) \*/

proc sort data=Final\_Extract\_Table nodupkey dupout=ExtractDuplicates; by Lob MemberID Measure SubMeasure EligibilityEventDate; run;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Part C Compliance \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Since I need to use the Member ID from the initial list so that files are consistent

I have to go backwards from the crosswalk ID. Since multiple numbers can be mapped

together I am opting for the safe route of building a format and tacking on compliance

data to the original list instead of trying to map member ID onto the compliance data

directly \*/

/\* Creating a Key for Transpose \*/

data PartC\_Compliance;

set Final\_Extract\_Table;

TransposeKey=catx(":",Final\_Measure\_Abreviation,Final\_Admin\_Numerator);

run;

/\* Sorting in preparation for tranpose \*/

proc sort data=PartC\_Compliance; by MemberXwalk TransposeKey; run;

/\* Getting All Measures on 1 Line \*/

proc transpose data=PartC\_Compliance out=PartC\_Compliance\_Trans;

by MemberXwalk;

var TransposeKey;

run;

/\* Putting all measures into one column for format \*/

data PartC\_Compliance\_Trans;

set PartC\_Compliance\_Trans;

AllCompliance=catx("|",of COL:);

run;

/\* Creating format \*/

%Formatme(work,PartC\_Compliance\_Trans,MemberXwalk,AllCompliance,$PartC\_Compliance);

/\* Attaching Part C to original list \*/

data PartC\_Compliance\_Final(keep=member\_id MemberXwalk

MeasureAbreviation MeasureCompliance);

set PopHealth\_Members(keep=member\_id MemberXwalk);

Compliance=put(MemberXwalk,$PartC\_Compliance.);

/\* Looping through to break out records \*/

do i=1 to 999;

temp\_Compliance=scan(Compliance,i,"|");

if temp\_Compliance="" then i=999;

else do;

MeasureAbreviation=scan(temp\_Compliance,1,":");

MeasureCompliance=input(scan(temp\_Compliance,2,":"),8.);

output;

end;

end;

run;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Part D Compliance \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Creating a Key for Transpose \*/

data PartD\_Compliance;

length Final\_Measure\_Abreviation $120.;

set partd.final\_days\_covered\_diab(in=Diab)

partd.final\_days\_covered\_ras(in=Ras)

partd.final\_days\_covered\_stat(in=Stat)

partd.supd\_final\_detail(in=SUPD);

/\* Applying crosswalk \*/

MemberXwalk=put(Mbr\_ID,$xwalk.);

/\* Classifying Measures \*/

if Diab=1 then Final\_Measure\_Abreviation="ADH - DIAB";

else if Ras=1 then Final\_Measure\_Abreviation="ADH - RAS";

else if Stat=1 then Final\_Measure\_Abreviation="ADH - STAT";

else if SUPD=1 then Final\_Measure\_Abreviation="SUPD";

TransposeKey=catx(":",Final\_Measure\_Abreviation,Compliant\_Flag);

run;

/\* Sorting in preparation for tranpose \*/

proc sort data=PartD\_Compliance; by MemberXwalk TransposeKey; run;

/\* Getting All Measures on 1 Line \*/

proc transpose data=PartD\_Compliance out=PartD\_Compliance\_Trans;

by MemberXwalk;

var TransposeKey;

run;

/\* Putting all measures into one column for format \*/

data PartD\_Compliance\_Trans;

set PartD\_Compliance\_Trans;

AllCompliance=catx("|",of COL:);

run;

/\* Creating format \*/

%Formatme(work,PartD\_Compliance\_Trans,MemberXwalk,AllCompliance,$PartD\_Compliance);

/\* Attaching Part D to original list \*/

data PartD\_Compliance\_Final(keep=member\_id MemberXwalk

MeasureAbreviation MeasureCompliance);

set PopHealth\_Members(keep=member\_id MemberXwalk);

Compliance=put(MemberXwalk,$PartD\_Compliance.);

/\* Looping through to break out records \*/

do i=1 to 999;

temp\_Compliance=scan(Compliance,i,"|");

if temp\_Compliance="" then i=999;

else do;

MeasureAbreviation=scan(temp\_Compliance,1,":");

MeasureCompliance=input(scan(temp\_Compliance,2,":"),8.);

output;

end;

end;

run;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Creating the Files\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Chase List File \*/

data export\_chaselist;

length ChaseListType $20. CampaignID $50. LOB $20. PersonUniqueID $80.

MemberID $80. LastName $50. FirstName $50. MidInit $1. DOB 8.

Lang $20. Gender $1. Address $250. City $50. State $2. ZIP $10.

Phone $10. MobilePhone $10. OtherPhone $10. PlanID $20.

PCPID $120. PCPNPI $120. ProviderEntity 8. ProviderOfficeName $100.

PCPLastName $100. PCPFirstName $55. TaxonomyCode $10.

PCPAddressLine1 $55. PCPAddressLine2 $55. PCPCity $40. PCPState $2.

PCPZIP $5. PCPOfficePhone $10. PCPMobilePhone $10. PCPFax $10.

PCPWebAddress $150. PCPTaxID $10. PCPMedicareID $10. PCPSpecCode $120.

cRiskLevel $20. cRiskScore 5.2 cRiskScoreType $20. ClientCareProgram $150.;

set PopHealth\_Members;

/\* Note some of these may change from file to file \*/

ChaseListType="XXXXXXXXXX";

CampaignID="XXXXXXXXXX";

PersonUniqueID=put(member\_id,$MBI.);

MemberID=member\_id;

LastName=put(MemberID,$Member\_LastName.);

FirstName=scan(put(MemberID,$Member\_FirstName.),1,"");

MidInit=scan(put(MemberID,$Member\_FirstName.),2,"");

DOB=input(put(MemberID,$Member\_DOB.),8.);

Lang=put(MemberID,$Language.);

Gender=put(MemberID,$Member\_Sex.);

Address=put(MemberAddno,$Member\_to\_Address.);

City=put(MemberAddno,$Member\_to\_City.);

State=put(MemberAddno,$Member\_to\_State.);

ZIP=put(MemberAddno,$Member\_to\_ZIP.);

Phone=put(MemberAddno,$MemberPrimaryPhone.);

MobilePhone="";

OtherPhone=put(MemberAddno,$MemberSecondaryPhone.);

Prog\_=put(MemberID,$Member\_to\_Program.);

if prog\_="XXXXXXXXXX" then do;

PlanID="XXXXXXXXXX"; LOB="Medicare";

end;

else if prog\_ in ("XXXXXXXXXX","XXXXXXXXXX") then do;

PlanID="XXXXXXXXXX"; LOB="Medicare";

end;

else if prog\_ in ("XXXXXXXXXX") then do;

PlanID="XXXXXXXXXX"; LOB="Medicare";

end;

if LOB="" then do;

LOB="Medicare";

end;

PCPID=put(MemberXwalk,$Member\_to\_Affiliation.);

PCP\_ID=put(MemberXwalk,$Member\_to\_PCP.);

PCPNPI=put(PCP\_ID,$Provider\_NPI.);

ProviderEntity=1; /\* Should all be people \*/

ProviderOfficeName="";

PCPLastName=put(PCP\_ID,$Provider\_LastName.);

PCPFirstName=put(PCP\_ID,$Provider\_FirstName.);

TaxonomyCode=put(PCPNPI,$Provider\_TaxonomyCode.);

PCPAddressLine1=put(PCP\_ID,$PCP\_to\_Address\_One.);

PCPAddressLine2=put(PCP\_ID,$PCP\_to\_Address\_Two.);

PCPCity=put(PCP\_ID,$PCP\_to\_City.);

PCPState=put(PCP\_ID,$PCP\_to\_State.);

PCPZIP=put(PCP\_ID,$PCP\_to\_ZIP.);

PCPOfficePhone=put(PCP\_ID,$PCP\_to\_Phone.);

PCPMobilePhone="";

/\* For PCP Fax using centralized Fax Number for Owners \*/

PCPTaxID=put(MemberXwalk,$Member\_to\_IRS.);

if PCPTaxID in ("XXXXXXXXXX","XXXXXXXXXX") then PCPFax="XXXXXXXXXX";

else if PCPTaxID in ("XXXXXXXXXX","XXXXXXXXXX") then PCPFax="XXXXXXXXXX";

else if PCPTaxID in ("XXXXXXXXXX") then PCPFax="XXXXXXXXXX";

else if PCPTaxID in ("XXXXXXXXXX") then PCPFax="XXXXXXXXXX";

else PCPFax=put(PCP\_ID,$Provider\_Fax.);

PCPWebAddress="";

PCPMedicareID="";

PCPSpecCode=put(put(MemberXwalk,$Member\_to\_Specialty.),$spdesc.);

cRiskLevel="";

cRiskScore=Current\_Score;

cRiskScoreType="CMS HCC";

ClientCareProgram="";

keep ChaseListType CampaignID LOB PersonUniqueID MemberID LastName

FirstName MidInit DOB Lang Gender Address City State ZIP

Phone MobilePhone OtherPhone PlanID PCPID PCPNPI ProviderEntity

ProviderOfficeName PCPLastName PCPFirstName TaxonomyCode

PCPAddressLine1 PCPAddressLine2 PCPCity PCPState PCPZIP

PCPOfficePhone PCPMobilePhone PCPFax PCPWebAddress PCPTaxID

PCPMedicareID PCPSpecCode cRiskLevel cRiskScore cRiskScoreType

ClientCareProgram;

format DOB yymmdd10. cRiskScore 6.2;

run;

/\* Sorting for export \*/

proc sort data=export\_chaselist; by PersonUniqueID MemberID; run;

/\* Care Gap File \*/

Data export\_caregap;

length PersonUniqueID $80. MemberID $80. GapType $20.

GapCode $20. GapDesc $100. GapCodeType 8.

GapSource $20. GapStatus $20. GapEventDate 8.;

set PartC\_Compliance\_Final(in=PartC)

PartD\_Compliance\_Final(in=PartD);

PersonUniqueID=put(member\_id,$MBI.);

MemberID=member\_id;

GapType="Care"; /\* These are all care gaps (Part C & D) \*/

GapCode=""; /\* Not required for Care Gaps \*/

GapSource="";

GapCodeType=.;

/\* Attaching Gap Description \*/

if MeasureAbreviation="BCS-E" then GapDesc="Breast Cancer Screening";\*ECDS;

else if MeasureAbreviation="COL-E" then GapDesc="Colorectal Cancer Screening";\*ECDS;

else if MeasureAbreviation="OMW" then GapDesc="Osteoporosis Management in Women who had a Fracture";

else if MeasureAbreviation="CDC Eye" then GapDesc="Diabetes Care - Eye Exam";

else if MeasureAbreviation="CDC A1c" then GapDesc="Diabetes Care - Blood Sugar Controlled";

else if MeasureAbreviation="CBP" then GapDesc="Controlling Blood Pressure";

else if MeasureAbreviation="SPC" then GapDesc="Statin Therapy for Patients With Cardiovascular Disease";

else if MeasureAbreviation="ADH - DIAB" then GapDesc="Medication Adherence for Diabetes Medications";

else if MeasureAbreviation="ADH - RAS" then GapDesc="Medication Adherence for Hypertension";

else if MeasureAbreviation="ADH - STAT" then GapDesc="Medication Adherence for Cholesterol";

else if MeasureAbreviation="SUPD" then GapDesc="Statin Use in Persons with Diabetes Measures";

else if MeasureAbreviation="KED" then GapDesc="Kidney Health Evaluation for Patients with Diabetes";

else GapDesc="DESCRIPTION MISSING";

/\* Converting Compliance to a Status \*/

if MeasureCompliance=1 then GapStatus="Closed";

else if MeasureCompliance=0 then GapStatus="Open";

/\* Adding Date Limitation - Different depending on the measure type because

Part D is calculated internally \*/

if PartC=1 then GapEventDate=&LimitationPartC;

else if PartD=1 then GapEventDate=&LimitationPartD;

keep PersonUniqueID MemberID GapType GapCode GapDesc GapCodeType

GapSource GapStatus GapEventDate;

format GapEventDate yymmdd10.;

run;

Proc Summary Data= PartC\_Compliance\_Final;

class MeasureAbreviation;

OUtput Out= AbreviationCheck; Run;

/\* Sorting for export \*/

proc sort data=export\_caregap; by PersonUniqueID MemberID GapDesc; run;

/\* Checking Gap Description Names \*/

proc freq data=export\_caregap noprint;

tables GapDesc/out=check\_names;

run;

/\* Removing the Description Missing records as they are mostly Overlapping Members

due to changes in Changes of Measure Abbriviation\*/

Data Final\_export\_caregap; Set export\_caregap;

Where GapDesc ne 'DESCRIPTION MISSING'; Run;

Proc Summary Data=Final\_export\_caregap nway missing;

Class GapDesc;

Output Out=Description\_Check; Run;

\*Create a place holder data set

This really has nothing important in it

I need it for the below macro check to work;

Data placeholder;

input VariablePH;

cards;

1

run;

%Macro Checking\_stuff (description);

Data Check\_Description; Set Description\_Check;

Where GapDesc = &description; run;

Data \_null\_; Set Check\_Description placeholder nobs=num\_obs;

If num\_obs = 2 Then Do;

End;

Else Do;

put 'Check Measure Mapping - Your Gap file has missing records for something';

abort;

end;

run;

%Mend;

%Checking\_stuff('Breast Cancer Screening')

%Checking\_stuff("Colorectal Cancer Screening")

%Checking\_stuff("Osteoporosis Management in Women who had a Fracture")

%Checking\_stuff("Diabetes Care - Eye Exam")

%Checking\_stuff("Diabetes Care - Blood Sugar Controlled")

%Checking\_stuff("Controlling Blood Pressure")

%Checking\_stuff("Statin Therapy for Patients With Cardiovascular Disease")

%Checking\_stuff("Medication Adherence for Diabetes Medications")

%Checking\_stuff("Medication Adherence for Hypertension")

%Checking\_stuff("Medication Adherence for Cholesterol")

%Checking\_stuff("Statin Use in Persons with Diabetes Measures")

%Checking\_stuff("Kidney Health Evaluation for Patients with Diabetes")

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Exporting the Files \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Exporting the Chase List File \*/

filename Chase "&XXXXXXXXXX\XXXXXXXXXX\Medicare\_&MeasurementYear.\_ChaseList\_&logdate..txt" termstr=CRLF;

proc export data=export\_chaselist

outfile=Chase

dbms=dlm label replace;

delimiter='|';

run;

proc export data=WORK.EXPORT\_CHASELIST

outfile="\\XXXXXXXXXX - Emcara IHA Target and Gap File\Output\Medicare\_2022\_ChaseList\_20230201\_0705.txt"

dbms=csv

replace;

delimiter='|';

run;

/\* Exporting the Care Gap File \*/

filename CareGap "&path\Output\Medicare\_&MeasurementYear.\_CareGap\_&logdate..txt" termstr=CRLF;

proc export data=Final\_export\_caregap

outfile=CareGap

dbms=dlm label replace;

delimiter='|';

run;