```
/******************************************************************************************
*********************
Title: Asthma 4
Original Author: Dr Lawrence Kleinman
Version written by: Robert Thombley
Last Edited Date: Aug 02, 2021
Description: SAS code to calculate Asthma 4 measure using administrative claims data.
Required Input Data Files: (see input file specification documentation for more details) All
input data should be
                                                        in SAS7BDAT format and conform to the
specifications listed.

                1) Eligibility Table: A list of all periods of continuous enrollment periods
longer than 3 months by member and
                                                        payer (aka insurance provider). Each row
is a continuous enrollment period for a given member
                                                        and insurance provider/payer. Records
for a given member should not overlap
                                                        in time, but each member may have
multiple records due to enrollment with different providers
                                                        or non-continuous enrollment periods
with the same provider.

                2) Medical Claim Table: All medical claims for the relevant population. Claims
will be filtered by time,
                                                                diagnosis code, and age at
service but can be pre-filtered to cut down on size.
                                                                Both the ED visits (denominator)
and PCP follow up visits (numerator) will be
                                                                selected from this data, so it
must include, at a minimum, all asthma related ED
                                                                visit claims and all office
visit claims for the lookback and evaluation periods.
                                                                Data should be in wide format (1
row = 1 service line - ie: reimbursed charge from the claim)
                                                                with, at minimum, the first two
ICD9/10 codes included.

                3) NPI and Taxonomy Code Table: A separate table that will serve as the
reference table for "acceptable"
                                                                followup provider NPIs. This can
be generated directly from the NPPES. It should include
                                                                the provider NPI as well as the
"CLASSIFICATION" field from the NUCC
                                                                (National Uniform Claim
Committee) Health Care Provider Taxonomy Code Set, which
                                                                can be found here
https://www.nucc.org/index.php/code-sets-mainmenu-41/provider-taxonomy-mainmenu-40
                                                                and cross referenced with the
primary specialty for each NPI, as listed in
                                                                the NPPES.


Output:  ast4_out.measure_data<file_suffix>: This table will contain all qualifying ED index
visits by member-month, as well as
                an indicator of whether each index visit had a followup within 14 days
(inclusive). The payer associated with each member
                month is the most recent payer for the member.

*******************************************************************************************
********************/


/**************************************** BEGIN USER EDITABLE CONFIGURATION
**************************************/
```

```
* Define the library name for use throughout the codebase;
libname ast4_in "<PATH/TO/INPUT/DATA/FILES>";
libname ast4_out "<PATH/TO/OUTPUT/DATA/FILES>";

* Define analysis years;
%let lookyear = 2014; * The lookback year;
%let evalyear = 2015; * The evaluation year;
%let file_suff = _15CA4; *File suffix to be used for files associated with the current analysis;

* Source Data Tables;
%let eligibility_table = /*ast4_in.ENROLLMENT_TABLE*/;
%let medical_claims = /*ast4_in.MEDICAL_CLAIMS*/;

* NPI Reference table - this table should include all distinct NPIs
(PROVIDING/RENDERING/BILLING) from the
* numerator claims data along with their NUCC "Classification" value;

%let npi_taxonomy_list = /*ast4_in.NPI_DATA*/;


/******************************************** END USER EDITABLE CONFIGURATION
*******************************************/

/******************************************** BEGIN INTERNAL CONFIGURATION - DO NOT EDIT
*******************************/

* Define beginning and ending dates - this should not need to be edited;
%let lookstart = 01Jan&lookyear.; *First date of the lookback year;
%let evalstart = 01Nov&lookyear.; *First date of the evaluation year;
%let evalend = 31Oct&evalyear.; *Last date of the evaluation year;

* Measure inclusion criteria;
* CPT & Revenue Codes;
%let ed_visit_cpt = '99281','99282','99283','99284','99285';
%let ed_visit_rev_code = '0450','0451','0452','0456','0459','0981';
%let office_visit_cpt =
'99201','99202','99203','99204','99205','99211','99212','99213','99214','99215',
'99241','99242','99243','99244','99245';
%let home_health_cpt = '99341','99342','99343','99344','99345','99347','99348','99349','99350';
%let cap_prev_med_cpt =
'99381','99382','99383','99384','99385','99391','99392','99393','99394','99395','99401','99402',
'99403','99404','99411','99412','99420','99429';
%let cap_prev_med_hcpcs = 'G0438','G0439';

* ICD9/10 Inclusion Codes;

* ICD9 = 493.*;
%let icd9_inclusion_asthma
='49300','49301','49302','49310','49311','49312','49320','49321','49322',
                                                '49381','49382','49390','49391','49392';
* icd10 = J45.*;
%let icd10_inclusion_asthma =
'J4520','J4521','J4522','J4530','J4531','J4532','J4540','J4541','J4542','J4550',

'J4551','J4552','J45901','J45902','J45909','J45990','J45991','J45998';


* The minimum and maximum ages to include;
%let min_included_age_at_service = 3;
%let max_included_age_at_service = 21;




/* Identify the last date in February for the evaluation year, since it may change if it is a
```

```
leapyear.
        The value is stored as the global variable &last_feb_day.
*/
data _NULL_;
        end_day = day(intnx('MONTH',mdy(2,1,&evalyear.),0,'end'));
        call symput('last_feb_day', put(end_day, 2.));
run;

/********************************************** END INTERNAL CONFIGURATION
***********************************************/



/********************************************** BEGIN TABLE SETUP
********************************************************/



/* Create reference tables for use throughout the analysis. */

* Create reference list of all unique member/payer enrollment periods within the study period;
        proc sql;
                CREATE TABLE enrollment_log as
                SELECT DISTINCT member_id,
                                                payer_id,
                                                payer_start_dt,
                                                payer_end_dt
                FROM &eligibility_table.;
        quit;


        /* Filter all medical claims for use in the denominator to include only those having:
                - diagnosis codes matching the measure inclusion critieria diagnosis codes
                - member/payer ids match those in the enrollment_log table
                - claim service date is within the measure time frame
                - claim service date is within the payer enrollment period for the member
        */
        proc sql;
                CREATE TABLE all_ast_clms_plus_enroll AS
                SELECT DISTINCT a.member_id,
                        a.service_dt,
                        a.zip_code,
                        a.DX1,
                        a.DX2,
                        a.procedure_code,
                        a.revenue_code,
                        a.payer_id,
                        a.product_type,
                        a.product_id,
                        a.age_at_service,
                        b.payer_start_dt,
                        b.payer_end_dt
                FROM &medical_claims. a
                INNER JOIN enrollment_log b
                ON a.member_id = b.member_id AND a.payer_id = b.payer_id
                WHERE
                        (a.DX1 IN (&icd9_inclusion_asthma., &icd10_inclusion_asthma.) OR
                        a.DX2 IN (&icd9_inclusion_asthma., &icd10_inclusion_asthma.))
                        AND a.service_dt BETWEEN "&lookstart"d AND "&evalend"d
                        AND a.service_dt BETWEEN b.payer_start_dt AND b.payer_end_dt;
        quit;

        /* Filter all medical claims for use in the numerator. */
        proc sql;
                        CREATE TABLE all_num_clms_plus_enroll AS
                        SELECT DISTINCT
                                        a.member_id,
                                        a.service_dt,
                                        a.procedure_code ,
                                        a.payer_id,
                                        a.age_at_service,
                                        a.NPI_PROVIDING,
                                        a.NPI_BILLING,
                                        a.NPI_RENDERING
                        FROM &medical_claims. a
                        WHERE a.service_dt BETWEEN "&lookstart"d AND "&evalend"d;
        quit;



* Filter the NPI list to include only providers with the follwing classifications: Allergy &
Immunology, Family Medicine,
* Internal Medicine, General Practice or Pediatrics.
* Note that Pulmonary medicine is included under Pediatrics and/or Internal Med;
        proc sql;
                create table TAXO_FILTERED_PROVS as
                select NPI, TAXONOMY_CODE, CLASSIFICATION, 1 as PCP_TYPE_OK
                from &npi_taxonomy_list. a
                where not missing(CLASSIFICATION) AND
                        (classification="Allergy & Immunology"
                                OR classification="Family Medicine"
                                OR classification="Internal Medicine"
                                OR classification="General Practice"
                                OR classification="Pediatrics");
        quit;


/********************************************* END TABLE SETUP
****************************************/

/********************************************* BEGIN DENOMINATOR CALCULATION
****************************************/

        /* Find all denominator qualifying events: ED visits with Asthma in the Dx1 or Dx2 slot
in the evaluation time frame.*/
        proc sql;
                create table a4_ed_visit as
                select member_id, service_dt, product_type, product_id, zip_code,
age_at_service, payer_id
                from all_ast_clms_plus_enroll a
                where (a.revenue_code in (&ed_visit_rev_code.) OR a.procedure_code in
(&ed_visit_cpt.))
                order by a.member_id, a.service_dt;
        quit;

        /* De-duplicate based on member_id, payer and service date to create unique service
events for each member_id,
                separated by payer, if necessary. Multiple ED claims from a single payer on a
single day will only be counted once. */
        proc sort data = a4_ed_visit out = all_ed_clean nodupkey;
                by member_id payer_id service_dt;
        run;

        * Build monthly lists of MEMBER level ED index dates, filtered to remove any members
that fail to meet the continuous enrollment
        criteria;

        %MACRO calcMonthDenom(mth, start_date, end_date, ce_end_date);

                * Restrict to the current month & remove any patients that are outside our
included age range;
                data &Mth;
                        set all_ed_clean;
                        where (&start_date. le service_dt le &end_date.);
                        if age_at_service lt &min_included_age_at_service.
                                or age_at_service gt &max_included_age_at_service. then delete;
                run;

                * Find all member ED visits where the member has at least 2 months of
eligibility (with any payer) beyond the current month;
                proc sql;
                        CREATE TABLE &mth._CE as
                                SELECT * from &Mth.
                                where MEMBER_ID IN (
                                        SELECT DISTINCT a.MEMBER_ID
                                        FROM &Mth. a
                                        inner join enrollment_log b
                                        on a.payer_id = b.payer_id and a.member_id = b.member_id
                                        WHERE b.payer_start_dt <= &start_date. and
b.payer_end_dt >= &ce_end_date.)
                                ORDER BY member_id;
                quit;

                proc sort data=&Mth._CE;
                        by MEMBER_ID descending SERVICE_DT;
                run;


                /* Since the measure is attributed at the member level, we want to have a unique
association of payer details to the member.
                For cases where there are multiple payers in a month, we will use the payer
level details from the member's last ED visit in the month.
                */

                data &Mth._payer_details;
                        set &Mth._CE;
                        where not missing(PAYER_ID);
                        by member_id;
                        if first.member_id then output;
                run;

                /* Add member level visit count to the most recent member-payer level details.
This creates a
                        table that identifies event totals per member and attributes them to the
most recent payer for that member.
                */
                proc sql;
                        create table &Mth._total_mem_level as
                        select a.member_id,
                                   a.payer_id,
                                   a.zip_code,
                                   a.product_type,
                                   a.product_id,
                                   a.age_at_service,
                                  "&Mth." as Month,
                                   b.tot_ed
                        from &Mth._payer_details a
                        inner join (select MEMBER_ID, count(DISTINCT(SERVICE_DT)) as tot_ed from
&Mth._CE group by MEMBER_ID) b
                        on a.member_id = b.member_id;
                quit;

            /* Now add all of the ED_Index visit dates back in. These are the member level ED
visit dates for the month */
                proc sql;
                        create table &Mth._denom_ce2 as
                        select a.MEMBER_ID, a.payer_id, a.product_type, a.product_id,
a.zip_code,
                                        a.month, a.tot_ed,
                                        b.service_dt, b.age_at_service
                        from &Mth._total_mem_level a
                        inner join &mth._CE b
                        on a.MEMBER_ID = b.MEMBER_ID;
                quit;

        %MEND;

        * Run for each month in the evaluation year;
        %calcMonthDenom (Nov,"01Nov&lookyear"d,"30Nov&lookyear."d, "31Jan&evalyear"d);
        %calcMonthDenom (Dec,"01Dec&lookyear."d,"31Dec&lookyear."d,
"&last_feb_day.Feb&evalyear."d);
        %calcMonthDenom (Jan,"01Jan&evalyear"d,"31Jan&evalyear."d, "31Mar&evalyear"d);
        %calcMonthDenom (Feb,"01Feb&evalyear"d,"&last_feb_day.Feb&evalyear."d,
"30Apr&evalyear"d);
        %calcMonthDenom (Mar,"01Mar&evalyear"d,"31Mar&evalyear."d, "31May&evalyear"d);
        %calcMonthDenom (Apr,"01Apr&evalyear"d,"30Apr&evalyear."d, "30Jun&evalyear"d);
        %calcMonthDenom (May,"01May&evalyear"d,"31May&evalyear."d, "31Jul&evalyear"d);
        %calcMonthDenom (Jun,"01Jun&evalyear"d,"30Jun&evalyear."d, "31Aug&evalyear"d);
        %calcMonthDenom (Jul,"01Jul&evalyear"d,"31Jul&evalyear."d, "30Sep&evalyear"d);
        %calcMonthDenom (Aug,"01Aug&evalyear"d,"31Aug&evalyear."d, "31Oct&evalyear"d);
        %calcMonthDenom (Sep,"01Sep&evalyear"d,"30Sep&evalyear."d, "30Nov&evalyear"d);
        %calcMonthDenom (Oct,"01Oct&evalyear"d,"31Oct&evalyear."d, "31Dec&evalyear"d);

/* Concatenate all of the monthly data into one large denominator table. */
        data denominator_raw_data (rename=(service_dt=ED_INDEX_DT));
                set nov_denom_ce2 (in=nov) dec_denom_ce2 (in=dec) jan_denom_ce2 feb_denom_ce2
mar_denom_ce2 apr_denom_ce2 may_denom_ce2 jun_denom_ce2
                        jul_denom_ce2 aug_denom_ce2 sep_denom_ce2 oct_denom_ce2;
                format eval_start_dt date9.;
                format eval_end_dt date9.;
                attrib yr length=$4;
                if nov = 1 or dec = 1 then yr = &lookyear.;
                else yr = &evalyear.;
                eval_start_dt = input("01" || month || yr, date9.);
                eval_end_dt = intnx('month',eval_start_dt,0,'end');
        run;

        proc sort data=denominator_raw_data;
                by member_id payer_id ed_index_dt;
        run;

* Deduplicate multiple ED visits that occur within 5 days of one another - use only the date of
the initial visit as the index date;
* Beyond this 5 day window, we keep multiple index dates for each patient;

        data ast4_out.denominator_raw_data (drop=lag_since_index blackout index_dt);
                set work.denominator_raw_data;
                by member_id;
                retain lag_since_index index_dt;
                format index_dt date9.;
                if first.member_id then do;
                        lag_since_index = 0;
                        blackout=0;
                        index_dt = ED_INDEX_DT;
                end;
                else do;
                        lag_since_index = ED_INDEX_DT - index_dt;
                        if lag_since_index <=5 then do;
                                blackout = 1;
                        end;
                        else do;
                                lag_since_index = 0;
                                blackout = 0;
                                index_dt = ED_INDEX_DT;
                        end;
                end;
                if blackout then delete;
        run;

/********************************************* END DENOMINATOR CALCULATION
*****************************************/



/********************************************* BEGIN NUMERATOR CALCULATION
*****************************************/

/*  Calculating Numerator - Identify, for each month of the reporting year, those kids in the
denominator who had a followup
                                                         visit in an ambulatory setting with a
"PCP-type" provider. It is
                                                         permissible for a single patient to have
multiple ED index visits, as long as they are > 5 days apart from
                                                         one another. A follow up visit is
defined as an ambulatory visit with a "PCP-type" provider within
                                                         14 days of the index ED visit.
*/

* This macro will identify all potential followup visits and calculate our numerator score;
%MACRO calculateNumerator(Mth, Mth_num, BeginDt, EndDt);

        * Filter the numerator claims down to office visits within the current month;
        proc sql;
                create table &Mth._fu_visits_wide as
                        select a.*,
                                        a.service_dt as visit_dt
                        from all_num_clms_plus_enroll a
                                        where a.procedure_code IN (&office_visit_cpt.,
&home_health_cpt., &cap_prev_med_cpt., &cap_prev_med_hcpcs.)
                                        AND a.service_dt BETWEEN &beginDt. AND &endDt.
                                        AND a.age_at_service BETWEEN
&min_included_age_at_service. and &max_included_age_at_service.;
        quit;
        * For each of the potential visits, determine whether any of the NPIs included on the
claim match one of the 'permissible'
        * NPIs we determined above;

        proc sql;
                create table &Mth._fu_visits_npi as
                select a.*,
                        coalesce(r.CLASSIFICATION,p.CLASSIFICATION,b.CLASSIFICATION) as
CLASSIFICATION,
                        coalesce(r.PCP_TYPE_OK, p.PCP_TYPE_OK, b.PCP_TYPE_OK, 0) as PCP_TYPE_OK
                from &Mth._fu_visits_wide a
                left join TAXO_FILTERED_PROVS b
                on b.NPI = a.NPI_BILLING
                left join TAXO_FILTERED_PROVS p
                on p.NPI = a.NPI_PROVIDING
                left join TAXO_FILTERED_PROVS r
                on r.NPI = a.NPI_RENDERING;
        run;

        * Identify all followup visits that have an NPI associated with one of the provider
classifications of interest;
        proc sort data=&Mth._fu_visits_npi out=fu_taxo_unique nodupkey;
                where pcp_type_ok = 1;
                by MEMBER_ID  SERVICE_DT;
        run;

        * Match the follow up visits with the index dates and exclude any that are 15 or more
days after the ED visit;
        proc sql;
                create table &Mth._numerator_all as
                select a.*,
                                b.service_dt as visit_dt format=date10.,
                                b.pcp_type_ok as PCP_FU,
                                b.CLASSIFICATION
                from ast4_out.denominator_raw_data a
                inner join work.fu_taxo_unique b
                on a.member_id = b.member_id
                        and 0 <= (b.service_dt - a.ED_INDEX_DT) <= 14
                where eval_start_dt = &beginDt.
                order by a.member_id, a.ED_INDEX_DT;
        quit;


        * Remove all but the FIRST qualifying followup, if any, for a given member-ED index
visit pairing;
        proc sort data=&mth._numerator_all out=&Mth._fu_first_14 nodupkey;
                by member_id ed_index_dt;
        run;

        * Join the numerator data to the denominator data, at the monthly level;
        proc sql;
                create table &Mth._num_out as
                select a.member_id, a.payer_id,
                a.PRODUCT_TYPE, a.PRODUCT_ID,
                a.ED_INDEX_DT, a.age_at_service,
                a.month, a.eval_start_dt, a.eval_end_dt, a.zip_code,
                fu.VISIT_DT as DATE_PCP_FU_14,
                fu.CLASSIFICATION as PCP_FU_14_TAXO,
                coalesce(fu.PCP_FU,0) as PCP_FU_14
                from ast4_out.denominator_raw_data a
                left outer join &Mth._fu_first_14 fu
                on a.member_id = fu.member_id and a.ed_index_dt = fu.ed_index_dt
                where a.eval_start_dt = &beginDt.;
        quit;
%MEND calculateNumerator;

        * Calculate numerator counts for each member, by month;
        %calculateNumerator (Nov,11,"01Nov&lookyear."d,"31Jan&evalyear"d);
        %calculateNumerator (Dec,12,"01Dec&lookyear."d,"&last_feb_day.Feb&evalyear"d);
        %calculateNumerator (Jan,1,"01Jan&evalyear"d,"31Mar&evalyear"d);
        %calculateNumerator (Feb,2,"01Feb&evalyear"d,"30Apr&evalyear"d);
        %calculateNumerator (Mar,3,"01Mar&evalyear"d,"31May&evalyear"d);
        %calculateNumerator (Apr,4,"01Apr&evalyear"d,"30Jun&evalyear"d);
        %calculateNumerator (May,5,"01May&evalyear"d,"31Jul&evalyear"d);
        %calculateNumerator (Jun,6,"01Jun&evalyear"d,"31Aug&evalyear"d);
        %calculateNumerator (Jul,7,"01Jul&evalyear"d,"30Sep&evalyear"d);
        %calculateNumerator (Aug,8,"01Aug&evalyear"d,"31Oct&evalyear"d);
        %calculateNumerator (Sep,9,"01Sep&evalyear"d,"30Nov&evalyear"d);
        %calculateNumerator (Oct,10,"01Oct&evalyear"d,"31Dec&evalyear"d);

        * Join all months together;
        data numerator_ref_data;
                set nov_num_out dec_num_out jan_num_out feb_num_out mar_num_out apr_num_out
may_num_out jun_num_out
                        jul_num_out aug_num_out sep_num_out oct_num_out;
        run;

        * Output final dataset;
        proc sort data = numerator_ref_data out=ast4_out.measure_data&file_suff.;
                by member_id ed_index_dt;
        run;

/********************************************** END NUMERATOR CALCULATION
************************************************/
```