SUMMARY TABLE V2 PROGRAMMING SPECIFICATIONS

Version 1.0

June 24, 2016

Table of Contents

I. DOCUMENT HISTORY

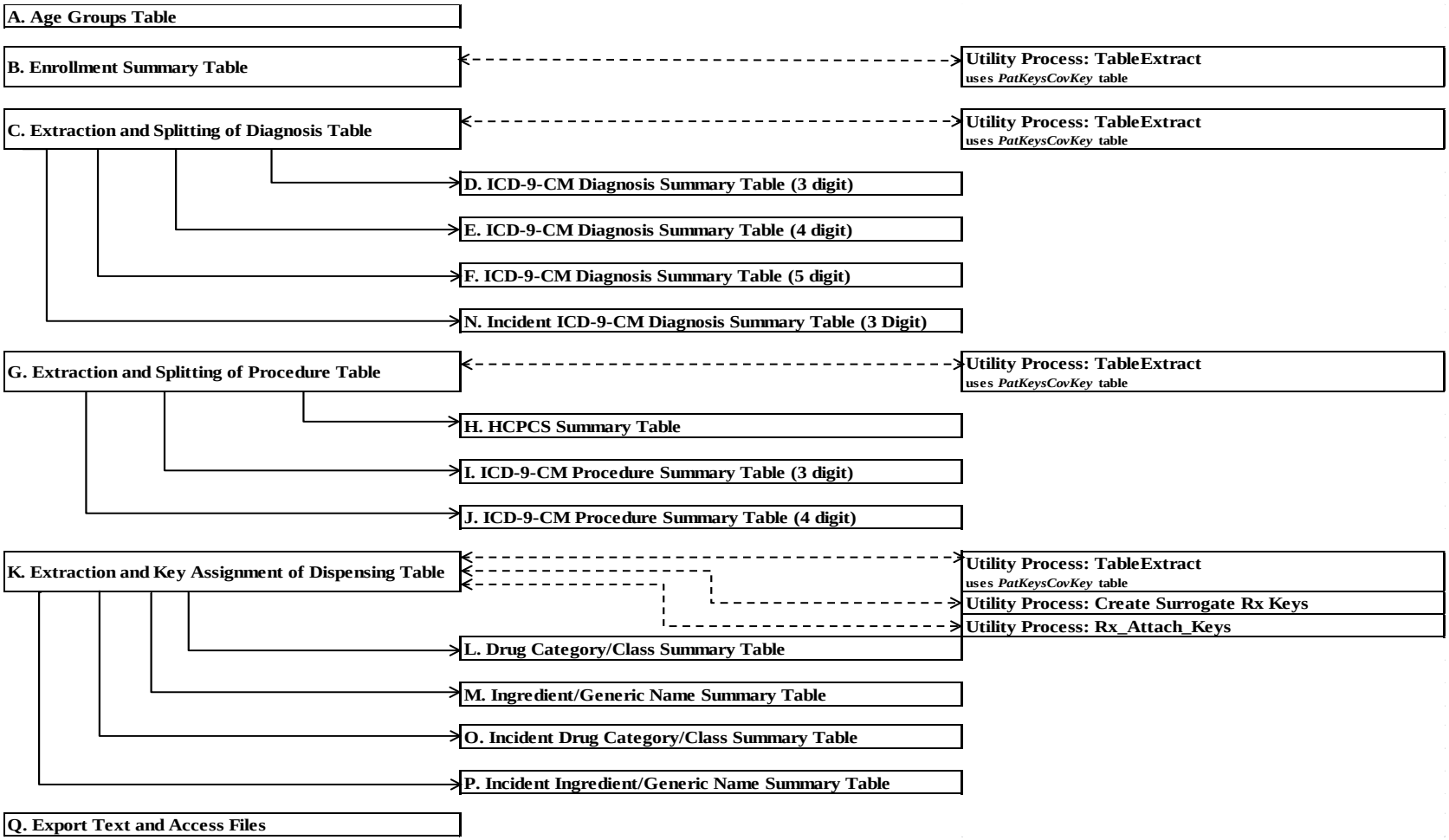
The following table is a revision history for this document.

Date	Version	Comments
June 24, 2016	1.0	First production version

II. OVERVIEW

The purpose of this programming specification is to describe the logic and processes required to create Summary Tables for the Sentinel Query Tool. These Summary Tables use selected Sentinel Common Data Model (SCDM) core tables as the data source. The Summary Tables are aggregates of the SCDM based on a number of business rules and strata, described in each table.

This specification describes the process for programming each of 13 tables. The process for creating some tables is dependent upon intermediate results from utility processes. This diagram illustrates both the dependencies and utility processes.



III. UTILITY PROCESSES/MODULES

For creating the Summary Tables, these are processes that are utilized by the project and are developed as distinct modules.

A. CREATE PATKEYSCOVKEY TABLE

The **PatKeysCovKey** table will contain a list of each patient who has any valid enrollment. The table will include the patient's birth date, sex and a flag indicating whether they have had at least one day each of medical coverage and drug coverage in each year of Data Partner data. This table will be used for filtering in of all utilization records.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
PatID	Char(##)	$##.	Patient ID	Free text of Data Partners' PatID	SCDM Demographic table
PatKey	Num(#)	#.	PatKey	Numeric key on a 1:1 relationship with PatID	This is a surrogate key for the PatID to purposely contain a fewer number of bytes that the PatID; the length of the variable will be the smallest SAS numeric LENGTH needed to contain the largest value
Birth_Date	Num(4)	mmddyy10.	Birth Date	Non-missing birth date	SCDM Demographic table
Sex	Char(1)	1.	Sex	F = Female M = Male	SCDM Demographic table
CovKeyDrug	Num(#)	#.	Drug Coverage Key	Numeric key with bits representing drug coverage for each year of Data Partner data	This is a unique numeric value consisting of 0/1 bits. Each position represents a calendar year
CovKeyMed	Num(#)	#.	Medical Coverage Key	Numeric key with bits representing medical coverage for each year of Data Partner data	This is a unique numeric value consisting of 0/1 bits. Each position represents a calendar year
CovKeyDrugAndMed	Num(#)	#.	Drug and Medical Coverage Key	Numeric key with bits representing combined coverage for each year of Data Partner data	This is a unique numeric value consisting of 0/1 bits. Each position represents a calendar year

Methods for Creating ***PatKeyCovKey*** table

1. From the SCDM Enrollment table, create a list of all unique patients that satisfy these conditions:
   1.1. MedCov is one of values "Y" or "N".
   1.2. DrugCov is one of values "Y" or "N".
   1.3. Enr_Start is earlier than or equal to Enr_End and neither variable is missing.
2. Link the table from Step 1 with the SCDM Demographic table on PatID, keeping records where Demographic.Birth_Date is not missing and Demographic.Sex is one of values "F" or "M".
3. Create variable PatKey as a numeric key with values, starting with 1, that are a 1:1 relationship to PatID.
4. Sort the table by PatID, name it as **PatKey** and save the table to the temporary work area.

5. Create temporary table for use in all prevalent tables: Read the table from Step 1 and link it with the **PatKey** table on PatID, to obtain the PatKey variable.
6. For each PatKey, create a row per Data Partner calendar year as follows.
   6.1. Create a new variable, Year, which is a 4-character representation of the calendar year
   6.2. Save the the MedCov and DrugCov variables.
7. Sort this file by PatKey and Year, name it **Prevalent_Patients** and save it to the temporary work area.
8. Read the table from Step 7 and, for each patient (PatKey), create a numeric variable for each of drug coverage (CovKeyDrug) and medical coverage (CovKeyMed) as follows.
   8.1. For each year in Data Partner data, create a 0/1 bit representation of coverage. This is a unique numeric value consisting of 0/1 bits. Each position represents a calendar year, ordered from left to right. For example, if the Data Partner has data for 2006-2008 (3 years of coverage), then the possible binary and decimal values for each of CovKeyDrug and CovKeyMed will be the following.

Binary Decimal Interpretation

000	0	No coverage in any of 2006-2008
001	1	Coverage in 2008 only
010	2	Coverage in 2007 only
011	3	Coverage in 2007 and 2008
100	4	Coverage in 2006 only
101	5	Coverage in 2006 and 2008
111	6	Coverage in 2006, 2007, and 2008

   8.2. Create a variable, CovKeyDrugAndMed, structured similarly to CovKeyDrug and CovKeyMed. Set each position's 0/1 value to 1 when both CovKeyDrug and CovKeyMed have coverage in the same year. If either or both of CovKeyDrug and CovKeyMed don't have coverage, then set the position to 0.
9. Structure the table as per the data dictionary above with one row per patient. Set the numeric number of bytes for variables PatKey, CovKeyDrug, CovKeyMed, and CovKeyDrugAndMed to the minimal necessary to hold the largest decimal integer value.
10. Sort the table by PatKey, naming it **PatKeyCovKey** and save it to the temporary work area.

B. TABLEEXTRACT MACRO

This macro performs the following functions in creating derivatives of SCDM tables for use in the Summary Tables programming process:

11. Reads a SCDM or other specified table
12. Filters in specific rows based on specified criteria
13. Performs minor transformations/recoding/cleaning, including creation of new variables

14. Links in demographic variables Birth_Date and Sex
    14.1. Calculates Age Group Keys for both prevalent and incident purposes
15. Creates surrogate key(s) for specific variables
16. Names specific variables to be saved
17. Splits the data by time segment, if requested (e.g., calendar year, calendar quarters)

Inputs:

Requires DEM_ETL completion
Macro parameters
SCDM table

Outputs:

Filtered, transformed SCDM tables split by time segment, if requested, *OR* a single view

Methods for Creating ***TableExtract*** macro

1. Create a macro with the following calling parameters:

Parameter	Short Description	Requirements and Details
MSCDMTable	Name of SCDM table to be read	Required parameter; must be filled; fully qualified (1-level or 2-level) name
TableInVars	Full list of variables to be read from SCDM table	Required parameter; includes all variables required for filtering and writing
EventDate	Utilization event date variable	Required parameter; must meet naming requirements for a SAS variable; this is the variable that is used for calculating age for tables
FilterCode	Logical expression code for filtering in observations	Required parameter; can contain multiple expressions
TransformCode	Code for transformations, recoding, cleaning, including creation of new variables	Required parameter; must be valid SAS DATA STEP code, with semi-colons per statement
ComputeAge	Y=Compute age keys N=Do not compute age keys	Required parameter. Determine whether or not to compute age key surrogate variables for age as of beginning of period and age as of event date. Default=Y
DemHashTable	Name of Demographic derivative table to be read	Required parameter; must name a valid SAS dataset, to be loaded into DATA STEP hash memory
HashVarKeys	Key variable(s) for linking in hash join	Required parameter; must name valid SAS variable(s) found in the table named by DemHashTable parameter

Parameter	Short Description	Requirements and Details
HashVarsRead	List of additional variables from DemHashTable to be placed in hash memory	Required parameter; must name valid SAS variable(s) found in the table named by DemHashTable parameter; HashVarsRead can include variables named by HashVarKeys, although this is not necessary
CovKeyVars	Surrogate key variable(s) for years of drug and medical coverage	Optional parameter; name of numeric variable(s) that contain unique keys indicating drug and/or medical or combined coverage; any CovKeyVars specified here must exist in the DemHashTable
OutName	Output file name (or name stub for split files)	Required parameter as single word as a name for a valid SAS dataset; can include two-level names
OutType	Specifies output as datasets(s) or a single view	Optional parameter as single character: D = Dataset(s); this is the default V = View
Lengths	Enables specifying variable names and lengths	Optional parameter to control lengths of variables output to the file(s) specified by OutName; follows syntax requirements of the SAS LENGTH statement
TableOutVars	Full list of variables to be written out to output dataset(s)	Required parameter; must name valid SAS variables
SplitTime	Time split intervals: Y=by calendar year Q=by calendar quarters N=single output file with no splitting	Required parameter and must be Y, Q, or N. If Y or Q, output file names are prefixed with OutName; then when: =Y, 4-digit year is suffix =Q, 4-digit year, followed by "Q", followed by 1-4 as suffix NOTE: If OutType=V then SplitTime *must* = N
ReplaceVarPairs	Pairs of replacement variable names for SORTEDBY dataset parameter	Optional parameter. Pairs are delimited from each other by a blank. Pairs are delimited by a pipe ("|") character in the pattern: OriginalVariableName|ReplacementVariableName

2. The first part of the macro enables passing on the SORTEDBY variables, if any from the file named by the MSCDMTable parameter to the file(s) named by the OutName parameter. Note the following:
   2.1. If a source SORTEDBY variable is not named in the TableOutVars parameter, then the ReplaceVarPairs must be checked to see if a source variable is being replaced by a target file variable.
3. The second part of the macro develops the list of output filename(s) as follows:
   3.1. When SplitTime=N, then the output file name is set equal to the OutName parameter; e.g. DxExtract.
   3.2. When SplitTime=Y, then the output file names are created for each year, inclusive, using the OutName as prefix, followed by the 4-digit year value; e.g. DxExtract2007.
   3.3. When SplitTime=Q, then the output file names are created for each year and quarter, inclusive, using the OutName as prefix, followed by the 4-digit year value, a "Q", and the quarter number; e.g. DxExtract2007Q2.

3.4. If OutType=V, then the output filename is structure using single view syntax (e.g., OutputFileName / View=OutputFileNameV).

4. The third part of the macro is a single DATA STEP, that utilizes a hash join to filter incoming utilization rows, based on the patient identifier being found in a derivative of the demographic tables. The parameters for the macro call control how this data step is constructed, the file and variables read in, filtering logic, control of output variables, and whether a view or split files are created.
5. The macro accomplishes the following:
   5.1. Passes on the SORTEDBY variables names from the dataset named in the MSCDMTable parameter to the dataset(s) named in the OutName parameter.
   5.2. If SpiltTime is Y or Q, creates splits based on the calendar year or calendar quarter calculated on the basis of the variable named by the EventDate parameter.
   5.3. When ComputeAge=Y, creates both AgeTimePeriodKey and AgeEventKey surrogate keys for age:
      5.3.1. AgeTimePeriodKey is calculated as of the 1st date of a period (e.g., January 1 for yearly splits and 1st date of calendar quarter for quarter splits)
      5.3.2. AgeEventKey is calculated as of the EventDate.
   5.4. Uses variables named in the CovKeyVars parameter, to additionally filter rows from the utilization tables, determining if the EventDate falls in a year during which coverage is indicated.
      5.4.1. The variable(s) named by the CovKeyVars parameter are decimal integers and their binary value provides, per bit position, whether there is coverage within a calendar year.
      5.4.2. For any event row to be written to the output dataset(s), both MedCov and DrugCov are required.

C. CREATE SURROGATE KEYS FOR RX DRUGCLASS AND GENERICNAMES

The lookup/reference file for NDCs provides both a broad drug category/class as well as a generic name for each NDC. An individual NDC can fall into more than one drug category/class and more than one generic name. The purpose of this program is to improve processing performance. It takes an existing NDC reference tables and assigns unique surrogate key values for each drug category/class and for each generic name. These surrogate key values utilize many fewer bytes than do the full drug category/classes and generic names. When the program is run at MSOC it generates a PDF report of Drug Classes and Generic Names that share NDCs. This is the structure of the target table.

Variable	Data Type	Format	Label	Valid Values	Source/Comments
NDC	Char(11)	$11.	NDC	Unique numeric identifier of National Drug Codes	Uses original values
DrugClass	Char(70)	$70.	Drug Class	Free text	Uses original values
GenericName	Char(30)	$30.	Generic Name	Free text	Uses original values
DrugClassKey	Num(#)	#.	Drug Class Key	1+	Length of variable is driven by the maximum number of values found
GenericNameKey	Num(#)	#.	Generic Name Key	1+	Length of variable is driven by the maximum number of values found

Methods for the creating surrogate keys for rx drugclass and genericnames

1. Take the NDC lookup table and create a new table as a unique list for the following variables: NDC, DrugClass, and GenericName.
2. Take this new table and create two lists of the unique values for each of DrugClass and GenericName.
3. From the lists, determine the number of unique values for each of DrugClass and GenericName.
4. Determine the minimum length in bytes required to represent the maximum number of unique DrugClass and GenericName values, based on SAS platforms of UNIX and Windows. This table is an extract of the SAS LENGTH statement documentation:

Length in Bytes	Largest Integer Represented Exactly
3	8,192
4	2,097,152
5	536,870,912
6	137,438,953,472
7	35,184,372,088,832
8	9,007,199,254,740,992

1. Create key variables DrugClassKey and GenericNameKey respectively as follows:
   - 1.1. Read the list and assign a key value, starting at 1, up to the number of unique values, saving two files respectively that contain the following variables:
     - 1.1.1. DrugClassKeys: DrugClass and DrugClassKey.
     - 1.1.2. GenericNameKeys: GenericName and GenericNameKey.
2. Create intermediate tables:
   - 2.1. Read the unique NDC table from step 1 and link with the tables DrugClassKeys and GenericNameKeys, on DrugClass and GenericName variables respectively, saving the variables and structuring the final file as shown in the data dictionary above.
   - 2.2. Name the file **NDC_lookup_table_msoc_keyed**, sort by NDC, GenericName, and DrugClass, and save to the location for lookup tables used by the project. Note that an NDC can exist in multiple rows.
   - 2.3. Create an additional table, that is a 1:1 relationship of DrugClassKey and the DrugClass variables, named **DrugClassLookup**.
   - 2.4. Create an additional table, that is a 1:1 relationship of GenericNameKey and the GenericName variables, named **GenericNameLookup**.
3. When this program is run at MSOC, it generates a PDF report of Drug Classes and Generic Names that share NDCs.
   - 3.1. Take the NDC table from step 1 and for any NDC that has more than one value of DrugClass, create a table of distinct values of those DrugClasses. Get and save a count of the number of such DrugClasses.
   - 3.2. Take the NDC table from step 1 and for any NDC that has more than one value of GenericName, create a table of distinct values of those GenericNames. Get and save a count of the number of such GenericNames.
   - 3.3. Into a single PDF, print the list of DrugClasses and GenericNames. Insert into the respective titles, the following information:

3.3.1. Total number of DrugClass/GenericName
3.3.2. Number of DrugClasses/GenericNames that are shared by more than one NDC.
3.3.3. Percent calculated as (Value from Step 7.3.1 / Value from Step 7.3.2), formatted as ##.#%.

4. Create final table for use in a later hash join linking the DrugClassKeys and the GenericNameKeys to Dispensing rows. This table has the following structure:

Variable	Data Type	Format	Label	Valid Values	Source/Comments
NDC	Char(11)	$11.	NDC	Unique National Drug Codes	Uses original values
DrugClassKey1	Num(#)	#.	Drug Class Key	1+	Length of variable is driven by the maximum number of values found
DrugClassKey*n**	Num(#)	#.	Drug Class Key	1+	Length of variable is driven by the maximum number of values found
GenericNameKey1	Num(#)	#.	Generic Name Key	1+	Length of variable is driven by the maximum number of values found
GenericNameKey*n**	Num(#)	#.	Generic Name Key	1+	Length of variable is driven by the maximum number of values found

*The number of DrugClassKey and GenericNameKeys is dynamically created

1. From the **NDC_lookup_table_msoc_keyed** table, determine the maximum required number of DrugClassKey and GenericNameKey variables.
2. Read in the **NDC_lookup_table_msoc_keyed**, sorted by NDC. For each NDC,
   2.1. Output the DrugClassKeys and the GenericNameKeys to the final table
   2.2. Note that some values of DrugClassKey*n* and GenericNameKey*n* may be missing.

D. ATTACH RX SURROGATE KEYS TO DISPENSING ROWS

The **Rx_Attach_Keys** program links the DrugClass and GenericName keys to each NDC row in the Dispensing tables, utilizing a data step hash join. The result of this process is the following table(s), used for both prevalent and incident drug dispensing tables. Tables are named in this pattern: RxDrugClassYYYY and RxGenericNameYYYY, where "YYYY" is the calendar year.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
RxID	Num(#)	#.	RxID	1+	Unique ID per each row in Dispensing table
RxDate	Num(4)	mmddyy10.	RxDate	Dates with range of DP data	From the SCDM Dispensing table
RxSup	Num(4)	3.	RxSup	Days Supply	From the SCDM Dispensing table

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
AgeTimePeriodGroupKey	Char(1)	1.	AgeTimePeriodGroupKey	"0" – "9"	Key value mapping 1:1 with Age Groups, age calculated as of the beginning of the Year
AgeEventGroupKey	Char(1)	1.	AgeEventGroupKey	"0" – "9"	Key value mapping 1:1 with Age Groups, age calculated as of the RxDate
Sex	Char(1)	$1.	Sex	M or F	From the SCDM Demographic table
Period	Char(4)	$4.	Period	YYYY	4-digit year values from DP data
Quarter	Char(6)	$6.	Quarter	YYYYQ#	4-digit year values, "Q", quarter number as #, from DP data
AgeQuarterKey	Char(1)	1.	AgeQuarterKey	"0" – "9"	Key value mapping 1:1 with Age Groups, age calculated as of the beginning of the quarter
DrugClassKey*	Num(#)	#.	DrugClassKey	1+	Length of variable is driven by the maximum number of values found
GenericNameKey*	Num(#)	#.	GenericNameKey	1+	Length of variable is driven by the maximum number of values found

*Only one of these two variables will be included in the respective RxDrugClass or RxGenericName datasets.

1. For each year of DP Dispensing RxDates, perform the following:
   1.1. Create a dataset output name, for each of DrugClasses and GenericNames.
   1.2. Name the variables to be output in each dataset, including any SORTEDBY variables from the DP's Dispensing table.
2. Establish a data step hash join with one set of output datasets for DrugClasses and one set of output datasets for GenericNames. The data step should function in a single step to read all source Dispensing split files and output all required output dataset names as per step 1.1.
3. Output a single record to the **RxDrugClassYYYY** datasets for each unique combination of NDC and filled DrugClass.
4. Output a single record to the **RxGenericNameYYYY** datasets for each unique combination of NDC and filled GenericName.

IV. SUMMARY TABLES

A. AGE GROUPS TABLE

The age groups table provides a key for the age group stratifications within each summary table. This table provides a unique Age Group ID for one of the ten following age groups: ' 0-1', '2-4', '5- 9', '10-14', '15-18', '19-21', '22-44', '45-64', '65-74' and '75+'. This table is used to minimize the

complexity of the query created by the Sentinel Query Tool. The table is created as part of the distributed SAS code. The file will not change with each data refresh, but it must be held in the local summary table database at all times to enable the query process.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
ID	Num(3)	2.	ID	1+	
Strat10_name	Char(5)	$5.	10 Stratifications	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Strat10_sort_order	Num(3)	3.	Strat 10 Sort Order	10, 20, 30, 40, 50, 60, 70, 80, 90, 100	
Strat7_name	Char(5)	$5.	7 Stratifications	0-4, 5-9, 10-18, 19-21, 22-44, 45-64, 65+	
Strat7_sort_order	Num(3)	2.	Strat 7 Sort Order	10, 20, 30, 40, 50, 60, 70	
Strat4_name	Char(5)	$5.	4 Stratifications	0-21, 22-44, 45-64, 65+	
Strat4_sort_order	Num(3)	2.	Strat 4 Sort Order	10, 20, 30, 40	
Strat2_name	Char(8)	$8.	2 Stratifications	Under 65, 65+	
Strat2_sort_order	Num(3)	2.	Strat 2 Sort Order	10, 20	

Full Representation of Table

ID	Strat10_name	Strat10_sort_order	Strat7_name	Strat7_sort_order	Strat4_name	Strat4_sort_order	Strat2_name	Strat2_sort_order
1	0-1	10	0-4	10	0-21	10	Under 65	10
2	2-4	20	0-4	10	0-21	10	Under 65	10
3	5-9	30	5-9	20	0-21	10	Under 65	10
4	10-14	40	10-18	30	0-21	10	Under 65	10
5	15-18	50	10-18	30	0-21	10	Under 65	10
6	19-21	60	19-21	40	0-21	10	Under 65	10
7	22-44	70	22-44	50	22-44	20	Under 65	10
8	45-64	80	45-64	60	45-64	30	Under 65	10
9	65-74	90	65+	70	65+	40	65+	20
10	75+	100	65+	70	65+	40	65+	20

Methods for Creating **Age Groups Table**:

1. Create the structure for a table following the data dictionary at the top of this section.
2. Fill the 10 rows of the table with values shown above in the "Full Representation of Table." Note the relationship between a Strat##_name and a Strat##_sort_order. Also note the values of the ID variable in relationship to all other values across a row.

3. Name the table Age_Groups, sort by ID, and save to the DPLocal storage area.

B. ENROLLMENT SUMMARY TABLE

The enrollment table provides a count of unique members and days covered as defined below. The member count and days covered are stratified by age group, sex, year, quarter (note values in data dictionary for the Year variable), drug coverage status and medical coverage status. The count of unique members or days covered can be used as denominators to calculate crude prevalence rates.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	
Year	Char(6)	$6.	Year and Year/Quarter	YYYY=calendar year YYYYQ#, where: #= calendar quarter 1 through 4	Based on Enr_Start and Enr_End
DrugCov	Char(1)	$1.	Drug Coverage	Y, N	
MedCov	Char(1)	$1.	Medical Coverage	Y, N	
DaysCovered	Num(8)	15.	Days Covered	1+	Whole integers
Members	Num(8)	15.	# of Members	1+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating Enrollment Summary Table:

1. Create an extract of the SCDM Enrollment table, with variables PatID, Enr_Start, Enr_End, MedCov, and Drug_Cov. Keep only rows where the PatID is found in the PatKey table, DrugCov is either "Y" or "N" only, MedCov is either "Y" or "N" only, Enr_Start <= EnrEnd, and all extracted variables are not missing. Note that a PatID can be found in the Enrollment table more than once with distinct rows. Name this table Enr_Spans.
2. Fulfilling these requirements requires linking in demographic variables Birth_Date, and Sex with the Enr_Spans dataset, on the basis of PatKey and creating a table names EnrDemo.
3. For each row in EnrDemo, by patient, identify the different calendar years across their Enr_Start through Enr_End spans. Per PatKey, for each calendar year identified, create a new row and do the following:
   3.1. Set the value of Year to the calendar year identified.
   3.2. Calculate the number of days inclusive within the calendar year as *end-date* minus *start-date* + 1 and set this value to a temporary variable AllDays.. Note that the *start-date* through *end-date* relationship will be one of the following:
      3.2.1. January 1 through December 31

3.2.2. Enr_Start through December 31
3.2.3. January 1 through Enr_End
3.2.4. Enr_Start through Enr_End
3.3. Calculate AgeTimePeriodGroupKey as of January 1st of the Year for which the AllDays variable is being calculated.

4. Again for each row in EnrDemo, by patient, identify the different calendar quarters across their Enr_Start through Enr_End spans. Per PatID, for each calendar quarter identified, create a new row and do the following:
   4.1. Set the value of Year to the calendar quarter identified, as YYYYQ#.
   4.2. Calculate the number of days inclusive within the calendar quarter as *end-date* minus *start-date* + 1 and set this value to a temporary variable AllDaysQuarter. Note that the *start-date* through *end-date* relationship will be one of the following:
      4.2.1. 1st day in quarter through last day in quarter
      4.2.2. Enr_Start through last day in quarter
      4.2.3. 1st day in quarter through Enr_End
      4.2.4. Enr_Start through Enr_End
   4.3. Calculate AgeQuarterKey as of the first day in the quarter.
5. Aggregate rows from all of Step 3 across the classification variables of AgeTimePeriodGroupKey, Sex, Year, MedCov, DrugCov. For each combination of these variables, perform the following:
   5.1. Sum the number of AllDays and set DaysCovered to this value.
   5.2. Count the number of distinct PatIDs and set Members to this value.
6. Aggregate rows from all of Step 4, across the classification variables of AgeQuarterKey, Sex, Quarter, MedCov, and DrugCov. For each combination of these variables, perform the following:
   6.1. Sum the number of AllDaysQuarter and set DaysCovered to this value.
   6.2. Count the number of distinct PatIDs and set Members to this value.
7. Combine resulting aggregate rows from steps 5 and 6 into a single table. In doing so, with the aggregate from Step 6, rename AgeQuarterKey to AgeTimePeriodGroupKey and rename Quarter to Year.
8. Using the AgeTimePeriodGroupKey variable, create the variables Age_Group and Age_Group_ID.
9. Name the table **Enrollment** and structure table as per data dictionary above. Sort by Age_Group_ID, Year, Sex, MedCov, and DrugCov. Save to the DPLocal storage area.
10. Using the **Enr_Spans** table created earlier in Step 1, create a subset of only those spans, where both DrugCov ="Y" and MedCov = "Y". Per patient, join spans together where the number of days between an earlier Enr_End to the next Enr_Start is less than or equal to 45 days. Call this new table **Enr_FullCoverage_Spans_Bridged**. This will be used for the three incident tables below.

C. EXTRACTION AND SPLITTING OF DIAGNOSIS TABLE

1. Extract the SCDM Diagnosis table, enabling the following:

1.1. Filter for rows only where DXCode_Type="09", EncType is one of: AV, ED, IP, IS, or OA, and none of the following variables are missing: PatID, DX, DxCode_Type, EncType, and ADate, and the ADate is in between the Data Partner MSDD start and end dates, inclusive.
1.2. Ensure that the PatKey and year of ADate are found in the **PatKeyCovKey** table, with MedCov="Y" and DrugCov="Y", created above.
1.3. EncType gets renamed to Setting.
1.4. Any occurrences of decimal points in the Dx variable are removed.
1.5. Dx is renamed to Code.
1.6. Create surrogate key variables for age as of the event (i.e., ADate) and age as of the beginning of the Year (i.e., January 1st).

2. Enable the processing in Step 1 by calling the *TableExtract* macro with the following parameters:
2.1. MSCDMTable: Diagnosis
2.2. TableInVars: PatID Dx DxCode_Type ADate EncType
2.3. EventDate: ADate
2.4. FilterCode: A single expression to enable all of these conditions:
2.4.1. DX_CodeType="09"
2.4.2. EncType is one of AV, ED, IP, IS, or OA
2.4.3. None of these variables are missing: PatID, DX, EncType, and ADate
2.4.4. ADate is in between the Data Partner MSDD start and end dates, inclusive.
2.5. TransformCode: Code that executes each of the following:
2.5.1. Create a new character variable Period, that is equal to the year of ADate.
2.5.2. Recode any EncType value of OA to AV.
2.5.3. Recode any EncType value of IS to IP.
2.5.4. Remove any decimal points from the Dx variable
2.5.5. Rename Dx to Code
2.5.6. Rename EncType to Setting
2.6. ComputeAge: Y
2.7. DemHashTable: Demographic derivative table: PatKeysCovKey
2.8. HashVarKeys: PatID
2.9. HashVarsRead: PatKey, Birth_Date, and Sex
2.10. CovKeyVars:CovKeyDrugAndMed
2.11. OutName: Dx
2.12. OutType: V
2.13. TableOutVars: PatKey Code ADate Setting Period AgeEventGroupKey AgeTimePeriodGroupKey Sex
2.14. Lengths: Code $5 Setting $2
2.15. SplitTime: N
2.16. ReplaceVars: PatID|PatKey Dx|Code

The *TableExtract* macro will return two variables, AgeEventGroupKey and AgeTimePeriodGroupKey, that are surrogate keys for the patient's age as of the ADate and as of the beginning of the Time Period (e.g., Year, i.e. January 1), respectively.

3. Using the **Dx** view created using the *TableExtract* macro, create three new temporary tables as follows:
   - 3.1. **DX_3_Digit:** Keep all variables and extract only the left-most 3 characters of DX.
   - 3.2. **DX_4_Digit:** Keep all variables and extract only the left-most 4 characters of DX, but only when the 4th character is not blank.
   - 3.3. **DX_5_Digit:** Keep all variables and extract only the left-most 5 characters of DX, but only when the 4th and 5th characters are both not blank.
   - 3.4. Save all three files for use in creating other Summary Tables later in this document. The counts for the rows in the three tables should have this relationship: DX_3_Digit ≥ DX_4_Digit ≥ DX_5_Digit.

D. ICD-9-CM DIAGNOSIS SUMMARY TABLE (3 DIGIT)

The 3 digit ICD-9-CM diagnosis table provides a count of unique members with a diagnosis observed during the period and a count of events experienced within each stratum.

The counts are stratified by setting of visit (inpatient, outpatient, emergency department, any), age group, sex, year, and 3 digit ICD-9-CM code. Members are categorized into visit setting by the encounter type: **inpatient** includes acute inpatient hospital stay and non-acute institutional stays; **emergency department** includes emergency department encounters; **outpatient** includes ambulatory visit, telephone encounters, email encounters and other outpatient encounters. The **Any** setting includes the members with a visit in any of the care settings; for example, if a member has the same diagnosis code observed across multiple care settings during a period, the member will be counted once in the member count and all the visits with the code will be summed for the event counts.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	Based on calculating age
Sex	Char(1)	$1.	Sex	M or F	From Demographic Table
Period	Char(4)	$4.	Year	2004, 2005, etc.	Based on ADate
Code	Char(3)	$3.	3-digit Dx Code	Free text	
DxName	Char(35)	$35.	Diagnosis Name	Free text	Based on lookup table values
Setting	Char(2)	$2.	Setting	AN = Any (AV, ED, IP, IS, or OA) AV = Outpatient (AV or OA) ED = Emergency department IP = Inpatient (IP or IS)	
Members	Num(8)	15.	# of Members	1+	Whole integers
Events	Num(8)	15.	# of Events	1+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating ICD-9-CM Diagnosis Summary Table (3 digit):

1. Use the **DX_3_Digit** temporary table created in the **Extraction and Splitting of Diagnosis Table** process.
2. Aggregate the rows across the classification variables of AgeTimePeriodGroupKey, Sex, Period, and Code. For each combination of these variables, perform the following:
   2.1. Set Setting = "AN".
   2.2. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   2.3. Count the number of rows and set Events to this value.
3. Aggregate the temporary **DX_3_Digit** table again, across the classification variables of AgeTimePeriodGroupKey, Sex, Period, Code and Setting. For each combination of these variables, perform the following:
   3.1. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   3.2. Count the number of rows and set Events to this value.
4. Combine all of the rows from Steps 2 and 3 into a single table, with corresponding variables.
5. Using the AgeTimePeriodGroupKey variable, create the variables Age_Group and Age_Group_ID.
6. Link the resulting table with the Dx_ICD9_3dig_Lookup table, on the basis of Lookup.Code = DX_3_Digit.Code to get the Lookup.Srt_Descrip variable, renaming this variable to DxName. Keep only rows where the linking identifies a DxName for the Code.
7. Name the table **ICD9_Diagnosis** and structure table as per data dictionary above. Sort by Age_Group_ID, Sex, Period, Code, and Setting. Save to the DPLocal storage area.

E. ICD-9-CM DIAGNOSIS SUMMARY TABLE (4 DIGIT)

The 4 digit ICD-9-CM diagnosis table provides a count of unique members with a diagnosis observed during the period and a count of events experienced within each stratum. The counts are stratified by setting of visit as described above (3-digit diagnosis summary tables). This table is identical to the ICD-9-CDM Diagnosis Summary Table (3 digit) except for use of 4-digit clinical codes instead of 3-digit codes.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	Based on calculating age
Sex	Char(1)	$1.	Sex	M or F	From Demographic Table
Period	Char(4)	$4.	Year	2004, 2005, etc.	Based on ADate
Code	Char(4)	$4.	4-digit Dx Code	Free text	
DxName	Char(35)	$35.	Diagnosis Name	Free text	

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Setting	Char(2)	$2.	Setting	AN = Any (AV, ED, IP, IS, or OA) AV = Outpatient (AV or OA) ED = Emergency department IP = Inpatient (IP or IS)	
Members	Num(8)	15.	# of Members	1+	Whole integers
Events	Num(8)	15.	# of Events	1+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating ICD-9-CM Diagnosis Summary Table (4 digit):

1. Use the **DX_4_Digit** temporary table created in the **Extraction and Splitting of Diagnosis Table** process.
2. Aggregate the rows across the classification variables of AgeTimePeriodGroupKey, Sex, Period, and Code. For each combination of these variables, perform the following:
   2.1. Set Setting = "AN".
   2.2. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   2.3. Count the number of rows and set Events to this value.
3. Aggregate the resulting rows again across the classification variables of AgeTimePeriodGroupKey, Sex, Period, Code, and Setting. For each combination of these variables, perform the following:
   3.1. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   3.2. Count the number of rows and set Events to this value.
4. Combine the rows from Steps 2 and 3 into a single table.
5. Using the AgeTimePeriodGroupKey variable, create the variables Age_Group and Age_Group_ID.
6. Link the resulting table with the Dx_ICD9_4dig_Lookup table, on the basis of Lookup.Code = DX_4_Digit.Code to get the Lookup.Srt_Descrip variable, renaming this variable to DxName. Keep only rows where the linking identifies a DxName for the Code.
7. Name the table **ICD9_Diagnosis_4_Digit** and structure table as per data dictionary above. Sort by Age_Group_ID, Sex, Period, Code, and Setting. Save to the DPLocal storage area.

F. ICD-9-CM DIAGNOSIS SUMMARY TABLE (5 DIGIT)

The 5 digit ICD-9-CM diagnosis table provides a count of unique members with a diagnosis observed during the period and a count of events experienced within each stratum. The counts are stratified by setting of visit as described above (3-digit diagnosis summary tables). This table is identical to the ICD-9-CDM Diagnosis Summary Table (3 digit) except for use of 5-digit clinical codes instead of 3-digit codes.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	
Period	Char(4)	$4.	Year	2004, 2005, etc.	Based on ADate
Code	Char(5)	$5.	5-digit Dx Code		
DxName	Char(35)	$35.	Diagnosis Name		
Setting	Char(2)	$2.	Setting	AN = Any (AV, ED, IP, IS, or OA) AV = Outpatient (AV or OA) ED = Emergency department IP = Inpatient (IP or IS)	
Members	Num(8)	15.	# of Members	1+	Whole integers
Events	Num(8)	15.	# of Events	1+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating ICD-9-CM Diagnosis Summary Table (5 digit):

1. Use the **DX_5_Digit** temporary table created in the **Extraction and Splitting of Diagnosis Table** process.
2. Aggregate the rows across the classification variables of AgeTimePeriodGroupKey, Sex, Period, and Code. For each combination of these variables, perform the following:
   2.1. Set Setting = “AN”.
   2.2. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   2.3. Count the number of rows and set Events to this value.
3. Aggregate the resulting rows again, across the classification variables of AgeTimePeriodGroupKey, Sex, Period, Code, and Setting. For each combination of these variables, perform the following:
   3.1. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   3.2. Count the number of rows and set Events to this value.
4. Combine the rows from Steps 2 and 3 into a single table.
5. Using the AgeTimePeriodGroupKey variable, create the variables Age_Group and Age_Group_ID.

6. Link the resulting table with the Dx_ICD9_5dig_Lookup table, by joining Lookup.Code = DX_5_Digit.Code to get the Lookup.Srt_Descrip variable, renaming this variable to DxName. Keep only rows where the linking identifies a DxName for the Code.
7. Name the table **ICD9_Diagnosis_5_Digit** and structure table as per data dictionary above. Sort by Age_Group_ID, Sex, Period, Code, and Setting. Save to the DPLocal storage area.

G. EXTRACTION AND SPLITTING OF PROCEDURE TABLE

1. Extract the SCDM Procedure table, enabling the following:
   - 1.1. Filter for rows only where PxCode_Type is one of "C4", "09", or "HC", EncType is one of: AV, ED, IP, IS, or OA, and none of the following variables are missing: PatID, PX, PxCode_Type, EncType, and ADate, and the ADate is in between the Data Partner MSDD start and end dates, inclusive.
   - 1.2. Ensure that the PatKey and year of ADate are found in the PatKeyCovKey table, with MedCov="Y" and DrugCov="Y", created above.
   - 1.3. EncType gets renamed to Setting.
   - 1.4. Any occurrences of decimal points in the Px variable are removed.
   - 1.5. Px is renamed to Code.
   - 1.6. Create surrogate key variables for age as of the event (i.e., ADate) and age as of the beginning of the Year (i.e., January 1st).
2. To fulfill these requirements, call the *TableExtract* macro, using the following parameters:
   - 2.1. MSCDMTable: Procedure
   - 2.2. TableInVars: PatID Px PxCode_Type ADate EncType
   - 2.3. EventDate: ADate
   - 2.4. FilterCode: A single expression to enable all of these conditions:
     - 2.4.1. PxCode_Type is one of "C4", "09" or "HC" only
     - 2.4.2. EncType is one of AV, ED, IP, IS, or OA
     - 2.4.3. None of the following variables are missing: PatID, PX, PxCode_Type EncType, and ADate
     - 2.4.4. ADate is in between the Data Partner MSDD start and end dates, inclusive.
   - 2.5. TransformCode: Code that executes each of the following:
     - 2.5.1. Create a new character variable Period, that is equal to the year of ADate.
     - 2.5.2. Recode any EncType value of OA to AV.
     - 2.5.3. Recode any EncType value of IS to IP.
     - 2.5.4. Remove any decimal points from the Px variable
     - 2.5.5. Rename Px to Code
     - 2.5.6. Rename EncType to Setting
   - 2.6. ComputeAge: Y
   - 2.7. DemHashTable: Demographic derivative table: PatKeysCovKey

2.8. HashVarKeys: PatID
2.9. HashVarsRead: PatKey, Birth_Date, and Sex
2.10. CovKeyVars: CovKeyDrugAndMed
2.11. OutName: Px
2.12. OutType: V
2.13. TableOutVars: PatKey Code PxCode_Type Setting Period AgeTimePeriodGroupKey Sex
2.14. Lengths: Code $5 Setting $2
2.15. SplitTime: N
2.16. ReplaceVars: PatID|PatKey Px|Code

The *TableExtract* macro will return two variables, AgeEventGroupKey and AgeTimePeriodGroupKey, that are surrogate keys for the patient's age as of the date of the ADate and as of the beginning of the Time Period (i.e., January 1), respectively.

3. Using the returned **Px** view, create three new temporary tables as follows:
   3.1. **HCPCS:** Subset for PX_CodeType="HC" or "C4".
   3.2. **PX_ICD9_3_Digit:** Subset for PX_CodeType ="09". Extract only the left-most 3 characters of PX, recoding Code to this value.
   3.3. **PX_ICD9_4_Digit:** Subset for PX_CodeType ="09". Extract only the left-most 4 characters of PX, but only when the 4th character is not blank, recoding Code to this value.
   3.4. Save all three tables for use in creating other Summary Tables later in this document. The count of rows for the two PX_ICD9 tables should have this relationship: PX_3_Digit ≥ PX_4_Digit.

H. HCPCS SUMMARY TABLE

The HCPCS table provides a count of unique members who had a procedure observed during the period and a count of events experienced within each stratum. Although this is called the "HCPCS" Summary Table, it includes both CPT (Common Procedural Terminology) and HCPCS (Healthcare Common Procedure Coding System) codes.

The counts are stratified by setting of visit (as defined for the Diagnosis tables above), age group, sex, year, and procedure code.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	
Period	Char(4)	$4.	Year	2004, 2005, etc.	Based on ADate
PX_Code	Char(5)	$5.	CPT or HCPCS Code	Free text	
PxName	Char(35)	$35.	Procedure Name	Free text	

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Setting	Char(2)	$2.	Setting	AN = Any (AV, ED, IP, IS, or OA) AV = Outpatient (AV or OA) ED = Emergency department IP = Inpatient (IP or IS)	
Members	Num(8)	15.	# of Members	1+	Whole integers
Events	Num(8)	15.	# of Events	1+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating **HCPCS Summary Table**:

1. Use the **HCPCS** temporary table created in the **Extraction and Splitting of Procedure Table** process.
2. Aggregate the rows across the classification variables of Age_Group, Sex, Period, and Code. For each combination of these variables, perform the following:
   2.1. Set Setting = "AN".
   2.2. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   2.3. Count the number of rows and set Events to this value.
3. Aggregate the temporary **HCPCS** table again, across the classification variables of Age_Group, Sex, Period, Code, and Setting. For each combination of these variables, perform the following:
   3.1. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   3.2. Count the number of rows and set Events to this value.
4. Combine the rows from Steps 2 and 3 into a single table.
5. Using the AgeTimePeriodGroupKey variable, create the variables Age_Group and Age_Group_ID.
6. Link the resulting table with the Px_Lookup table on the basis of Lookup.Code = HCPCS.Code and either: (1) Lookup.source="hcpcs" and HCPCS.PX_CodeType="HC" or (2) Lookup.source="cpt" and HCPCS.PX_CodeType="C4", to get the Lookup.Srt_Descrip variable, renaming this variable to PxName. Keep only rows where the linking identifies a PxName for the Code.
7. Name the table **HCPCS** and structure the table as per data dictionary above. (Note that unlike all other prevalent Diagnosis and Procedure tables, the HCPCS table and ICD9_Procedure_4_Digit table have the variable named PX_Code instead of Code.) Sort by Age_Group_ID, Sex, Period, Code, and Setting. Save to the DPLocal storage area.

I. ICD-9-CM PROCEDURE SUMMARY TABLE (3 DIGIT)

The ICD-9-CM 3 digit procedure table provides a count of unique members with an ICD-9-CM coded procedure observed during the period and a count of events in each stratum.

The counts are stratified by setting of visit (as defined above), age group, sex, year, and 3 digit ICD-9-CM procedure code.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	
Period	Char(4)	$4.	Year	2004, 2005, etc.	Based on ADate
Code	Char(3)	$3.	3-digit ICD9 Px Code	Free text	
PxName	Char(35)	$35.	Procedure Name	Free text	
Setting	Char(2)	$2.	Setting	AN = Any (AV, ED, IP, IS, or OA) AV = Outpatient (AV or OA) ED = Emergency department IP = Inpatient (IP or IS)	
Members	Num(8)	15.	# of Members	1+	Whole integers
Events	Num(8)	15.	# of Events	1+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating ICD-9-CM Procedure Summary Table (3 digit):

1. Use the **PX_ICD9_3_Digit** temporary table created in the **Extraction and Splitting of Procedure Table** process.
2. Aggregate the rows across the classification variables of Age_Group, Sex, Period, and Code. For each combination of these variables, perform the following:
   2.1. Set Setting = “AN”.
   2.2. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   2.3. Count the number of rows and set Events to this value.
3. Aggregate the rows again across the classification variables of Age_Group, Sex, Period, Code, and Setting. For each combination of these variables, perform the following:
   3.1. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   3.2. Count the number of rows and set Events to this value.
4. Combine the rows from Steps 2 and 3 into a single table.
5. Using the AgeTimePeriodGroupKey variable, create the variables Age_Group and Age_Group_ID.

6. Link the resulting table with the Px_ICD9_3Dig_Lookup table, by joining Lookup.Code = PX_ICD9_3_Digit.Code to get the Lookup.Srt_Descrip variable, renaming this variable to PxName. Keep only rows where the linking identifies a PxName for the Code.
7. Name the table **ICD9_Procedure** and structure table as per data dictionary above. Sort by Age_Group_ID, Sex, Period, Code, and Setting. Save to the DPLocal storage area.

J. ICD-9-CM PROCEDURE SUMMARY TABLE (4 DIGIT)

The ICD-9-CM 4 digit procedure table provides a count of unique members with an ICD-9-CM coded procedure observed during the period and a count of events in each stratum.

The counts are stratified by setting of visit (as defined above), age group, sex, year, and 4 digit ICD-9-CM procedure code.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	
Period	Char(4)	$4.	Year	2004, 2005, etc.	Based on ADate
PX_Code	Char(4)	$4.	4-digit ICD9 Px Code	Free text	
PxName	Char(35)	$35.	Procedure Name	Free text	
Setting	Char(2)	$2.	Setting	AN = Any (AV, ED, IP, IS, or OA) AV = Outpatient (AV or OA) ED = Emergency department IP = Inpatient (IP or IS)	
Members	Num(8)	15.	# of Members	1+	Whole integers
Events	Num(8)	15.	# of Events	1+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating ICD-9-CM Procedure Summary Table (4 digit):

1. Use the **PX_ICD9_4_Digit** temporary table created in the **Extraction and Splitting of Procedure Table** process.
2. Aggregate the rows across the classification variables of Age_Group, Sex, Period, and Code. For each combination of these variables, perform the following:
   2.1. Set Setting = “AN”.
   2.2. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   2.3. Count the number of rows and set Events to this value.

3. Aggregate the rows again across the classification variables of Age_Group, Sex, Period, Code, and Setting. For each combination of these variables, perform the following:
   3.1. Count the number of distinct patients (i.e., PatKey) and set Members to this value.
   3.2. Count the number of rows and set Events to this value.
4. Combine the rows from Steps 2 and 3 into a single table.
5. Using the AgeTimePeriodGroupKey variable, create the variables Age_Group and Age_Group_ID.
6. Link the resulting table with the Px_ICD9_4Dig_Lookup table, on the basis of Lookup.Code = PX_ICD9_4_Digit.Code to get the Lookup.Srt_Descrip variable, renaming this variable to PxName. Keep only rows where the linking identifies a PxName for the Code.
7. Name the table **ICD9_Procedure_4_Digit** and structure the table as per data dictionary above. (Note that unlike all other prevalent Diagnosis and Procedure tables, the ICD9_Procedure_4_Digit table and the HCPCS table have the variable named PX_Code instead of Code.) Sort by Age_Group_ID, Sex, Period, Px_Code, and Setting. Save to the DPLocal storage area.

K. EXTRACTION AND KEY ASSIGNMENT OF DISPENSING TABLE

1. Create an extract from the SCDM Dispensing table, keeping the following variables in the resulting file: PatID, RxDate, NDC, RxSup, and RxAmt. Keep only rows that meet these conditions and process as indicated:
   1.1. All variables are not missing.
   1.2. Ensure that the PatKey and year of RxDate are found in the PatKeyCovKey table, with MedCov="Y" and DrugCov="Y", created above.
   1.3. RxDate is in between the Data Partner MSDD start and end dates, inclusive.
   1.4. RxSup is greater than or equal to 1, thus excluding rows with missing or zero values.
   1.5. NDC is 11 digits only.
   1.6. Create the Period variable as the calendar year and the Quarter variable of the RxDate, in the format YYYYQ#.
2. To fulfill these requirements, call the *TableExtract* macro, using the following parameters:
   2.1. MSCDMTable: Dispensing
   2.2. TableInVars: PatID RxDate NDC RxSup RxAmt
   2.3. EventDate: RxDate
   2.4. FilterCode: A single expression to enable all of these conditions:
      2.4.1. RxSup >= 1
      2.4.2. None of these variables are missing: PatID, RxDate NDC RxSup RxAmt
   2.5. TransformCode: Code that executes each of the following:
      2.5.1. Create a new character variable Period, that is equal to the year of RxDate
      2.5.2. Create a new character variable Quarter, that is equal to the calendar quarter of RxDate, formatted as YearQ#.
      2.5.3. Calculates AgeQuarterKey variable, as of the beginning of the Quarter, identified in step 2.5.2.
   2.6. ComputeAge: Y

2.7. DemHashTable: Demographic derivative table: PatKeysCovKey
2.8. HashVarKeys: PatID
2.9. HashVarsRead: PatKey, Birth_Date, and Sex
2.10. CovKeyVars: CovKeyDrugAndMed
2.11. OutName: Rx
2.12. OutType: V
2.13. Lengths: Quarter $6 RxID (# of bytes needed to hold the # of observations in the CDM Dispensing table)
2.14. TableOutVars: PatKey Sex RxDate NDC RxSup RxID AgeEventGroupKey AgeTimePeriodGroupKey Period Quarter AgeQuarterKey
2.15. SplitTime: N
2.16. ReplaceVars: PatID|PatKey

3. Perform the process **Attach Rx Surrogate Keys to Dispensing Rows** (above) to link the NDCs with their respective DrugClassKey(s) and GenericNameKey(s), resulting in files named in the pattern of RxDrugClassYYYY and a set of files named in the pattern of RxGenericNameYYYY.

L. DRUG CATEGORY/CLASS SUMMARY TABLE

The Drug Category (also known as Drug Class) table provides a count of unique members who had a drug dispensing during the period and a count of dispensings received by all of these members by strata. Additionally, a count of total days supply (sum of days supply for all members by strata) is included.

The counts are stratified by drug class, age group, sex, year-quarter and year. The drug category is standardized using a look-up table provided by the Sentinel Operations Center.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	
Period	Char(6)	$6.	Year and Year/Quarter	YYYY=calendar year YYYYQ#, where: #= calendar quarter 1 through 4	
DrugClass	Char(70)	$70.	Drug Class/Category	Free text	
Members	Num(8)	15.	# of Members	1+	Whole integers
Dispensings	Num(8)	15.	# of Dispensings	1+	Whole integers
DaysSupply	Num(8)	15.	Days Supply	1+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating Drug Category/Class Summary Table:

1. Use the file(s) RxDrugClassYYYY, created by the process of **Extraction and Key Assignment of Dispensing Table** above, Step 3.
2. Aggregate the file(s), per AgeTimePeriodGroupKey, Sex, Period, and DrugClassKey, as follows:
   2.1. Count the number of distinct PatKeys and set the result value to variable Members.
   2.2. Count the number of rows and set the result value to variable Dispensings.
   2.3. Sum the values of RxSup and set the result value to variable DaysSupply.
   2.4. Name the aggregate file as **RxDrugClassYYYY**. It should contain the following variables: AgeTimePeriodGroupKey, Sex, Period, DrugClassKey, Members, Dispensings, and DaysSupply.
3. Aggregate the resulting file(s) from Step 3 again, per AgeQuarterKey, Sex, Quarter, and DrugClassKey, as follows:
   3.1. Count the number of distinct PatKeys and set the result value to variable Members.
   3.2. Count the number of rows and set the result to variable Dispensings.
   3.3. Sum the values of RxSup and set the result value to variable DaysSupply.
   3.4. Name the aggregate file as **RxDrugClassQuarterYYYY**. It should contain the following variables: AgeQuarterKey, Sex, Quarter, DrugClassKey, Members, Dispensings, and DaysSupply.
4. Combine all rows from **RxDrugClassYYYY** and **RxDrugClassQuarterYYYY**. In doing so with **RxDrugClassQuarterYYYY**, rename variable Quarter to Period and AgeQuarterKey to AgeTimePeriodGroupKey.
5. Link the resulting table again with the NDC_Lookup_Table on the basis of Lookup.DrugClassKey = RxAggregates.DrugClassKey to get the DrugClass variable. Drop variable DrugClassKey from further processing.
6. Using the AgeTimePeriodGroupKey variable, create the variables Age_Group and Age_Group_ID.
7. Name the table **Drug_Class** and structure table as per data dictionary above. Sort by Age_Group_ID, Sex, Period, and DrugClass. Save to the DPLocal storage area.

M. INGREDIENT/GENERIC NAME SUMMARY TABLE

The ingredient name (also known as Generic Name) table provides a count of unique members who had a drug dispensing during the period, a count of dispensings received by all of these members, and total days supplied by strata.

The counts are stratified by drug class, age group, sex, year-quarter and year. The drug category is standardized using a look-up table provided by the Sentinel Operations Center.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Period	Char(6)	$6.	Year and Year/Quarter	YYYYQ#, where: YYYY=calendar year #= calendar quarter 1 through 4	
GenericName	Char(30)	$30.	Generic Drug Name	Free text	
Members	Num(8)	15.	# of Members	1+	Whole integers
Dispensings	Num(8)	15.	# of Events	1+	Whole integers
DaysSupply	Num(8)	15.	Days Supply	1+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating Ingredient Name Summary Table:

1. Use the file(s) RxGenericNameYYYY, created by the process of **Extraction and Key Assignment of Dispensing Table** above, Step 3.
2. Aggregate the resulting file(s), per AgeTimePeriodGroupKey, Sex, Period, and DrugClassKey, as follows:
   - 2.1. Count the number of distinct PatKeys and set the result value to variable Members.
   - 2.2. Count the number of rows and set the result value to variable Dispensings.
   - 2.3. Sum the values of RxSup and set the result value to variable DaysSupply.
   - 2.4. Name the aggregate file as **RxGenericNameYYYY**. It should contain the following variables: Age_Group, Sex, Period, GenericNameKey, Members, Dispensings, and DaysSupply.
3. Aggregate the resulting file(s) from Step 1 again, per AgeQuarterKey, Sex, Quarter, and GenericNameKey, as follows:
   - 3.1. Count the number of distinct PatKeys and set the result value to variable Members.
   - 3.2. Count the number of rows and set the result value to variable Dispensings.
   - 3.3. Sum the values of RxSup and set the result value to variable DaysSupply.
   - 3.4. Name the aggregate file as **RxGenericNameQuarterYYYY**. It should contain the following variables: AgeQuarterKey, Sex, Quarter, GenericNameKey, Members, Dispensings, and DaysSupply.
4. Combine all rows from **RxGenericNameYYYY** and **RxGenericNameQuarterYYYY**. In doing so with **RxGenericNameQuarterYYYY**, rename variable Quarter to Period and AgeQuarterKey to AgeTimePeriodGroupKey.
5. Link the resulting table again with the NDC_Lookup_Table on the basis of Lookup.GenericNameKey = RxAggregates.GenericNameKey to get the GenericName variable. Drop variable GenericNameKey from further processing.
6. Using the AgeTimePeriodGroupKey variable, create the variables Age_Group and Age_Group_ID.
7. Name the table **Generic_Name** and structure table as per data dictionary above. Sort by Age_Group_ID, Sex, Period, and GenericName. Save to the DPLocal storage area.

N. INCIDENT ICD-9-CM DIAGNOSIS SUMMARY TABLE (3 DIGIT)

The incident ICD-9-CM diagnosis table provides a count of unique members and unique incident diagnosis events of each 3-digit ICD-9-CM category in one of four care settings of interest (i.e., inpatient, emergency department, ambulatory, and any) stratified by age group, sex, and year.

An incident event is defined as a member with an encounter with the diagnosis of interest (i.e., the index date), in the care setting of interest, in the year of interest, with *no evidence* of that diagnosis in the 90, 180 and 270 days (i.e., the lookback periods) before the index date in any care setting. Both continuous medical and drug coverage are required during the 3 possible lookback periods, through the date of the incident events. When defining continuous coverage, enrollment gaps of <=45 days are bridged.

If a patient has more than one qualifying incident event within a calendar year, for a given diagnosis care setting, and lookback period, all incident events qualifying will be reported, although incident members will be counted at most once within a calendar year for a given diagnosis and care setting. Note that while reporting is by calendar year, lookback periods can extend to the prior year to ascertain incidence.

Counts are stratified by setting of visit, age group, sex, year, 3 digit ICD-9-CM code(s) of interest. For each stratum the number of unique members and incident events for the 90, 180 and 270 lookback scenarios are reported.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	
Period	Char(4)	$4.	Year		
Code	Char(3)	$3.	3-digit Dx Code		
DxName	Char(35)	$35.	Diagnosis Name		
Setting	Char(2)	$2.	Setting	AN = Any (AV, ED, IP, IS, or OA) AV = Outpatient (AV or OA) ED = Emergency department IP = Inpatient (IP or IS)	
Members90	Num(8)	15.	Members in 90-day lookback	1+	Whole integers
Events90	Num(8)	15.	# of Events from 90-day lookback	1+	Whole integers
Members180	Num(8)	15.	Members in 180-day lookback	0+	Whole integers
Events180	Num(8)	15.	# of Events from 180-day lookback	0+	Whole integers
Members270	Num(8)	15.	Members in 270-day lookback	0+	Whole integers
Events270	Num(8)	15.	# of Events from 270-day lookback	0+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating Incident ICD-9-CM Diagnosis Summary Table (3 Digit)

1. Use the **DX_3_Digit** temporary table(3) created in the **Extraction and Splitting of Diagnosis Table** process. Drop the AgeTimePeriodGroupKey variable.
2. Merge the table from Step 1 with the **Enr_FullCoverage_Spans_Bridged** table (built as part of creating the Enrollment Summary Table, Step 10) on the basis of PatKey and keep only those rows from the Step 1 table(s), where the ADate is in between the patient's Enr_Start and Enr_End dates. Note that per patient, there can be multiple spans of Enr_Start to Enr_End, each of which should be checked. Sort this table by PatKey, Code, and ADate. Keep the following variables: PatKey, AgeEventGroupKey, Sex, Setting, Code, and ADate.
3. Then build a patient level temporary table, named **DX_3_Digit_All,** that will have the following structure. Follow Step 4 to create the table.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
PatKey	Num(#)	#.	PatKey	Numeric key on a 1:1 relationship with PatID	Surrogate key for the PatID
AgeEventGroupKey	Char(1)	1.	AgeEventGroupKey	"0" – "9"	Key value mapping 1:1 with Age Groups, age calculated as of the ADate
Sex	Char(1)	$1.	Sex	M or F	
Setting	Char(2)	$2.	Setting	AV = Outpatient (AV or OA) ED = Emergency department IP = Inpatient (IP or IS)	
Code	Char(3)	$3.	Code	ICD-9 3-digiti diagnosis codes	
ADate	Num(4)	mmddyy10.	ADate	Valid dates	
Inc90	Num(1)	1.	Inc90	0 = Not incident 90 days prior 1 = Incident 90 days prior	
Inc180	Num(1)	1.	Inc180	0 = Not incident 180 days prior 1 = Incident 180 days prior	
Inc270	Num(1)	1.	Inc270	0 = Not incident 270 days prior 1 = Incident 270 days prior	

4. For each row (defined by PatKey, Code, and ADate) in the new **DX_3_Digit_All** table, initialize each of these variables to zero: Inc90, Inc180, and Inc270. Then read rows from the Step 2 table. For each ADate, look earlier within all rows, across all Settings, for the same Code.
   - 4.1. When no same Code is found within 90 days earlier, not including the examined ADate, and the examined ADate is more than 90 days later than the earliest Data Partner MSDD minimum date, set the value of Inc90 to 1.
   - 4.2. When no same Code is found within 180 days earlier, not including the examined ADate, and the examined ADate is more than 180 days later than the earliest Data Partner MSDD minimum date, set the value of Inc180 to 1.
   - 4.3. When no same Code is found within 270 days earlier, not including the examined ADate, and the examined ADate is more than 270 days later than the earliest Data Partner MSDD minimum date, set the value of Inc270 to 1.

5. Aggregate for each Setting: Using the **DX_3_Digit_All** file as input, creates three groups of subsets (for each value of 90, 180, and 270), for each distinct Period, where Period equals the year-value of ADate, and including rows where at least one Inc### = 1. For each subset (i.e., Period by ###), on the classification combination of Code, AgeEventGroupKey, Sex, and Setting, create aggregates as follows:
    5.1. Count the number of unique PatKeys and save this value to Members###.
    5.2. Sum the Inc### and save this value to Events###.
6. Aggregate for "Any" Setting: Use the same subsets as in Step 5, and for each subset (i.e., Period by ###), on the classification combination of Code, AgeEventGroupKey, and Sex, create aggregates as follows:
    6.1. Count the number of unique PatKeys and save this value to Members###.
    6.2. Sum the Inc### and save this value to Events###.
    6.3. Set Setting to "AN".
7. Merge all aggregates from Steps 5 and 6 on the linking variables AgeEventGroupKey, Sex, Period, Code, and Setting and write to file **IncDx3**. Set any missing values to zero, for variables Members90, Events90, Members180, Events180, Members270, or Events270.
8. Using the AgeEventGroupKey variable, create the variables Age_Group and Age_Group_ID.
9. Then link this file with Dx_ICD9_3dig_Lookup, on IncDx3.Code = Lookup.Code, to get the Lookup.Srt_descrip variable, renaming this variable to DxName.
10. Name the final table **Incident_ICD9_Diagnosis** and structure table as per data dictionary above. Sort by Age_Group_ID, Sex, Period, Code and Setting. Save to the DPLocal storage area.

O. INCIDENT DRUG CLASS/CATEGORY SUMMARY TABLE

The incident Drug Class table provides a count of unique members with an incident dispensing for each drug category (e.g., betablocker, antidiabetic) of interest stratified by age group, sex, and year.

Incidence is defined as a member with a dispensing with the Drug Class of interest (i.e., the index date), in the year of interest with no evidence of a dispensing for that drug category in the 90, 180 and 270 days (i.e., the lookback periods) before the index date.

Both medical and drug coverage are required during the 3 possible lookback periods, allowing for eligibility gaps of <=45 days. Additionally, both forms of enrollment coverage are required throughout the period of the dispensing episode. Following rules in Cohort Identification and Data Analysis (CIDA), only dispensings with the start of their dispensing within enrollment coverage are utilized. That is, the start of the dispensing must have full enrollment coverage, both before and after stockpiling. After stockpiling, if the start of a dispensing has been changed, it is the new start date that must be fully covered by enrollment. Stockpiling is the process of resetting to a later start date, those dispensing rows whose original start date overlapped prior dispensings, using the assumption that a patient will consume all doses in all filled prescriptions.

In addition to reporting the number of members with an incident dispensing, for each such incident user a treatment episode starting on the index date is created, and the total number of dispensings, days supplied and length of treatment episodes (in days) is measured for each treatment episode. Treatment gaps of <= 15 days are allowed when creating episodes and are considered part of the same treatment episode. Treatment

episodes are censored by a gap in treatment, a gap in enrollment, or the end of Data Partner data, whichever occurs first. Although a member can have multiple index events in a given calendar year the first one only is counted and used for reporting.

The counts are stratified by Drug Class, age group, sex, and year. Drug Categories are standardized using a look-up table provided by the Sentinel Operations Center. For each stratum the results contain 3 separate sections for each of the 90, 180 and 270 lookback scenarios. Each section contains the total number of members, total dispensings, total days supplied and total length of all episodes, as well as a quarterly breakdown of index dates (must sum up to total number of members).

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	
Period	Char(4)	$4.	Year	2000+	
DrugClass	Char(70)	$70.	Drug Class/Category Name	Text description	
Members90	Num(8)	15.	Members with no same drug dispensing in 90-day lookback prior to index date	1+	Whole integers
Dispensings90	Num(8)	15.	# of Dispensings for 90-day lookback	1+	Whole integers
DaysSupply90	Num(8)	15.	# of total days supply for all episodes for 90-day lookback	1+	Whole integers
EpisodeSpan90	Num(8)	15.	# of total length in days of all episodes for 90-day lookback	1+	Whole integers
Members90Q1	Num(8)	15.	# of index dates in Period/Q1 for 90-day lookback	0+	Whole integers
Members90Q2	Num(8)	15.	# of index dates in Period/Q2 for 90-day lookback	0+	Whole integers
Members90Q3	Num(8)	15.	# of index dates in Period/Q3 for 90-day lookback	0+	Whole integers
Members90Q4	Num(8)	15.	# of index dates in Period/Q4 for 90-day lookback	0+	Whole integers
Members180	Num(8)	15.	Members with no same drug dispensing in 180-day lookback prior to index date	0+	Whole integers
Dispensings180	Num(8)	15.	# of Dispensings for 180-day lookback	0+	Whole integers
DaysSupply180	Num(8)	15.	# of total days supply for all episodes for 180-day lookback	0+	Whole integers
EpisodeSpan180	Num(8)	15.	# of total length in days of all episodes for 180-day lookback	0+	Whole integers
Members180Q1	Num(8)	15.	# of index dates in Period/Q1 for 180-day lookback	0+	Whole integers
Members180Q2	Num(8)	15.	# of index dates in Period/Q2 for 180-day lookback	0+	Whole integers

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Members180Q3	Num(8)	15.	# of index dates in Period/Q3 for 180-day lookback	0+	Whole integers
Members180Q4	Num(8)	15.	# of index dates in Period/Q4 for 180-day lookback	0+	Whole integers
Members270	Num(8)	15.	Members with no same drug dispensing in 270-day lookback prior to index date	0+	Whole integers
Dispensings270	Num(8)	15.	# of Dispensings for 270-day lookback	0+	Whole integers
DaysSupply270	Num(8)	15.	# of total days supply for all episodes for 270-day lookback	0+	Whole integers
EpisodeSpan270	Num(8)	15.	# of total length in days of all episodes for 270-day lookback	0+	Whole integers
Members270Q1	Num(8)	15.	# of index dates in Period/Q1 for 270-day lookback	0+	Whole integers
Members270Q2	Num(8)	15.	# of index dates in Period/Q2 for 270-day lookback	0+	Whole integers
Members270Q3	Num(8)	15.	# of index dates in Period/Q3 for 270-day lookback	0+	Whole integers
Members270Q4	Num(8)	15.	# of index dates in Period/Q4 for 270-day lookback	0+	Whole integers
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating Incident Drug Class Summary Tables

1. Use the file(s) RxDrugClassYYYY, created by the process of **Extraction and Key Assignment of Dispensing Table** above, Step 3.
2. Merge these tables with the **Enr_FullCoverage_Spans_Bridged** table (built as part of creating the Enrollment Summary Table, Step 10) on the basis of PatKey and keep only dispensing rows where the RxDate is in between the dates of Enr_Start and Enr_End. Note that per patient, there can be multiple spans of Enr_Start to Enr_End, each of which should be checked.
3. For possible identical Drug Class dispensing rows to the same patient on the same RxDate:
    3.1. Set the value of RxSup to the maximum value of RxSup across such identical dispensing rows
    3.2. Count all rows within such grouping and save into variable NumClaims.
    3.3. Save one row per identical Drug Class dispensing rows to the same patient on the same RxDate.
4. Assess the extent of overlap and gaps between dispensing rows: While maintaining all rows, create and adjust selected variables of the dispensing rows as follows:
    4.1. Create a new variable, ExpireDt, set as RxDate + RxSup – 1 for all rows.
    4.2. Define groups of dispensing rows on the basis of patient, Drug Class, and RxDate.
    4.3. For the first row within a grouping:
        4.3.1. Create a new variable to count dispensing rows within a grouping, WithinGroupRowID, initialized to 1.
        4.3.2. Create a new variable to track the latest examined ExpireDt within a grouping, LExpireDt, initialized to missing.
        4.3.3. Create a new variable to track the latest examined RxSup within a grouping, LRxSup, initialized to missing.

4.3.4. Create a new variable which calculates in days the extent of overlap between dispensing row spans, Overlap_Prior, initialized to missing.
4.3.5. Create a new variable, FillStatus, that indicates the status of a row within a grouping, and set to "F", for first fill.
4.4. For the subsequent rows within a grouping:
4.4.1. Increment WithinGroupRowID by 1.
4.4.2. Set LExpireDt to ExpireDt, thus tracking the latest expiration date of a drug episode.
4.4.3. Set LRxSup to RxSup, thus tracking the latest days supply of a drug episode.
4.4.4. Set Overlap_Prior to the previous row's ExpireDt (i.e., LExpireDt) - RxDate +1. Then check OverLap_Prior and if greater than zero, perform the following:
4.4.4.1. Create PercentDay_Prior and set equal to Overlap_Prior / LRxSup, thus establishing the percent of overlap between the current and prior row's RxSup.
4.4.4.2. Set FillStatus = "+".
4.4.4.3. Set RxDate = previous row's ExpireDt (i.e., LExpireDt) + 1. Note that this may set the RxDate variable to a later date, when there is overlap of dispensing rows.
4.4.4.4. Set ExpireDt = RxDate + RxSup - 1, thus setting the end date of the current dispensing based on the newly adjusted RxDate.
4.4.5. If Overlap_Prior is zero or less, than we have a gap between rows. Set FillStatus = "G".
4.5. Write all rows to a table named Rx_DrugCategory_Dispensing_Assessed.

5. Identify dispensing rows in context of enrollment: Using the table **Rx_DrugCategory_Dispensing_Assessed** from all of Step 4, separate rows into two groups:
5.1. Where RxDate is later than Enr_End, write to table named as **Rx_DrugCategory_Exclude**. These rows will be examined later for ascertaining whether they fall in between later enrollment spans for a patient.
5.2. For all other rows, process as follows and then write to table named as **Rx_DrugCategory_Stockpiled**.
5.2.1. If ExpireDt is later than Enr_End or later than end of the Data Partner's data, then:
5.2.1.1. Set ExpireDt to the earliest of Enr_End or DP data, thus truncating the dispensing span to the end of the patient's enrollment span or DP data, whichever is earlier.
5.2.1.2. Reset RxSup equal to ExpireDt - RxDate + 1.

6. "Rescue" dispensing rows that fell beyond enrollment: Using the table **Rx_DrugCategory_Exclude** from Step 5.1 above, see if any of these dispensing rows are within a future enrollment span for the patients. This is done by merging **Rx_DrugCategory_Exclude** with the **Enr_FullCoverage_Spans_Bridged** table (built as part of creating the Enrollment Summary Table, Step 10), on PatKey, and keeping only dispensing rows where the RxDate is in between the dates of Enr_Start and Enr_End, writing any such rows to table **RX_DrugCategoryRescue** and then process as follows:
6.1. If RxDate is not equal to the original RxDate, then the AgeEventGroupKey may need to be changed. On the basis of PatKey, merge these rows only, with the **PatKeysCovKey** table and calculate AgeEventGroupKey, using RxDate and PatKeysCovKey.Birth_Date.
6.2. Set ExpireDt to the earliest of either the existing ExpireDt or to Enr_End.
6.3. Reset RxSup equal to ExpireDt - RxDate + 1.

6.4. Then interleave all of these rows with table **Rx_DrugCategory_Stockpiled**, on the basis of patient, Drug Class, and RxDate.
6.5. Write rows to new table Rx_DrugCategory_Stockpiled_All.

7. From multiple dispensing rows, identify single episodes across such dispensing rows, taking into account gap days: While maintaining all dispensing rows, create and adjust selected variables of the dispensing rows as follows:
   7.1. Define groups of dispensing rows on the basis of patient and Drug Class.
   7.2. Create variables that are carried from one row to another and are used to compare values in current rows to prior rows. These variables are: LRunOutDate, LEnrStart and an error-trapping counter variable, _Trap_.
   7.3. For the first row within a grouping, based on the earliest RxDate:
      7.3.1. Create a new variable LRunOutDate (last date of episode) equal to ExpireDt.
      7.3.2. Create a new variable LEnrStart (last enrollment start date) equal to Enr_Start.
      7.3.3. Create a new variable Episode and sequentially number each row within a grouping, starting at 1.
      7.3.4. Create a new variable Gap and set it to missing.
   7.4. For all other rows, set Gap equal to RxDate - LRunOutDate - 1. Then, If Gap is greater than the prescribed maximum number of gap days (i.e., 15), or LEnrStart is not equal to Enr_Start, then the current row is the beginning of a new drug episode.
      7.4.1. Increment Episode by 1.
      7.4.2. Set LRunOutDate to ExpireDt in the current row.
   7.5. When the condition in Step 7.4 is false, check if LRunOutDate is later than the current ExpireDt. If so, then we have overlapping dispensing rows.
      7.5.1. Increment _Trap_ by 1.
      7.5.2. Set LRunOutDate to the latter date of ExpireDt in the current row or to LRunOutDate, thus always setting this variable to the latest expiration date (ExpireDt) of a grouping.
   7.6. For each row, set LEnrStart equal to Enr_Start.
   7.7. Write the results of these dispensing rows to table **RX_DrugCategory_Episode_Claims**.

8. Summarize dispensing rows into episode rows: On the basis of patient, Drug Class, and Episode, summarize the dispensing rows as follows:
   8.1. Keep the following individual variables: Patient, Drug Class, and Episode.
   8.2. Take the lowest value of AgeEventGroupKey, to ensure that age key is set as of the beginning of the episode.
   8.3. Total number of NumClaims into variable EpiNumClaims, to get the number of dispensing rows per episode.
   8.4. Sum RxSup into variable EpiRxSup, to get total days supply per episode.
   8.5. Set the earliest RxDate into variable EpiStart, to ensure that the beginning of the episodes are set to the earliest dispensing row.
   8.6. Set the latest ExpireDt into variable EpiEnd, to ensure that the end date of the episodes are set to the latest expiration date of a dispensing row.
   8.7. Set the earliest EStart into same variable EStart, to ensure the earliest enrollment start date for the episode.
   8.8. Sort the episode rows by patient, Drug Class, and Episode (thereby sorting also by EpiStart), and save to table named **Rx_DrugCategory_Episodes**.

9. Filter episodes for those that meet the minimal 90-day lookback criterion: This will minimize the number of episodes that can possibly satisfy the 90, 180, and 270-day lookback periods. Read in the **Rx_DrugCategory_Episodes** table by a grouping of patient, Drug Class, and Episode.
   - 9.1. Calculate the end date of the prior episode's end date as variable PriorEpiEnd. For the first row per grouping, set this to missing.
   - 9.2. Set a variable, MinLookBackDt, as the minimal look back date from the beginning of an episode in a group equal to Epistart (see Step 8.5) – 90 or DP data start + 90, whichever is later.
   - 9.3. If PriorEpiEnd is earlier than MinLookBackDt and EStart (see Step 8.7) is earlier than or equal to MinLookBackDt, then the current episode is an incident episode, as no observed episode is found earlier than 90 days and we have observed the start of enrollment earlier than or equal to the minimal lookback date. When this occurs, calculate the following variables:
     - 9.3.1. Year as the 4-character year value of EpiStart.
     - 9.3.2. Qtr as the 1-character value of the calendar quarter of EpiStart.
     - 9.3.3. EpiDuration as EpiEnd - EpiStart + 1.
   - 9.4. Sort the kept episode rows by PatKey, Drug Class, Year, and Episode and write episode rows to table named **Candidates_RX_DrugCategory**.
10. Select the first incident episode per period, outputting a row for each of the 3 lookback periods (i.e., 90, 180, or 270 days) for which it qualifies: Read in table **Candidates_RX_DrugCategory** in its sorted order and subset only the first episode row per calendar year. Assign a variable, LookBack, to each of lookback periods 90, 180, and 270 and then perform the following for each value of LookBack:
    - 10.1. Calculate a LookBackDt variable as EpiStart – LookBack.
    - 10.2. Keep episodes that meet the following conditions: Both DP start date and EStart are earlier than or equal to LookBackDt (thus both minimal DP data spans and enrollment criteria are maintained) and PriorEpiEnd (see Step 9.1) is earlier than LookBackDt (thus the current episode is incident.
    - 10.3. Write these saved rows to a table named **Rx_DrugCategory_Epi_LookBack**.
11. From **Rx_DrugCategory_Epi_LookBack**, create counts of index dates within quarterly Periods on the classification combination of AgeEventGroupKey, Sex, Period, and Drug Class for each lookback value (i.e., 90, 180, 270).
    - 11.1. Count the number of index dates in the first calendar quarter and set variable Members###Q1 to this value.
    - 11.2. Count the number of index dates in the second calendar quarter and set variable Members###Q2 to this value.
    - 11.3. Count the number of index dates in the third calendar quarter and set variable Members###Q3 to this value.
    - 11.4. Count the number of index dates in the fourth calendar quarter and set variable Members###Q4 to this value.
12. From **Rx_DrugCategory_Epi_LookBack** again, create three aggregates as follows, on the classification combination of AgeEventGroupKey, Sex, Period, and Drug Class, for each lookback value (i.e., 90, 180, 270):
    - 12.1. Count the number of unique PatIDs and save this value to Members###
    - 12.2. Sum EpiNumClaims and save this value to Dispensings###
    - 12.3. Sum EpiRxSup and save this value to DaysSupply###.
    - 12.4. Sum EpiDuration and save this value to EpisodeSpan###.
    - 12.5. Note that the sum of counts in Members###Q1 through Members###Q4 must be equal to Members### within all strata.

13. Merge all aggregates on the basis of AgeEventGroupKey, Sex, Period, and Drug Class to file **IncRxSumm**. Set any missing values to zero, for variables Members###, Dispensings###, DaysSupply###, EpisodeSpan###, Members###Q1, Members###Q2, Members###Q3, and Members###Q4.
14. Using the AgeEventGroupKey variable, create the variables Age_Group and Age_Group_ID.
15. Then link this file with NDC_Lookup_Table, on IncRxClass.DrugCategoryKey = Lookup.DrugCategoryKey, to get the Lookup.DrugClass variable.
16. Name the table **Incident_Drug_Class** and structure table as per data dictionary above. Sort by Age_Group_ID, Sex, Period, and DrugClass. Save to the DPLocal storage area.

P. INCIDENT INGREDIENT/GENERIC NAME SUMMARY TABLE

The incident Generic Name table provides a count of unique members with an incident dispensing for each generic drug name of interest (e.g. prednisone, tamsulosin HCL) stratified by age group, sex, and year.

Incidence is defined as a member with a dispensing with the Generic Name of interest (i.e., the index date), in the year of interest with no evidence of a dispensing for that generic name in the 90, 180 and 270 days (i.e., the lookback periods) before the index date.

Both medical and drug coverage are required during the 3 possible lookback periods, allowing for eligibility gaps of <=45 days. Further, both forms of enrollment coverage are required throughout the period of the dispensing episode. Following rules in Cohort Identification and Data Analysis (CIDA), only dispensings with the start of their dispensing within enrollment coverage are utilized. That is, the start of the dispensing must have full enrollment coverage, both before and after stockpiling. After stockpiling, if the start of a dispensing has been changed, it is the new start date that must be fully covered by enrollment. Stockpiling is the process of resetting to a later start date, those dispensing rows whose original start date overlapped prior dispensings, using the assumption that a patient will consume all doses in all filled prescriptions.

In addition to reporting the number of members with an incident dispensing, for each such incident user a treatment episode starting on the index date is created, and the total number of dispensings, days supplied and length of treatment episodes (in days) is measured for each treatment episode. Treatment gaps of <= 15 days are allowed when creating episodes and are considered part of the same treatment episode. Treatment episodes are censored by a gap in treatment, a gap in enrollment, or the end of Data Partner data, whichever occurs first. Although a member can have multiple index events in a given calendar year the first one only is counted and used for reporting.

The counts are stratified by Generic Name, age group, sex, and year. Generic Names are standardized using a look-up table provided by the Sentinel Operations Center. For each stratum the results contain 3 separate sections for each of the 90, 180 and 270 lookback scenarios. Each section contains the total number of members, total dispensings, total days supplied and total length of all episodes, as well as a quarterly breakdown of index dates (must sum up to total number of members).

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group	Char(5)	$5.	Age Group	0-1, 2-4, 5-9, 10-14, 15-18, 19-21, 22-44, 45-64, 65-74, 75+	
Sex	Char(1)	$1.	Sex	M or F	
Period	Char(4)	$4.	Year	2000+	
GenericName	Char(30)	$30.	Generic Drug Name	Text description	
Members90	Num(8)	15.	Members with no same drug dispensing in 90-day lookback prior to index date	1+	Whole integers
Dispensings90	Num(8)	15.	# of Dispensings for 90-day lookback	1+	Whole integers
DaysSupply90	Num(8)	15.	# of total days supply for all episodes for 90-day lookback	1+	Whole integers
EpisodeSpan90	Num(8)	15.	# of total length in days of all episodes for 90-day lookback	1+	Whole integers
Members90Q1	Num(8)	15.	# of index dates in Period/Q1 for 90-day lookback	0+	Whole integers
Members90Q2	Num(8)	15.	# of index dates in Period/Q2 for 90-day lookback	0+	Whole integers
Members90Q3	Num(8)	15.	# of index dates in Period/Q3 for 90-day lookback	0+	Whole integers
Members90Q4	Num(8)	15.	# of index dates in Period/Q4 for 90-day lookback	0+	Whole integers
Members180	Num(8)	15.	Members with no same drug dispensing in 180-day lookback prior to index date	0+	Whole integers
Dispensings180	Num(8)	15.	# of Dispensings for 180-day lookback	0+	Whole integers
DaysSupply180	Num(8)	15.	# of total days supply for all episodes for 180-day lookback	0+	Whole integers
EpisodeSpan180	Num(8)	15.	# of total length in days of all episodes for 180-day lookback	0+	Whole integers
Members180Q1	Num(8)	15.	# of index dates in Period/Q1 for 180-day lookback	0+	Whole integers
Members180Q2	Num(8)	15.	# of index dates in Period/Q2 for 180-day lookback	0+	Whole integers
Members180Q3	Num(8)	15.	# of index dates in Period/Q3 for 180-day lookback	0+	Whole integers
Members180Q4	Num(8)	15.	# of index dates in Period/Q4 for 180-day lookback	0+	Whole integers
Members270	Num(8)	15.	Members with no same drug dispensing in 270-day lookback prior to index date	0+	Whole integers
Dispensings270	Num(8)	15.	# of Dispensings for 270-day lookback	0+	Whole integers
DaysSupply270	Num(8)	15.	# of total days supply for all episodes for 270-day lookback	0+	Whole integers
EpisodeSpan270	Num(8)	15.	# of total length in days of all episodes for 270-day lookback	0+	Whole integers
Members270Q1	Num(8)	15.	# of index dates in Period/Q1 for 270-day lookback	0+	Whole integers
Members270Q2	Num(8)	15.	# of index dates in Period/Q2 for 270-day lookback	0+	Whole integers
Members270Q3	Num(8)	15.	# of index dates in Period/Q3 for 270-day lookback	0+	Whole integers
Members270Q4	Num(8)	15.	# of index dates in Period/Q4 for 270-day lookback	0+	Whole integers

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Age_Group_ID	Num(3)	2.	ID	1+	

Methods for Creating Incident Generic Name Summary Tables

1. Use the file(s) RxGenericNameYYYY, created by the process of **Extraction and Key Assignment of Dispensing Table** above, Step 3.
2. Merge these tables with the **Enr_FullCoverage_Spans_Bridged** table (built as part of creating the Enrollment Summary Table, Step 10) on the basis of PatKey and keep only dispensing rows where the RxDate is in between the dates of Enr_Start and Enr_End. Note that per patient, there can be multiple spans of Enr_Start to Enr_End, each of which should be checked.
3. For possible identical Generic Name dispensing rows to the same patient on the same RxDate:
   - 3.1. Set the value of RxSup to the maximum value of RxSup across such identical dispensing rows
   - 3.2. Count all rows within such grouping and save into variable NumClaims.
   - 3.3. Save one row per identical Drug Class dispensing rows to the same patient on the same RxDate.
4. Assess the extent of overlap and gaps between dispensing rows: While maintaining all rows, create and adjust selected variables of the dispensing rows as follows:
   - 4.1. Create a new variable, ExpireDt, set as RxDate + RxSup – 1 for all rows.
   - 4.2. Define groups of dispensing rows on the basis of patient, Generic Name, and RxDate.
   - 4.3. For the first row within a grouping:
     - 4.3.1. Create a new variable to count dispensing rows within a grouping, WithinGroupRowID, initialized to 1.
     - 4.3.2. Create a new variable to track the latest examined ExpireDt within a grouping, LExpireDt, initialized to missing.
     - 4.3.3. Create a new variable to track the latest examined RxSup within a grouping, LRxSup, initialized to missing.
     - 4.3.4. Create a new variable which calculates in days the extent of overlap between dispensing row spans, Overlap_Prior, initialized to missing.
     - 4.3.5.Create a new variable, FillStatus, that indicates the status of a row within a grouping, and set to “F”, for first fill.
   - 4.4. For the subsequent rows within a grouping:
     - 4.4.1. Increment WithinGroupRowID by 1.
     - 4.4.2. Set LExpireDt to ExpireDt, thus tracking the latest expiration date of a drug episode.
     - 4.4.3. Set LRxSup to RxSup, thus tracking the latest days supply of a drug episode.
     - 4.4.4. Set Overlap_Prior to the previous row’s ExpireDt (i.e., LExpireDt) - RxDate +1. Then check OverLap_Prior and if greater than zero, perform the following:
       - 4.4.4.1. Create PercentDay_Prior and set equal to Overlap_Prior / LRxSup, thus establishing the percent of overlap between the current and prior row’s RxSup.
       - 4.4.4.2. Set FillStatus = “+”.
       - 4.4.4.3. Set RxDate = previous row’s ExpireDt (i.e., LExpireDt) + 1. Note that this may set the RxDate variable to a later date, when there is overlap of dispensing rows.

4.4.4.4. Set ExpireDt = RxDate + RxSup - 1, thus setting the end date of the current dispensing based on the newly adjusted RxDate.
4.4.5. If Overlap_Prior is zero or less, than we have a gap between rows. Set FillStatus = "G".
4.5. Write all rows to a table named Rx_DrugCategory_Dispensing_Assessed.

5. Identify dispensing rows in context of enrollment: Using the table **Rx_DrugCategory_Dispensing_Assessed** from all of Step 4, separate rows into two groups:
   5.1. Where RxDate is later than Enr_End, write to table named as **Rx_DrugCategory_Exclude**. These rows will be examined later for ascertaining whether they fall in between later enrollment spans for a patient.
   5.2. For all other rows, process as follows and then write to table named **Rx_DrugCategory_Stockpiled**.
      5.2.1. If ExpireDt is later than Enr_End or later than end of the Data Partner's data, then:
         5.2.1.1. Set ExpireDt to the earliest of Enr_End or DP data, thus truncating the dispensing span to the end of the patient's enrollment span or DP data, whichever is earlier.
         5.2.1.2. Reset RxSup equal to ExpireDt - RxDate + 1.
6. "Rescue" dispensing rows that fell beyond enrollment: Using the table **Rx_DrugCategory_Exclude** from Step 5.1 above, see if any of these dispensing rows are within a future enrollment span for the patients. This is done by merging **Rx_DrugCategory_Exclude** with the **Enr_FullCoverage_Spans_Bridged** table (built as part of creating the Enrollment Summary Table, Step 10), on PatKey, and keeping only dispensing rows where the RxDate is in between the dates of Enr_Start and Enr_End, writing any such rows to table **RX_DrugCategoryRescue** and then process as follows:
   6.1. If RxDate is not equal to the original RxDate, then the AgeEventGroupKey may need to be changed. On the basis of PatKey, merge these rows only, with the **PatKeysCovKey** table and calculate AgeEventGroupKey, using RxDate and PatKeysCovKey.Birth_Date.
   6.2. Set ExpireDt to the earliest of either the existing ExpireDt or to Enr_End.
   6.3. Reset RxSup equal to ExpireDt - RxDate + 1.
   6.4. Then interleave all of these rows with table **Rx_DrugCategory_Stockpiled**, on the basis of patient, Generic Name, and RxDate.
   6.5. Write rows to new table Rx_DrugCategory_Stockpiled_All.
7. From multiple dispensing rows, identify single episodes across such dispensing rows, taking into account gap days: While maintaining all dispensing rows, create and adjust selected variables of the dispensing rows as follows:
   7.1. Define groups of dispensing rows on the basis of patient and Generic Name.
   7.2. Create variables that are carried from one row to another and are used to compare values in current rows to prior rows. These variables are: LRunOutDate, LEnrStart and an error-trapping counter variable, _Trap_.
   7.3. For the first row within a grouping, based on the earliest RxDate:
      7.3.1. Create a new variable LRunOutDate (last date of episode) equal to ExpireDt.
      7.3.2. Create a new variable LEnrStart (last enrollment start date) equal to Enr_Start.
      7.3.3. Create a new variable Episode and sequentially number each row within a grouping, starting at 1.
      7.3.4. Create a new variable Gap and set it to missing.
   7.4. For all other rows, set Gap equal to RxDate - LRunOutDate - 1. Then, If Gap is greater than the prescribed maximum number of gap days (i.e., 15), or LEnrStart is not equal to Enr_Start, then the current row is the beginning of a new drug episode.

7.4.1. Increment Episode by 1.
7.4.2. Set LRunOutDate to ExpireDt in the current row.
7.5. When the condition in Step 7.4 is false, check if LRunOutDate is later than the current ExpireDt. If so, then we have overlapping dispensing rows.
7.5.1. Increment _Trap_ by 1.
7.5.2. Set LRunOutDate to the latter date of ExpireDt in the current row or to LRunOutDate, thus always setting this variable to the latest expiration date (ExpireDt) of a grouping.
7.6. For each row, set LEnrStart equal to Enr_Start.
7.7. Write the results of these dispensing rows to table **RX_DrugCategory_Episode_Claims**.

8. Summarize dispensing rows into episode rows: On the basis of patient, Generic Name, and Episode, summarize the dispensing rows as follows:
8.1. Keep the following individual variables: Patient, Generic Name, and Episode.
8.2. Take the lowest value of AgeEventGroupKey, to ensure that age key is set as of the beginning of the episode.
8.3. Total number of NumClaims into variable EpiNumClaims, to get the number of dispensing rows per episode.
8.4. Sum RxSup into variable EpiRxSup, to get total days supply per episode.
8.5. Set the earliest RxDate into variable EpiStart, to ensure that the beginning of the episodes are set to the earliest dispensing row.
8.6. Set the latest ExpireDt into variable EpiEnd, to ensure that the end date of the episodes are set to the latest expiration date of a dispensing row.
8.7. Set the earliest EStart into same variable EStart, to ensure the earliest enrollment start date for the episode.
8.8. Sort the episode rows by patient, Generic Name, and Episode (thereby sorting also by EpiStart), and save to table named **Rx_DrugCategory_Episodes**.

9. Filter episodes for those that meet the minimal 90-day lookback criterion: This will minimize the number of episodes that can possibly satisfy the 90, 180, and 270-day lookback periods. Read in the **Rx_DrugCategory_Episodes** table by a grouping of patient, Generic Name, and Episode.
9.1. Calculate the end date of the prior episode's end date as variable PriorEpiEnd. For the first row per grouping, set this to missing.
9.2. Set a variable, MinLookBackDt, as the minimal look back date from the beginning of an episode in a group equal to Epistart (see Step 8.5) – 90 or DP data start + 90, whichever is later.
9.3. If PriorEpiEnd is earlier than MinLookBackDt and EStart (see Step 8.7) is earlier than or equal to MinLookBackDt, then the current episode is an incident episode, as no observed episode is found earlier than 90 days and we have observed the start of enrollment earlier than or equal to the minimal lookback date. When this occurs, calculate the following variables:
9.3.1. Year as the 4-character year value of EpiStart.
9.3.2. Qtr as the 1-character value of the calendar quarter of EpiStart.
9.3.3. EpiDuration as EpiEnd - EpiStart + 1.
9.4. Sort the kept episode rows by PatKey, Generic Name, Year, and Episode and write episode rows to table named **Candidates_RX_DrugCategory**.

10. Select the first incident episode per period, outputting a row for each of the 3 lookback periods (i.e., 90, 180, or 270 days) for which it qualifies: Read in table **Candidates_RX_DrugCategory** in its sorted order and subset only the first episode row per calendar year. Assign a variable, LookBack, to each of lookback periods 90, 180, and 270 and then perform the following for each value of LookBack:
    10.1. Calculate a LookBackDt variable as EpiStart – LookBack.
    10.2. Keep episodes that meet the following conditions: Both DP start date and EStart are earlier than or equal to LookBackDt (thus both minimal DP data spans and enrollment criteria are maintained) and PriorEpiEnd (see Step 9.1) is earlier than LookBackDt (thus the current episode is incident.
    10.3. Write these saved rows to a table named **Rx_DrugCategory_Epi_LookBack**.
11. From **Rx_DrugCategory_Epi_LookBack**, create counts of index dates within quarterly Periods on the classification combination of AgeEventGroupKey, Sex, Period, and Generic Name for each lookback value (i.e., 90, 180, 270).
    11.1. Count the number of index dates in the first calendar quarter and set variable Members###Q1 to this value.
    11.2. Count the number of index dates in the second calendar quarter and set variable Members###Q2 to this value.
    11.3. Count the number of index dates in the third calendar quarter and set variable Members###Q3 to this value.
    11.4. Count the number of index dates in the fourth calendar quarter and set variable Members###Q4 to this value.
12. From **Rx_DrugCategory_Epi_LookBack** again, create three aggregates as follows, on the classification combination of AgeEventGroupKey, Sex, Period, and Generic Name, for each lookback value (i.e., 90, 180, 270):
    12.1. Count the number of unique PatIDs and save this value to Members###
    12.2. Sum EpiNumClaims and save this value to Dispensings###
    12.3. Sum EpiRxSup and save this value to DaysSupply###.
    12.4. Sum EpiDuration and save this value to EpisodeSpan###.
    12.5. Note that the sum of counts in Members###Q1 through Members###Q4 must be equal to Members### within all strata.
13. Merge all aggregates on the basis of AgeEventGroupKey, Sex, Period, and Generic Name to file **IncRxSumm**. Set any missing values to zero, for variables Members###, Dispensings###, DaysSupply###, EpisodeSpan###, Members###Q1, Members###Q2, Members###Q3, and Members###Q4.
14. Using the AgeEventGroupKey variable, create the variables Age_Group and Age_Group_ID.
15. Then link this file with NDC Lookup Table, on IncRxClass.GenericNameKey = Lookup.GenericNameKey, to get the Lookup.GenericName variable.
16. Name the table **Incident_Generic_Name** and structure table as per data dictionary above. Sort by Age_Group_ID, Sex, Period, and DrugClass. Save to the DPLocal storage area.

Q. EXPORT TEXT AND ACCESS FILES

The final process is to output flat text data files and optionally Access databases, containing all of the generated summary tables. The selection of which files are exported is based on both the capabilities of the Data Partner as well as their choices.

Methods for Generating Export Files – Environmental

1. Create multiple user-modifiable parameters as follows that will be completed by the Data Partner and entered at the top of the program package.

Parameter	Purpose	Parameter Values
TextFiles	REQUIRED Text files are always generated; this parameter indicates the type of text files	Must be one of the following values: C = Comma delimited P = Pipe delimited T = Tab delimited
ExportPath	OPTIONAL Indicates the path into which the text files and MS Access database (optionally) are to be written, if not DPLocal	Operating system full path
AccessFiles	REQUIRED Indicates if an Access database is to be generated and the type	Must be one of the following values: N = No, do not generate Access files M = Generate a MDB format, compatible with Office version up *to* 2007 A = Generate an ACCDB format, compatible with Office versions from 2007 and later
Access_DBMS	OPTIONAL, when AccessFiles = M or A Indicates the DBMS parameter for PROC EXPORT, dependent on platform	ACCESS: For translating between 32-bit SAS and 32-bit MS Office Access ACCESSCS: For translating between 64-bit SAS and 32-bit MS Office Access

2. Validate the parameters above as follows:
   - 2.1. TextFiles: This must be filled and only for the values shown above. If values fail these criteria, then put a message to the log and stop processing the entire package.
   - 2.2. ExportPath: If filled, then validate that the path exists and that the program has write privileges to the directory. Otherwise, do not check this parameter. If the path does not exist or if write privileges are not available, then put a message to the log and stop processing the entire package.
   - 2.3. AccessFiles: This must be filled and only with the values shown above. If values fail these criteria, then put a message to the log and stop processing the entire package.
     - 2.3.1. If *AccessFiles* is not equal to "N", then check the Common Components *_sas_access* parameter. If this parameter equals "N", then Access databases cannot be generated; put the following message to the log and stop processing the entire package:

Access Database cannot be generated as SAS/Access Interface to PC files is not indicated to be available
Check installation of SAS components and the setting for the _sas_Access parm in 'Common Components.sas'
Summary Table V2 Programming Specifications
Version 1.0

Summary Table processing will be aborted

Methods for Generating Export Files – Text files

1. Create a processing loop for all 13 Summary Tables (Age_Groups, Enrollment, ICD9_Diagnosis, ICD9_Diagnosis_4_Digit, ICD9_Diagnosis_5_Digit, HCPCS, ICD9_Procedure, ICD9_Procedure_4_Digit, Drug_Class, Generic_Name, Incident_ICD9_Diagnosis, Incident_Drug_Class, Incident_Generic_Name)
2. For each iteration through the loop per table, do the following:
   2.1. Determine if the Summary Table exists.
      2.1.1. If it does not exist, then put the following message to the log:
      WARNING: Table [name of table] does not exist and will not be exported to a text file.
      WARNING: This issue should be investigated by the DP and SOC.

   2.2. If it does exist, then write the table as *TableName.txt,* to the validated path indicated by the *TextPath* parameter.
   2.3. Output one data text line, per SAS observation.
3. Follow these formatting requirements per data type for the variables:
   3.1. Character: Output the values of the variables, embedded in double-quotes.
   3.2. Numeric: Output the values of the variables as only a string of digits, with no punctuation, and not embedded in double-quotes.
   3.3. In between the variables written, insert a delimiter as specified by the *TextFiles* parameter.
   3.4. Note that there are no date or time formatted variables to be processed.

Methods for Generating Export Files – Access Databases

1. If the *AccessFiles* parameter is not equal to "N", then create a processing loop for all Summary Tables, per step 3 above:
2. For each iteration through the loop, per table, do the following:
   2.1. Determine if the Summary Table exists.
      2.1.1. If it does not exist, then put the following message to the log:
      WARNING: Table [name of table] does not exist and will not be exported to Access database.
      WARNING: This issue should be investigated by the DP and SOC.

   2.2. If it does exist, then write the table as TableName, to an Access database named as Mini_Sentinel_Summary_Tables.*Extension*, to the path indicated by the *ExportPath* parameter, where *Extension* is based on the *AccessFiles* parameter and the *Access_DBMS* user parameters is applied as requested.

V. APPENDIX A: LOOKUP TABLE DX_ICD9_3DIG_LOOKUP

Includes descriptions for 3-character ICD9 codes.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Category	Char(120)	$120.	category	Free text	Broad classification of ICD9 codes
Code	Char(6)	$6.	Code	Free text	ICD9 Diagnosis code (3-characters)
Dcode	Char(6)	$6.	dcode	Free text	ICD9 Diagnosis code (3-characters)
Srt_descrip	Char(35)	$35.	srt_descrip	Free text	Short description for code
Lng_descrip	Char(48)	$48.	lng_descrip	Free text	Long description for code

Sample Rows

Category	Code	DCode	srt_descrip	lng_descrip
(580-629)Diseases Of The Genitourinary System	626	626	D/O MENS&OTH ABN BLEED FE GNT TRACT	D/O MENSTRUATION&OTH ABN BLEED FE GENIT TRACT
(580-629)Diseases Of The Genitourinary System	627	627	MENOPAUSAL&POSTMENOPAUSAL DISORDERS	MENOPAUSAL AND POSTMENOPAUSAL DISORDERS
(580-629)Diseases Of The Genitourinary System	628	628	FEMALE INFERTILITY	FEMALE INFERTILITY
(580-629)Diseases Of The Genitourinary System	629	629	OTH DISORDERS FEMALE GENITAL ORGANS	OTHER DISORDERS OF FEMALE GENITAL ORGANS
(630-676)Complications of Pregnancy, Childbirth, and the Puerperium	630	630	HYDATIDIFORM MOLE	HYDATIDIFORM MOLE
(630-676)Complications of Pregnancy, Childbirth, and the Puerperium	631	631	OTHER ABNORMAL PRODUCT CONCEPTION	OTHER ABNORMAL PRODUCT OF CONCEPTION
(630-676)Complications of Pregnancy, Childbirth, and the Puerperium	632	632	MISSED ABORTION	MISSED ABORTION

VI. APPENDIX B: LOOKUP TABLE DX_ICD9_4DIG_LOOKUP

Includes descriptions for 4-character ICD9 codes.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Category	Char(120)	$120.	Category	Free text	Broad classification of ICD9 codes
Code	Char(6)	$6.	Code	Free text	ICD9 Diagnosis code (4-characters)
Dcode	Char(6)	$6.	Dcode	Free text	ICD9 Diagnosis code (4-characters) with decimal point
Srt_descrip	Char(35)	$35.	srt_descrip	Free text	Short description for code
Lng_descrip	Char(48)	$48.	lng_descrip	Free text	Long description for code

Sample Rows

Category	Code	DCode	Srt_descrip	Lng_descrip
(140-239)Neoplasms	2397	239.7	NEO UNS NATUR ENDOCRNE&OTH NERV SYS	NEOPLSM UNS NATR ENDOCRN GLND&OTH PART NERV SYS
(140-239)Neoplasms	2398	239.8	NEOPLASM UNSPEC NATR OTH SPEC SITES	NEOPLASM UNSPEC NATURE OTHER SPEC SITES
(140-239)Neoplasms	2399	239.9	NEOPLASM UNSPEC NATURE SITE UNSPEC	NEOPLASM OF UNSPECIFIED NATURE SITE UNSPECIFIED
(240-279)Endocrine, Nutritional And Metabolic Diseases, And Immunity Disorders	2400	240.0	GOITER SPECIFIED AS SIMPLE	GOITER, SPECIFIED AS SIMPLE
(240-279)Endocrine, Nutritional And Metabolic Diseases, And Immunity Disorders	2409	240.9	GOITER UNSPECIFIED	GOITER, UNSPECIFIED
(240-279)Endocrine, Nutritional And Metabolic Diseases, And Immunity Disorders	2410	241.0	NONTOXIC UNINODULAR GOITER	NONTOXIC UNINODULAR GOITER

VII. APPENDIX C: LOOKUP TABLE DX_ICD9_5DIG_LOOKUP

Includes descriptions for 4-character ICD9 codes.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Category	Char(120)	$120.	Category	Free text	Broad classification of ICD9 codes
Code	Char(6)	$6.	Code	Free text	ICD9 Diagnosis code (5-characters)
Dcode	Char(6)	$6.	Dcode	Free text	ICD9 Diagnosis code (5-characters) with decimal point
Srt_descrip	Char(35)	$35.	srt_descrip	Free text	Short description for code
Lng_descrip	Char(48)	$48.	lng_descrip	Free text	Long description for code

Sample Rows

Category	Code	DCode	Srt_descrip	Lng_descrip
(001-139)Infectious And Parasitic Diseases	13109	131.09	OTHER UROGENITAL TRICHOMONIASIS	OTHER UROGENITAL TRICHOMONIASIS
(001-139)Infectious And Parasitic Diseases	13621	136.21	SPEC INFECTION DUE TO ACANTHAMOEBA	SPECIFIC INFECTION DUE TO ACANTHAMOEBA
(001-139)Infectious And Parasitic Diseases	13629	136.29	OTH SPECIFIC INF FREE-LIVING AMEBAE	OTHER SPECIFIC INFECTIONS FREE-LIVING AMEBAE
(140-239)Neoplasms	17300	173.00	UNS MALIGNANT NEOPLASM SKIN LIP	UNSPECIFIED MALIGNANT NEOPLASM OF SKIN OF LIP
(140-239)Neoplasms	17301	173.01	BASAL CELL CARCINOMA OF SKIN OF LIP	BASAL CELL CARCINOMA OF SKIN OF LIP
(140-239)Neoplasms	17302	173.02	SQUAMOUS CELL CARCINOMA SKIN OF LIP	SQUAMOUS CELL CARCINOMA OF SKIN OF LIP

VIII. APPENDIX D: LOOKUP TABLE PX_LOOKUP

Includes descriptions for 4-character ICD9 codes.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Source	Char(5)	$5.	Source	Free text	Categorization of PX codes; i.e., PX_CodeType
Code	Char(6)	$6.	Code	Free text	Procedure code
Srt_descrip	Char(35)	$35.	Srt_descrip	Free text	Short description for code
Lng_descrip	Char(48)	$48.	Lng_descrip	Free text	Long description for code
Category	Char(9)	$9.	Category	Free text	Category of code

Sample Rows

Source	Code	Srt_descrip	Lng_descrip	Category
cpt	22841	INSERT SPINE FIXATION DEVICE	INTERNAL SPINAL FIXATION WIRING SPINOUS PROCESS	UNDEFINED
cpt	48999	PANCREAS SURGERY PROCEDURE	UNLISTED PROCEDURE PANCREAS	UNDEFINED
cpt	90287	BOTULINUM ANTITOXIN	BOTULINUM ANTITOXIN EQUINE ANY ROUTE	UNDEFINED
cpt	99215	OFFICE/OUTPATIENT VISIT EST	OFFICE OUTPATIENT VISIT 40 MINUTES	UNDEFINED
hcpcs	D6074	ABUT RETN CAST METL FPD NOBL METL	ABUTMENT RETAINR CAST METAL FPD NOBLE METAL	UNDEFINED
hcpcs	E0370	AIR PRESSURE ELEVATOR FOR HEEL	AIR PRESSURE ELEVATOR FOR HEEL	UNDEFINED
hcpcs	G0240	BONE MARROW ASP/BIOPSY;SAME INCISN	BONE MARROW ASPIRATION/BIOPSY;SAME INCISION	UNDEFINED
hcpcs	J3390	INJ METHOXAMINE TO 20 MG	Injection, methoxamine hcl, up to 20 mg	UNDEFINED
hcpcs	L3252	FOOT SHOE MOLD PT PLASTAZOTE CSTM	FOOT SHOE MOLDED PT MDL PLASTAZOTE CSTM FABR EA	UNDEFINED
hcpcs	S0206	ADD CODE PRIM PROC DENOTE FACL&EQP	ADDITION CODE PRIM PROC DENOTE USE FACILITY&EQP	UNDEFINED

IX. APPENDIX E: LOOKUP TABLE PX_ICD9_3DIG_LOOKUP

Includes descriptions for 3-character ICD9 Procedure codes.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Category	Char(65)	$65.	Category	Free text	Category of code
DCode	Char(6)	$6.	DCode	Free text	ICD9 Procedure code, with decimal point
Code	Char(6)	$6.	Code	Free text	ICD9 Procedure code, no decimal point
Srt_descrip	Char(35)	$35.	Srt_descrip	Free text	Short description for code
Lng_descrip	Char(48)	$48.	Lng_descrip	Free text	Long description for code

Sample Rows

Category	DCode	Code	Srt_descrip	Lng_descrip
PROCEDURES AND INTERVENTIONS , NOT ELSEWHERE CLASSIFIED (00)	00.0	000	THERAPEUTIC ULTRASOUND	THERAPEUTIC ULTRASOUND
PROCEDURES AND INTERVENTIONS , NOT ELSEWHERE CLASSIFIED (00)	00.1	001	PHARMACEUTICALS	PHARMACEUTICALS
OPERATIONS ON THE NOSE, MOUTH, AND PHARYNX (21-29)	22.2	222	INTRANASAL ANTROTOMY	INTRANASAL ANTROTOMY
OPERATIONS ON THE NOSE, MOUTH, AND PHARYNX (21-29)	22.3	223	EXTERNAL MAXILLARY ANTROTOMY	EXTERNAL MAXILLARY ANTROTOMY
MISCELLANEOUS DIAGNOSTIC AND THERAPEUTIC PROCEDURES (87-99)	95.4	954	NONOPERATIVE PROC RELATED HEARING	NONOPERATIVE PROCEDURES RELATED TO HEARING
MISCELLANEOUS DIAGNOSTIC AND THERAPEUTIC PROCEDURES (87-99)	96.0	960	NONOP INTUBAT GI&RESPIRATORY TRACTS	NONOPERATIVE INTUBATION GI&RESPIRATORY TRACTS

X. APPENDIX F: LOOKUP TABLE PX_ICD9_4DIG_LOOKUP

Includes descriptions for 4-character ICD9 procedure codes.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
Category	Char(65)	$65.	Category	Free text	Category of code
DCode	Char(6)	$6.	DCode	Free text	ICD9 Procedure code, with decimal point
Code	Char(6)	$6.	Code	Free text	ICD9 Procedure code, no decimal point
Srt_descrip	Char(35)	$35.	Srt_descrip	Free text	Short description for code
Lng_descrip	Char(48)	$48.	Lng_descrip	Free text	Long description for code

Sample Rows

Category	DCode	Code	Srt_descrip	Lng_descrip
PROCEDURES AND INTERVENTIONS , NOT ELSEWHERE CLASSIFIED (00)	00.32	0032	COMPUTER ASSISTED SURGERY W/MR/MRA	COMPUTER ASSISTED SURGERY WITH MR/MRA
PROCEDURES AND INTERVENTIONS , NOT ELSEWHERE CLASSIFIED (00)	00.33	0033	COMPUTER ASSTD SURGERY W/FLUORO	COMPUTER ASSISTED SURGERY WITH FLUOROSCOPY
OPERATIONS ON THE DIGESTIVE SYSTEM (42-54)	52.84	5284	AUTOTPLNT CELLS ISLETS LANGERHANS	AUTOTRANSPLANTATION CELLS ISLETS LANGERHANS
OPERATIONS ON THE DIGESTIVE SYSTEM (42-54)	52.85	5285	ALLOTPLNT CELLS ISLETS LANGERHANS	ALLOTRANSPLANTATION CELLS ISLETS LANGERHANS
OPERATIONS ON THE MUSCULOSKELETAL SYSTEM (76-84)	84.80	8480	INSRT/REPLCMT INTRSPINUS PRC DEVC	INSERTION/REPLCMT INTERSPINOUS PROCESS DEVICE(S)
OPERATIONS ON THE MUSCULOSKELETAL SYSTEM (76-84)	84.81	8481	REV INTERSPINOUS PROCESS DEVICE(S)	REVISION OF INTERSPINOUS PROCESS DEVICE(S)

XI. APPENDIX G: LOOKUP TABLE NDC_LOOKUP_TABLE

Includes generic names and drug class descriptions drug dispensings by NDC.

Variable	Type (Length)	Format	Label	Valid Values	Source / Comments
NDC	Char(11)	$11.	NDC	Digits only	National Drug Code
GenericName	Char(30)	$30.	Generic Name	Free text	Generic name for drug
GenericNameKey	Num(3)	6.	Generic Name Key	1-8192*	Unique numeric value that has a 1:1 relationship to GenericName values
DrugClass	Char(70)	$70.	Drug Class	Free text	Drug class/category for drug
DrugClassKey	Num(3)	6.	Drug Class Key	1-8192*	Unique numeric value that has a 1:1 relationship to DrugClass values

*This is the maximum value that can be accurately represented with Num(3) under UNIX and Windows systems. If more values are required, this must be Num(4).

Sample Rows

NDC	GenericName	GenericNameKey	DrugClass	DrugClassKey
00002323030	OLANZAPINE/FLUOXETINE HCL	2130	Antidepressant Combinations	70
00002751601	INSULIN LISPRO	1515	Injectable Antidiabetic Agents	373
50428167114	HYDROCORTISONE	1393	Dermatological - Glucocorticoid	257
15330018801	LEVOTHYROXINE SODIUM	1684	Thyroid Hormones and Combinations	594
62856027630	PERAMPANEL	2299	Anticonvulsant - AMPA-Type Glutamate Receptor Antagonists	43
99207051122	FLUOCINONIDE	1180	Dermatological - Glucocorticoid	258