ADE Prevention and Monitoring: Warfarin Time in Therapeutic Range
Supplemental SQL Logic Reference
(CMS179, version 2, updated 5/15/13)

The purpose of this document is to support the implementation of the clinical quality measure "ADE Prevention and Monitoring: Warfarin Time in Therapeutic Range" by providing an example of the structured query language (SQL) that underwent field testing. The defined SQL logic below provides a full view of its content, but the specifications supplied in the header section of the Health Quality Measure Format (HQMF) of the clinical quality measure should be the primary basis for implementation of the measure. The HQMF files for this clinical quality measure contain instructions in the Definition and Guidance section which indicate the ultimate purpose of the SQL logic defined in this document. Since the SQL implementation may vary depending on an EHR system's table structure and data definitions, EHR system programmers and vendors should replace the field names and table names as needed based on their knowledge of their EHR system and its requirements in order to fulfill the measure's intent.

TTR percentage will be calculated for each patient that meets the criteria for the Measure Population. The average of these values is reported as the Measure Observation.

ADE Prevention and Monitoring
Percent of Time in Therapeutic Range (TTR)

The initial part of the SQL logic calculates the percent TTR for each patient (PctTTR in the temporary table #PatientTTR). Percent of time in therapeutic range (TTR) is calculated within the logic originally developed by the Veterans Affairs (VA).

Warfarin time in therapeutic range is the percentage of time in which patients with atrial fibrillation or flutter who are on chronic warfarin therapy have INR test results within the therapeutic range (2.0 - 3.0) during the measurement period.

The following filters are applied to the INR results prior to the calculation of TTR for each patient:

1) INR value closest to 2.5 when there are more than one INR result on a single date
2) INR values greater than 10 will be replaced with an INR value of 10
3) INR values less than 0.8 are ignored and eliminated from the final TTR calculation for each patient

The logic keeps track of the number of valid INR intervals for each patient. A Valid INR Interval is defined as a pair of INR start dates that are less than or equal to 56 days apart. Patients without 2 such intervals will be excluded from the calculation of the providers' Average PctTTR later on.

Identifiers for the patient's provider and the practice site are also included. The identifier for the provider that is ultimately responsible for warfarin management should be used. The identifier for the practice site at which the patient's warfarin is managed should be used.

```
USE [Datamart_Staging]
GO
/****** Object:  StoredProcedure [dbo].[ADE_TTRCalculationWithFilters]
Script Date: 04/10/2013 09:01:41 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[ADE_TTRCalculationWithFilters]

AS

SET NOCOUNT ON;

SELECT
          Patient_ID,
          Practice_Site,
          provider_ID,
          [QDM_Attribute Result Value],
          ABS(2.5 - [QDM_Attribute Result Value])AS ValDiff,
          DATEADD(DAY,0, DATEDIFF(DAY, 0,[Start DateTime])) AS [Start
DateTime]
INTO
          #LabResults1
FROM    dbo.ADE_LabResults a JOIN
            ADE_VocabularyDictionary b ON a.DataElement_Code = b.Code
WHERE
            b.[QDM Category] = 'Laboratory Test, Result' AND b.[Value Set
Name] = 'INR'
ORDER BY patient_ID, [Start DateTime]
SELECT      Patient_ID,
          [Start DateTime],
          MIN(ValDiff) AS ValDiff
INTO   #LabResults2
FROM   #LabResults1
GROUP BY Patient_ID,
            [Start DateTime]

SELECT  a.Patient_ID,
            Practice_Site,
            Provider_ID,
        a.[Start DateTime],
CASE WHEN a.[QDM_Attribute Result Value] >10 THEN 10 ELSE
a.[QDM_Attribute Result Value] END AS [QDM_Attribute Result Value]

INTO   #FilteredLabResults
FROM   #LabResults1 a
JOIN   #LabResults2 b ON a.Patient_ID = b.Patient_ID
AND    a.[Start DateTime] = b.[Start DateTime]
AND    a.ValDiff = b.ValDiff
WHERE  a.[QDM_Attribute Result Value] >= 0.8
```

```
DROP TABLE #LabResults2
DROP TABLE #LabResults1


SELECT
            Patient_ID,
            Practice_Site,
            Provider_ID,
            [QDM_Attribute Result Value],
            [Start datetime],
            RANK () OVER (PARTITION BY Patient_ID ORDER BY [Start
datetime]) AS INROrder
INTO
          #OrderedINRList
FROM    #FilteredLabResults

ORDER BY
          [Start datetime]

DECLARE          @INRLowerBound AS DECIMAL(20,4)
SET              @INRLowerBound = 2.0
DECLARE          @INRUpperBound AS DECIMAL(20,4)
SET              @INRUpperBound = 3.0

SELECT
            Patient_ID,
            Practice_Site,
            Provider_ID,
            INROrder,
            INR1Date,
            INR1Result,
            TimeBetweenSamples,
            INRDiff,
            INRShiftKPI2,
            IsValidInterval,
CASE
      WHEN
            INRShiftKPI2 = 0.0 AND (INR1Result >= @INRLowerBound AND
INR1Result <= @INRUpperBound
            AND INR2Result >= @INRLowerBound AND INR2Result <=
@INRUpperBound)THEN CAST(TimeBetweenSamples AS DECIMAL)
      ELSE isnull(cast(TimeBetweenSamples AS DECIMAL) * ABS((INRShiftKPI2
/ NULLIF(INRDiff,0))),0)
      END  AS TherapeuticDaysKPI2

INTO
      #TherapeuticDays
FROM
(
      SELECT
            inr1.Patient_ID,
            inr1.Practice_Site,
            inr1.Provider_ID,
```

```
            inr1.INROrder,
            inr1.[Start datetime] AS INR1Date,
            inr1.[QDM_Attribute Result Value] AS INR1Result,
            inr2.[Start datetime] AS INR2Date,
            inr2.[QDM_Attribute Result Value] AS INR2Result,
            DATEDIFF(DAY,inr1.[Start datetime],inr2.[Start datetime]) AS
TimeBetweenSamples,
            inr2.[QDM_Attribute Result Value] - inr1.[QDM_Attribute
Result Value] AS INRDiff,
            dbo.DifferenceWithinRange_v2  (inr1.[QDM_Attribute Result
Value],inr2.[QDM_Attribute Result Value],@INRLowerBound,@INRUpperBound)
AS INRShiftKPI2,
            CASE
                  WHEN (ABS(DATEDIFF(DAY,inr1.[Start datetime],inr2.[Start
datetime])) <= 56)
                        THEN 1
                        ELSE 0
            END AS IsValidInterval
      FROM
            #OrderedINRList inr1
            INNER JOIN #OrderedINRList inr2
            ON inr2.INROrder = inr1.INROrder + 1 AND inr1.Patient_ID =
inr2.Patient_ID
      WHERE
            inr2.[Start datetime] >= inr1.[Start datetime]
) x
ORDER BY
      INR1Date
SELECT
      Patient_ID ,
      Practice_Site,
      Provider_ID,
      ROUND(100 * (SUM(TherapeuticDaysKPI2) / SUM(TimeBetweenSamples)),2)
AS PctTTR,
      SUM(IsValidInterval) as NumValidIntervals
INTO
      #PatientTTR
FROM
      #TherapeuticDays

GROUP BY Patient_ID,Practice_Site, Provider_ID
ORDER BY Patient_ID


DROP TABLE #FilteredLabResults
DROP TABLE #TherapeuticDays
DROP TABLE #OrderedINRList
```

Cumulative Medication Duration

Cumulative medication duration (CMD) includes the total number of calendar days the patient is actively using Warfarin. The SQL logic below does not include the specific medication codes that are used to identify each individual warfarin prescription for a patient. In the HQMF file for the clinical quality measure, the value set for the data element Medication, Active "Warfarin" contains the RxNorm codes that should be used to identify patients on warfarin therapy. The HQMF file for the clinical quality measure also defines cumulative medication duration >=180 days.
For testing purposes, the measurement start date was set to 1/1/2011, and the look-back period for an active medication of warfarin is 200 days prior to measurement start date. Depending on how cumulative medication duration is captured in the site's EHR, SQL logic may need to be modified in order to include this particular data set.

```
DECLARE @MeasurementStartDate DATETIME
DECLARE @LookBackDate DATETIME

SET @MeasurementStartDate = '1/1/2011'
SET @LookBackDate = @MeasurementStartDate - 200

SELECT DISTINCT a.Patient_ID,
                      a.Practice_Site,
                      a.Provider_ID,
                      a.PctTTR,
                      B.[Start DateTime],
                      B.[Stop DateTime],
                      DATEDIFF(DAY,B.[Start DateTime] , B.[Stop
DateTime]) AS DateDifference,
                      CASE WHEN  b.[Start DateTime] < @LookBackDate THEN
DATEDIFF(DAY,B.[Start DateTime] ,
                                        B.[Stop DateTime]) -
DATEDIFF(DAY,B.[Start DateTime] , @LookBackDate)
                      WHEN       b.[Stop DateTime] >=
@MeasurementStartDate THEN DATEDIFF(DAY,B.[Start DateTime] ,
                                        B.[Stop DateTime]) -
DATEDIFF(DAY,@MeasurementStartDate,B.[Stop DateTime])
                      ELSE DATEDIFF(DAY,B.[Start DateTime] , B.[Stop
DateTime])  END AS ActualUsageIn200DayPeriod
INTO             #PatientTTRWithMedDates
FROM             #PatientTTR A
      JOIN       ADE_Medications B ON A.Patient_ID = B.Patient_ID
      JOIN       ADE_VocabularyDictionary C ON B.DataElement_Code =
C.Code
WHERE            B.[Start DateTime] IS NOT NULL
                      AND   (b.[Start DateTime] >= @LookBackDate or
b.[Stop DateTime] >= @LookBackDate)
                      AND (b.[Start DateTime] <= @MeasurementStartDate)
                      AND C.[QDM Category] = 'Medication, Active'

SELECT      Patient_ID,
```

```
                 Practice_Site,
                 Provider_ID,
                 PctTTR,
                 SUM(ActualUsageIn200DayPeriod) AS
CumlativeMedicationUsage
INTO        #PatientTTRWithMin180DaysMeds
FROM        #PatientTTRWithMedDates
GROUP BY    Patient_ID,Practice_Site, Provider_ID,PctTTR
HAVING      SUM(ActualUsageIn200DayPeriod) >=180
ORDER BY    Patient_ID,Practice_Site, Provider_ID
```

Age Requirements

The logic in this section contains a filter that states the patient must be 18 years or older during the measurement period.

```
SELECT  a.Patient_id,
           b.BirthDate,
           a.Practice_Site,
           a.Provider_ID,
           a.PctTTR
INTO   #PatientTTRAbove18WithMin180DaysMeds
FROM   #PatientTTRWithMin180DaysMeds a
JOIN   ADE_Patients B ON a.Patient_ID = b.Patient_ID
WHERE  DATEDIFF(YEAR,b.birthdate,@MeasurementStartDate) >=18
ORDER BY A.Practice_Site
```

Active Diagnosis (including exclusion criteria)

Atrial Fibrillation Diagnosis

Patients who have an active diagnosis of atrial fibrillation or atrial flutter that started and did not end before the first day of the measurement period must be included in this measure.

Valvular Heart Disease

If patients contain an active diagnosis of valvular heart disease that started and did not end before the start of the measurement period, they should be excluded from the data set.

```
SELECT   a.Patient_Id,
            a.BirthDate,
            a.Practice_Site,
            a.Provider_ID,
            a.PctTTR
INTO     #PatientTTRAbove18WithMin180DaysMedsAndDiagnosis
```

```
FROM     #PatientTTRAbove18WithMin180DaysMeds a JOIN
            ADE_Diagnosis b ON a.Patient_id =B.Patient_ID
WHERE
            b.[start DateTime] < @MeasurementStartDate
            AND b.[Stop DateTime] > @MeasurementStartDate
            AND b.DataElement_Code IN (SELECT CODE FROM
ADE_VocabularyDictionary WHERE [QDM Category] = 'Diagnosis, Active' AND
([Value Set Name] = 'Atrial Fibrillation/Flutter'))
            AND b.Patient_id NOT IN (SELECT Patient_Id FROM
ADE_Diagnosis where (DataElement_Code IN (SELECT CODE FROM
ADE_VocabularyDictionary WHERE [QDM Category] = 'Diagnosis, Active' AND
([Value Set Name] = 'Valvular Heart Disease'))))
            AND (b.[start DateTime] <= @MeasurementStartDate AND b.[Stop
DateTime] >= @MeasurementStartDate)

ORDER BY A.Patient_ID

DROP TABLE #PatientTTRAbove18WithMin180DaysMeds
```

Valid INR Intervals

The SQL logic below calculates patients who have at least two valid INR intervals during the measurement period. A valid INR interval is defined as a pair of INR results that are less than or equal to 56 days apart. If multiple INR results are present on the same day, only one is noted for the TTR calculation (filter mentioned in Percent TTR section).

```
SELECT    A.Patient_Id,
             A.BirthDate,
             A.Practice_Site,
             A.Provider_ID,
             A.PctTTR
INTO      #PatientsWithTwoValidIntervals
FROM      #PatientTTRAbove18WithMin180DaysMedsAndDiagnosis A
JOIN    #PatientTTR B ON A.Patient_ID = B.Patient_ID
WHERE
           B.NumValidIntervals >= 2
```

Encounter Data

The logic below includes patients that have at least one outpatient visit during the measurement period. Patient encounter codes and definitions are site specific and must capture the relative encounters needed to meet the criteria of the measure.

```
SELECT    a.Patient_Id,
             a.BirthDate,
             a.Practice_Site,
             a.Provider_ID,
             a.PctTTR
INTO      #PatientTTRWithDaysAgeDiagnosisEncounter
FROM      #PatientsWithTwoValidIntervals a
JOIN      ADE_Encounters B ON A.Patient_Id = b.Patient_ID
JOIN    ADE_VocabularyDictionary C ON B.DataElement_Code = C.Code
WHERE
             (C.[Value Set Name] = 'Face-to-Face Interaction' OR
C.[Value Set Name] = 'Office Visit')
           AND b.[start datetime] >= @MeasurementStartDate
ORDER BY  Practice_site
```

Average TTR by Provider and Practice

In order to calculate an AverageTTR by provider, patients who meet all the criteria above will be grouped by unique provider identifier. The provider IDs should be assigned by the site (e.g., actual provider identifier). The identifier for the provider that is ultimately responsible for warfarin management should be used.

Note: the logic also includes the calculation of AverageTTR by practice site (e.g., an anticoagulation clinic). This is for reference purposes only and is not required for the quality measure or its reporting. Ideally, the identifier for the practice site at which the patient's warfarin is managed should be used.

```
SELECT Practice_Site,AVG(PctTTR) AS AvgTTRByPracticeSite
FROM #PatientTTRWithDaysAgeDiagnosisEncounter
GROUP BY Practice_Site

SELECT Provider_ID,AVG(PctTTR) AS AvgTTRByProvider
FROM #PatientTTRWithDaysAgeDiagnosisEncounter
GROUP BY Provider_ID
```

FUNCTION [dbo].[DifferenceWithinRange_v2]

The following function is required for the calculation of TTR. This function calculates the difference between two numbers that falls within a specified range. For example, given a range of 2.0 to 3.0, the difference between 1.5 and 2.5 within this range is 0.5. The function is intended for use in calculating differences between INR values within the context of the Rosendaal method of calculating TTR (time in therapeutic range), which requires the proportion of an INR difference from one sample to the next that falls within the therapeutic range.

```
USE [V01DW]

GO



/****** Object:  UserDefinedFunction [dbo].[DifferenceWithinRange_v2]
Script Date: 01/03/2013 13:51:42 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE FUNCTION [dbo].[DifferenceWithinRange_v2]
(     --inputs:
      @Val1 as decimal(10,5),
      @Val2 as decimal(10,5),
      @LowerBound as decimal(10,5),
      @UpperBound as decimal(10,5)
)
RETURNS decimal(10,5)
AS
BEGIN
      -- Declare the return variable here
      DECLARE @result as decimal(10,5)


      set @result =
      (
      SELECT
            case
      -- inr values are both outside the range in the same direction
            when @Val1 > @UpperBound and @Val2 > @UpperBound then null
            when @Val1 < @LowerBound and @Val2 < @LowerBound then null
      -- inr values are straddling the range
            when (@Val1 > @UpperBound and @Val2 < @LowerBound)
                   OR (@Val2 > @UpperBound and @Val1 < @LowerBound)

                  then @UpperBound - @LowerBound
      -- both inr values are within the range
            when @Val1 between @LowerBound and @UpperBound
```

```
                  and @Val2 between @LowerBound and @UpperBound
            then (@Val2 - @Val1)
      -- one value is in the range and one is outside
            when @Val1 > @Val2
                  and @Val1 > @UpperBound
            then (@UpperBound - @Val2)*(-1) --/ (@Val1 - @Val2)
            when @Val2 > @Val1
                  and @Val2 > @UpperBound
            then (@UpperBound - @Val1)*(-1) --/ (@Val2 - @Val2)

            when @Val1 > @Val2
                  and @Val2 < @LowerBound
            then (@Val1 - @LowerBound)*(-1)
            when @Val2 > @Val1
                  and @Val1 < @LowerBound
            then (@Val2 - @LowerBound)*(-1)
            else null
      end
      )

      -- Return the result of the function
      RETURN @result

END
GO
```