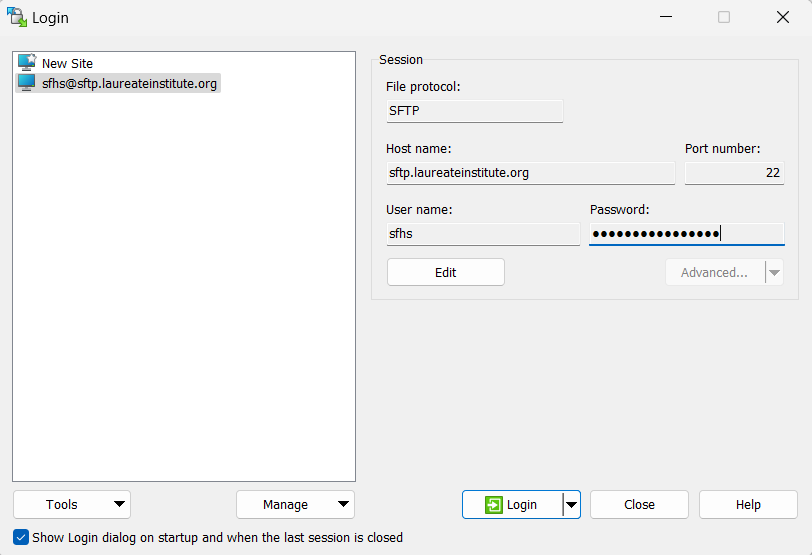
The LIBR Treatment Resistant Depression (TRD) study involves 6 queries: one pulled from Clarity, and the other from Caboodle. This documentation refers to the Caboodle SHFS dataset. There are 6 separate queries to pull, and the file naming convention is as follows:

* Person\_Table-YY\_MM\_DD-vX
* Encounter\_Table-YY\_MM\_DD-vX
* Diagnosis\_Table-YY\_MM\_DD-vX
* Medication\_Table-YY\_MM\_DD-vX
* Procedure\_Table-YY\_MM\_DD\_vX
* RXNorm\_Table-YY\_MM\_DD\_vX (Clarity)

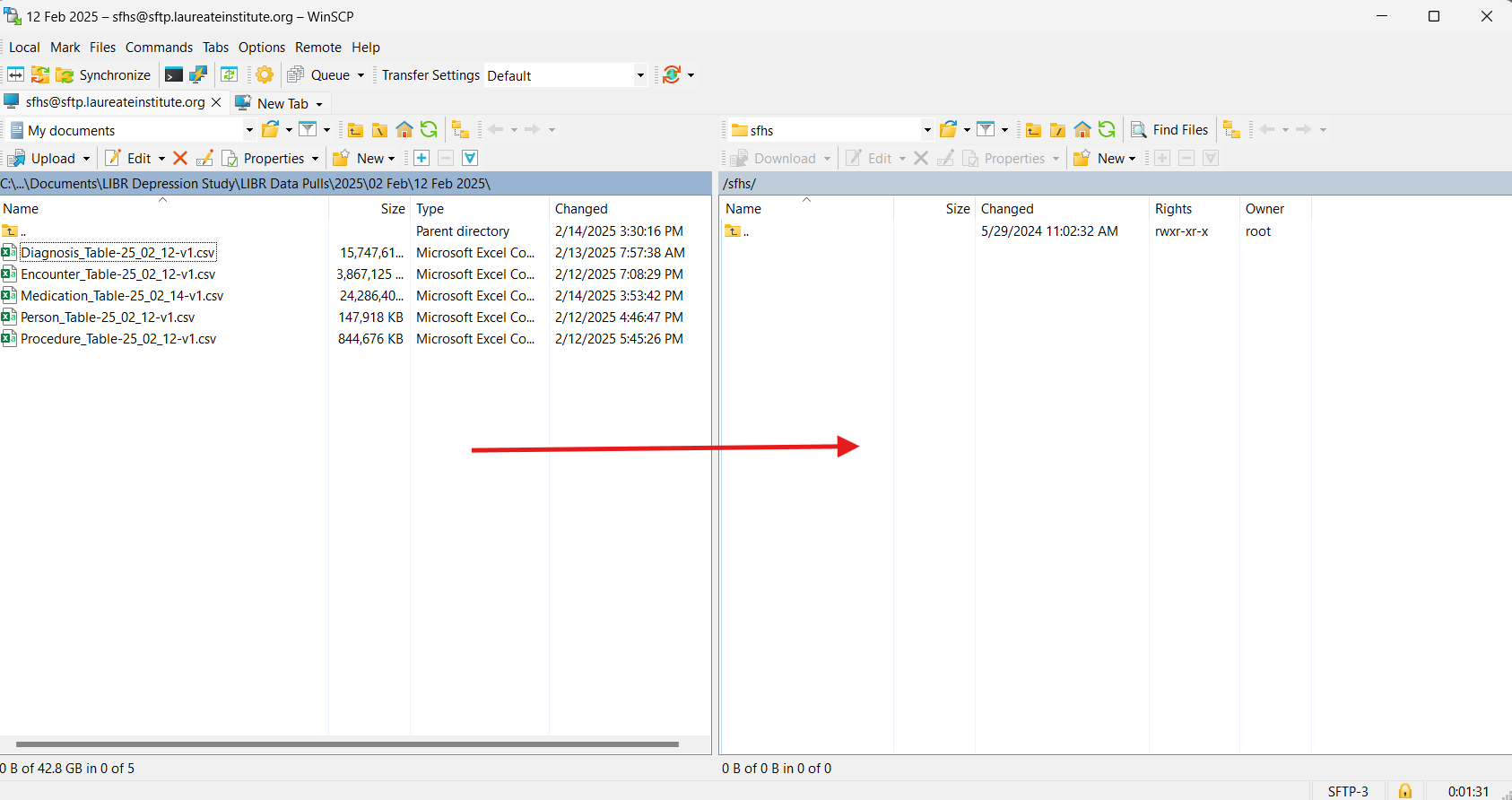
This naming convention allows the file recipients to clearly distinguish which table the file includes, the date the file was sent, and the file version (vX could be v1, v2, etc).

Data Upload - SFTP

The following file upload process applies to all data tables transferred to LIBR. **Credentials can be shared to individuals upon request.** WINSCP is the preferred file transfer software.



Once logged in, simply move the files over that you want to transfer. The LIBR server routinely looks for new files and moves them to a folder that LIBR contacts can access. When this occurs, the file will leave the folder shown in WINSCP.



Person Table

The Person table is limited by the following statements in the initial WHERE clause.

* Service Area = 10
* Patient is NOT a test patient or a historical patient
* Patient IS current and valid
* Patient is NOT deceased
* Patient’s current age is between 18 and 110 years.

The PatientDim table has 5 columns that list a patient’s various races. A crossapply function was used to pivot these values into columns easier to manipulate in a data model. The new columns are as follows:

* American\_Indian\_Or\_Alaska\_Native
* Asian
* Black\_Or\_African\_American
* Native\_Hawaiian\_Or\_Other\_Pacific\_Islander
* White\_Or\_Caucasian
* Unknown\_Race

The patient ethnicity column was also put through a crossapply function for the same purpose. The new columns are as follows:

* Not\_Hispanic\_Or\_Latino
* Hispanic\_Or\_Latino
* Unknown\_Ethnicity

The patient table also has 3 flags that identify a patient as having a Depression, Bipolar, or Schizophrenia diagnosis. These flags are based on both ICD9 and ICD10 codes provided by LIBR.

* Depression:
  + ICD9 Codes: 296.2, 296.3, 311, 300.4
  + ICD10 Codes: F32%, F33%, F34.1
* Bipolar:
  + ICD9 Codes: 296.0, 296.4, 296.5, 296.6, 296.7, 296.8
  + ICD10 Codes: F30%, F31%
* Schizophrenia:
  + ICD9 Codes: 295%, 298%
  + ICD10 Codes: F20%, F23%, F25%, F28%, F29%

The end of this query also generates a 4 to 1 sampling of the depression group vs. the control group. This is

**SQL Query:**

DROP TABLE IF EXISTS #PERSON\_TABLE;

WITH All\_Patients AS (

SELECT DISTINCT

pd.PatientEpicId,

pd.DurableKey,

pd.ProblemComboKey,

pd.PrimaryCareProviderKey,

pd.Name,

pd.EnterpriseId,

pd.PrimaryMRN,

pd.Sex,

pd.PreferredLanguage,

pd.Ethnicity,

pd.FirstRace,

pd.SecondRace,

pd.ThirdRace,

pd.FourthRace,

pd.FifthRace,

pd.BirthDate,

pd.AgeInYears,

pd.Status,

pd.PostalCode,

pd.SexualOrientation,

pd.MaritalStatus,

pd.Religion,

pd.SmokingStatus,

ppl.ServiceAreaEpicId,

ppl.ServiceAreaName

FROM PatientDim pd

INNER JOIN (

SELECT

ppl.PatientDurableKey,

dd.ServiceAreaEpicId,

dd.ServiceAreaName

FROM

PatientPrimaryLocationEventFact ppl

LEFT OUTER JOIN DepartmentDim dd on ppl.PrimaryLocationKey = dd.DepartmentKey

GROUP BY

ppl.PatientDurableKey,

dd.ServiceAreaEpicId,

dd.ServiceAreaName

) ppl on pd.DurableKey = ppl.PatientDurableKey

WHERE

-- not a test patient

pd.Test = 0

-- SF Service Area

AND ppl.ServiceAreaEpicId = '10'

-- current and valid, not historical

AND pd.IsCurrent = 1

AND pd.IsValid = 1

AND pd.IsHistoricalPatient = 0

AND (pd.AgeInYears >= 18 AND pd.AgeInYears <= 110)

AND pd.Status NOT IN ('Deceased')

),

Mapped\_Race\_Ethnicity AS (

SELECT

PatientEpicId,

CASE

WHEN Race IN ('American Indian or Alaska Native') THEN 'American Indian or Alaska Native'

WHEN Race IN ('Asian') THEN 'Asian'

WHEN Race IN ('Black or African American') THEN 'Black or African American'

WHEN Race IN ('Native Hawaiian or Other Pacific Islander') THEN 'Native Hawaiian or Other Pacific Islander'

WHEN Race IN ('White or Caucasian') THEN 'White or Caucasian'

WHEN Race IN ('No Race', 'Other', 'Patient Refused', 'Unknown') THEN 'Unknown'

END AS MappedRace,

CASE

WHEN Ethnicity IN ('Not Hispanic or Latino') THEN 'Not Hispanic or Latino'

WHEN Ethnicity IN ('Hispanic or Latino') THEN 'Hispanic or Latino'

WHEN Ethnicity IN ('\*Unspecified', 'Patient Refused', 'Unknown') THEN 'Unknown'

END AS MappedEthnicity

FROM All\_Patients

CROSS APPLY (VALUES

(FirstRace),

(SecondRace),

(ThirdRace),

(FourthRace),

(FifthRace)

) AS RaceColumns(Race)

),

Pivoted\_Race\_Ethnicity AS (

SELECT

PatientEpicId,

MAX(CASE WHEN MappedRace = 'American Indian or Alaska Native' THEN 1 ELSE 0 END) AS American\_Indian\_Or\_Alaska\_Native,

MAX(CASE WHEN MappedRace = 'Asian' THEN 1 ELSE 0 END) AS Asian,

MAX(CASE WHEN MappedRace = 'Black or African American' THEN 1 ELSE 0 END) AS Black\_Or\_African\_American,

MAX(CASE WHEN MappedRace = 'Native Hawaiian or Other Pacific Islander' THEN 1 ELSE 0 END) AS Native\_Hawaiian\_Or\_Other\_Pacific\_Islander,

MAX(CASE WHEN MappedRace = 'White or Caucasian' THEN 1 ELSE 0 END) AS White\_Or\_Caucasian,

MAX(CASE WHEN MappedRace = 'Unknown' THEN 1 ELSE 0 END) AS Unknown\_Race,

MAX(CASE WHEN MappedEthnicity = 'Not Hispanic or Latino' THEN 1 ELSE 0 END) AS Not\_Hispanic\_Or\_Latino,

MAX(CASE WHEN MappedEthnicity = 'Hispanic or Latino' THEN 1 ELSE 0 END) AS Hispanic\_Or\_Latino,

MAX(CASE WHEN MappedRace = 'Unknown' THEN 1 ELSE 0 END) AS Unknown\_Ethnicity

FROM Mapped\_Race\_Ethnicity

GROUP BY PatientEpicId

),

Depression\_Patients AS (

SELECT DISTINCT

ap.PatientEpicId

FROM All\_Patients ap

LEFT OUTER JOIN DiagnosisBridge db ON ap.ProblemComboKey = db.DiagnosisComboKey

LEFT OUTER JOIN DiagnosisDim dd ON db.DiagnosisKey = dd.DiagnosisKey

LEFT OUTER JOIN DiagnosisTerminologyDim dtd ON dd.DiagnosisKey = dtd.DiagnosisKey

WHERE

--patients in depression population

(dtd.Type = 'ICD-10-CM' AND (dtd.Value LIKE 'F33%' OR dtd.Value LIKE 'F32%' OR dtd.Value LIKE 'F34.1' ))

OR

(dtd.Type = 'ICD-9-CM' AND (dtd.Value IN ('296.2', '296.3', '311', '300.4')))

),

Bipolar\_Patients AS (

SELECT DISTINCT

ap.PatientEpicId

FROM All\_Patients ap

LEFT OUTER JOIN DiagnosisBridge db ON ap.ProblemComboKey = db.DiagnosisComboKey

LEFT OUTER JOIN DiagnosisDim dd ON db.DiagnosisKey = dd.DiagnosisKey

LEFT OUTER JOIN DiagnosisTerminologyDim dtd ON dd.DiagnosisKey = dtd.DiagnosisKey

WHERE

--patients in bipolar population

(dtd.Type = 'ICD-10-CM' AND (dtd.Value LIKE 'F30%' OR dtd.Value LIKE 'F31%'))

OR

(dtd.Type = 'ICD-9-CM' AND (dtd.Value IN ('296.0', '296.4', '296.5', '296.6', '296.7', '296.8')))

),

Schizophrenia\_Patients AS (

SELECT DISTINCT

ap.PatientEpicId

FROM All\_Patients ap

LEFT OUTER JOIN DiagnosisBridge db ON ap.ProblemComboKey = db.DiagnosisComboKey

LEFT OUTER JOIN DiagnosisDim dd ON db.DiagnosisKey = dd.DiagnosisKey

LEFT OUTER JOIN DiagnosisTerminologyDim dtd ON dd.DiagnosisKey = dtd.DiagnosisKey

WHERE

--patients in schizophrenia population

(dtd.Type = 'ICD-10-CM' AND (dtd.Value LIKE 'F20%' OR dtd.Value LIKE 'F23%' OR dtd.Value LIKE 'F25%' OR dtd.Value LIKE 'F28%' OR dtd.Value LIKE 'F29%' ))

OR

(dtd.Type = 'ICD-9-CM' AND (dtd.Value LIKE '295%' OR dtd.value LIKE '298%'))

)

--------Data Pull----------------

SELECT DISTINCT

CONVERT(VARCHAR(32), HASHBYTES('MD5', ap.PatientEpicId), 2) AS ShortHashedPatientEpicId,

CASE WHEN dp.PatientEpicID IS NOT NULL THEN 1 ELSE 0 END as 'DepressionDiagnosis',

CASE WHEN bp.PatientEpicID IS NOT NULL THEN 1 ELSE 0 END as 'BipolarDiagnosis',

CASE WHEN sp.PatientEpicID IS NOT NULL THEN 1 ELSE 0 END as 'SchizophreniaDiagnosis',

ap.Sex,

ap.PreferredLanguage,

pre.American\_Indian\_Or\_Alaska\_Native,

pre.Asian,

pre.Black\_Or\_African\_American,

pre.Native\_Hawaiian\_Or\_Other\_Pacific\_Islander,

pre.White\_Or\_Caucasian,

pre.Unknown\_Race,

pre.Not\_Hispanic\_Or\_Latino,

pre.Hispanic\_Or\_Latino,

pre.Unknown\_Ethnicity,

ap.AgeInYears,

ap.PostalCode,

ap.SexualOrientation,

ap.MaritalStatus,

ap.Religion,

ap.SmokingStatus,

ap.ServiceAreaName

INTO #PERSON\_TABLE

FROM All\_Patients ap

LEFT OUTER JOIN Depression\_Patients dp ON ap.PatientEpicId = dp.PatientEpicId

LEFT OUTER JOIN Bipolar\_Patients bp ON ap.PatientEpicId = bp.PatientEpicId

LEFT OUTER JOIN Schizophrenia\_Patients sp ON ap.PatientEpicId = sp.PatientEpicId

LEFT OUTER JOIN Pivoted\_Race\_Ethnicity pre ON ap.PatientEpicId = pre.PatientEpicId

**Data Cleaning:**

This is the requested data cleaning by LIBR. The code replaces specific values with NULL, or ‘NULL,’ depending on if the column allows for NULL values.

DECLARE @sql NVARCHAR(MAX) = '';

SELECT @sql = @sql +

'UPDATE #PERSON\_TABLE SET [' + name + '] = ''NULL''

WHERE [' + name + '] = ''''

OR [' + name + '] LIKE ''\*Unspecified''

OR [' + name + '] LIKE ''\*Unknown''

OR [' + name + '] LIKE ''Unknown''

OR [' + name + '] LIKE ''Never Assessed''

OR [' + name + '] LIKE ''\*Not Applicable'';' + CHAR(10)

FROM tempdb.sys.columns

WHERE object\_id = OBJECT\_ID('tempdb..#PERSON\_TABLE')

AND name IN ('Sex', 'PreferredLanguage', 'PostalCode','SexualOrientation','MaritalStatus','Religion','SmokingStatus'); -- Only include these 7 columns, which don't allow nulls

EXEC sp\_executesql @sql;

SELECT @sql = @sql +

'UPDATE #PERSON\_TABLE SET [' + name + '] = NULL

WHERE [' + name + '] = ''''

OR [' + name + '] LIKE ''\*Unspecified''

OR [' + name + '] LIKE ''\*Unknown''

OR [' + name + '] LIKE ''Unknown''

OR [' + name + '] LIKE ''Never Assessed''

OR [' + name + '] LIKE ''\*Not Applicable'';' + CHAR(10)

FROM tempdb.sys.columns

WHERE object\_id = OBJECT\_ID('tempdb..#PERSON\_TABLE') -- Get columns for the temp table

AND system\_type\_id IN (167, 175, 231, 239); -- Filter for text-based columns (varchar, char, nvarchar, nchar)

EXEC sp\_executesql @sql;

**4:1 Patient Sampling:**

DROP TABLE IF EXISTS #CG\_TABLE;

DROP TABLE IF EXISTS #DG\_TABLE;

DROP TABLE IF EXISTS ##Final\_4to1\_Sample;

SELECT

\*,

ROW\_NUMBER() OVER (ORDER BY NEWID()) AS RandomOrder

INTO #CG\_TABLE

FROM #PERSON\_TABLE

WHERE DepressionDiagnosis = 0

----

SELECT

\*,

ROW\_NUMBER() OVER (ORDER BY NEWID()) AS RandomOrder

INTO #DG\_TABLE

FROM #PERSON\_TABLE

WHERE DepressionDiagnosis = 1

----------------------------------------

-- 1) Get current row counts

----------------------------------------

DECLARE @CG\_Count INT = (SELECT COUNT(\*) FROM #CG\_TABLE);

DECLARE @DG\_Count INT = (SELECT COUNT(\*) FROM #DG\_TABLE);

----------------------------------------

-- 2) Compute how many rows we can sample to keep 4:1

----------------------------------------

DECLARE @DG\_Rows INT = (

SELECT CASE

WHEN @DG\_Count <= @CG\_Count / 4.0 THEN @DG\_Count

ELSE FLOOR(@CG\_Count / 4.0)

END

);

DECLARE @CG\_Rows INT = @DG\_Rows \* 4;

DECLARE @TotalRows INT = @DG\_Rows + @CG\_Rows;

----------------------------------------

-- 3) Pull that many rows from each table, randomizing

-- via ORDER BY NEWID()

----------------------------------------

WITH

CG\_Sample AS (

SELECT

TOP (@CG\_Rows)

\*, -- all columns from #CG\_TABLE

'CG\_TABLE' AS SourceTable

FROM #CG\_TABLE

ORDER BY NEWID()

),

DG\_Sample AS (

SELECT

TOP (@DG\_Rows)

\*, -- all columns from #DG\_TABLE

'DG\_TABLE' AS SourceTable

FROM #DG\_TABLE

ORDER BY NEWID()

)

SELECT \*

INTO ##Final\_4to1\_Sample

FROM CG\_Sample

UNION ALL

SELECT \*

FROM DG\_Sample;

-- ##Final\_4to1\_Sample will have an extra column named "SourceTable" indicating whether each row came from #CG\_TABLE or #DG\_TABLE.

----------------------------------------

-- Check row counts ---

----------------------------------------

SELECT @CG\_Rows AS [# from CG\_TABLE],

@DG\_Rows AS [# from DG\_TABLE],

@TotalRows AS [Total Rows in 4to1 Sample];

**Data Export:**

Run a simple select query, then save the results to a .csv file with the naming conventions specified at the beginning of this document.

SELECT

PatientEpicId\_SH,

DepressionDiagnosis,

BipolarDiagnosis,

SchizophreniaDiagnosis,

Sex,

PreferredLanguage,

American\_Indian\_Or\_Alaska\_Native,

Asian,

Black\_Or\_African\_American,

Native\_Hawaiian\_Or\_Other\_Pacific\_Islander,

White\_Or\_Caucasian,

Unknown\_Race,

Not\_Hispanic\_Or\_Latino,

Hispanic\_Or\_Latino,

Unknown\_Ethnicity,

AgeInYears,

BirthYear,

PostalCode,

SexualOrientation,

MaritalStatus,

Religion,

SmokingStatus,

RandomOrder,

CASE WHEN SourceTable = 'CG\_TABLE' THEN 'Control Group' ELSE 'Depression Group' END as 'GroupType'

FROM ##Final\_4to1\_Sample

Encounter Table

The Encounter table is limited by the following statements in the initial WHERE clause. It is also limited to encounters only for patients in the Patient table.

* Encounter Status = 10
* Encounter date is NOT unknown, and is less than or equal to today’s date.
* Patient class is one of the following: Inpatient, Outpatient, Observation, Emergency
* Encounter has at least one diagnosis.

##Final\_4to1\_Sample is a temporary table generated from the TRD\_Person\_Table-Current file. It exists in the global environment and can be run across files in a single session.

**SQL Query:**

Use Caboodle;

DROP TABLE IF EXISTS ##ENCOUNTER\_TABLE;

SELECT DISTINCT

ap.PatientEpicId,

ap.PatientEpicId\_SH,

CONVERT(VARCHAR(32), HASHBYTES('MD5', CAST(ef.EncounterKey AS VARBINARY(MAX))), 2) AS EncounterId\_SH,

ef.EncounterKey,

ef.EncounterEpicCSN,

ef.Type as PatientType,

ef.PatientClass,

ef.Date as StartVisit,

ef.EndInstant as EndVisit,

ef.DepartmentKey,

dd.DepartmentName,

ef.PlaceOfServiceKey,

pos.Name as PlaceOfServiceName,

pd.Name as ProviderName,

ef.PrimaryDiagnosisKey,

CASE WHEN ap.SourceTable = 'CG\_TABLE' THEN 'Control Group' ELSE 'Depression Group' END as 'GroupType'

INTO ##ENCOUNTER\_TABLE

FROM ##Final\_4to1\_Sample ap

INNER JOIN EncounterFact ef on ef.PatientDurableKey = ap.DurableKey

LEFT OUTER JOIN DepartmentDim dd on ef.DepartmentKey = dd.DepartmentKey

LEFT OUTER JOIN PlaceOfServiceDim pos on ef.PlaceOfServiceKey = pos.PlaceOfServiceKey

LEFT OUTER JOIN ProviderDim pd on ef.ProviderKey = pd.ProviderKey

WHERE

ef.DateKey NOT IN (-1)

AND ef.DerivedEncounterStatus = 'Complete'

AND ef.Date <= CAST(GETDATE() AS DATE) -- exclude future encounters

AND ef.PatientClass IN ('Inpatient','Outpatient','Observation','Emergency')

AND ef.PrimaryDiagnosisKey > 0

--------- RUN FOR DATA PULL ------------

DROP TABLE IF EXISTS #ENCOUNTER\_TABLE2;

SELECT

PatientEpicId\_SH,

EncounterId\_SH,

PatientType,

PatientClass,

StartVisit,

EndVisit,

DepartmentKey,

DepartmentName,

PlaceOfServiceKey,

PlaceOfServiceName,

ProviderName,

GroupType

INTO #ENCOUNTER\_TABLE2

FROM

##ENCOUNTER\_TABLE et

GROUP BY

PatientEpicId\_SH,

EncounterId\_SH,

PatientType,

PatientClass,

StartVisit,

EndVisit,

DepartmentKey,

DepartmentName,

PlaceOfServiceKey,

PlaceOfServiceName,

ProviderName,

GroupType

**Data Cleaning:**

This is the requested data cleaning by LIBR. The code replaces specific values with NULL.

DECLARE @sql NVARCHAR(MAX) = '';

SELECT @sql = @sql +

'UPDATE #ENCOUNTER\_TABLE2 SET [' + name + '] = NULL

WHERE [' + name + '] = ''''

OR [' + name + '] LIKE ''\*Unspecified''

OR [' + name + '] LIKE ''\*Unknown''

OR [' + name + '] LIKE ''NA''

OR [' + name + '] LIKE ''N/A''

OR [' + name + '] LIKE ''\*Not Applicable'';' + CHAR(10)

FROM tempdb.sys.columns

WHERE object\_id = OBJECT\_ID('tempdb..#ENCOUNTER\_TABLE2') -- Get columns for the temp table

AND system\_type\_id IN (167, 175, 231, 239); -- Filter for text-based columns (varchar, char, nvarchar, nchar)

EXEC sp\_executesql @sql;

**Data Export:**

Run a simple select query, then save the results to a .csv file with the naming conventions specified at the beginning of this document.

SELECT \* FROM #ENCOUNTER\_TABLE2

Diagnosis Table

The Diagnosis table is limited by the following statements in the initial WHERE clause. It is also limited to patients and encounters from the Patient and Encounters tables.

* The encounter key in the DiagnosisEventFact table must be greater than 0. This prevents pulling in diagnoses from encounters that may have been deleted from a specific table.

The Diagnosis table also removes duplicate ICD9 and ICD10 codes per encounter, then pivots each ICD9 and ICD10 codes to individual columns per encounter to prevent duplicate rows for each diagnosis per encounter.

##ENCOUNTER\_TABLE is a temporary table generated from the TRD\_Encounter\_Table-Current file. It exists in the global environment and can be run across files in a single session.

**SQL Query:**

Use Caboodle;

-- uses ##ENCOUNTER\_TABLE from global session

----------- DIAGNOSIS BUILD START ------------------------------------------------

DROP TABLE IF EXISTS #DIAGNOSIS\_TABLE;

SELECT

def.DiagnosisEventKey,

def.DiagnosisKey,

def.PatientDurableKey,

ROW\_NUMBER() OVER (

PARTITION BY def.EncounterKey, def.DiagnosisKey, dtd.Value

ORDER BY def.EncounterKey, def.DiagnosisKey, dtd.Value, def.IsPrimary desc

) AS ICD9\_10\_Rank\_1, -- Rank to handle duplicates

def.PatientKey,

ef.PatientEpicId\_SH,

def.EncounterKey,

ef.EncounterId\_SH,

ef.EncounterEpicCSN,

ef.StartVisit as 'Encounter\_StartVisit',

ef.EndVisit as 'Encounter\_EndVisit',

def.IsPrimary,

def.Status,

dd1.DiagnosisEpicId,

dd1.Name as DiagnosisName,

dtd.Type as ICD\_Code\_Type,

dtd.Value as ICD9\_10\_Code,

dtd.DisplayString as ICD9\_10\_Description,

def.UserEnteredDateKey as 'DiagnosisStartDate',

def.UserRemovedDateKey as 'DiagnosisEndDate',

ef.GroupType

INTO #DIAGNOSIS\_TABLE

FROM

DiagnosisEventFact def

INNER JOIN ##ENCOUNTER\_TABLE ef on def.EncounterKey=ef.EncounterKey --Only patients/encounters filtered in previous query

LEFT OUTER JOIN DiagnosisDim dd1 on def.DiagnosisKey = dd1.DiagnosisKey

LEFT OUTER JOIN DiagnosisTerminologyDim dtd on dd1.DiagnosisKey = dtd.DiagnosisKey AND dtd.Type IN ('ICD-10-CM','ICD-9-CM')

WHERE

def.EncounterKey > 0 -- exclude negative keys

GROUP BY

def.DiagnosisEventKey,

def.DiagnosisKey,

def.PatientDurableKey,

def.PatientKey,

ef.PatientEpicId\_SH,

def.EncounterKey,

ef.EncounterId\_SH,

ef.EncounterEpicCSN,

ef.StartVisit,

ef.EndVisit,

def.StartDateKey,

def.EndDateKey,

def.IsPrimary,

def.Status,

dd1.DiagnosisEpicId,

dd1.Name,

dtd.Type,

dtd.Value,

dtd.DisplayString,

def.UserEnteredDateKey,

def.UserRemovedDateKey,

ef.GroupType

-----------------------------------------------------

-- Filter to first occurance of each ICD10 or ICD9 code per diagnosis per encounter

DROP TABLE IF EXISTS #FILTERED\_ICD\_1;

SELECT \*

INTO #FILTERED\_ICD\_1

FROM #DIAGNOSIS\_TABLE

WHERE

ICD9\_10\_Rank\_1 = 1

-----------------------------------------------------

-- Re-rank ICD codes after removing duplicates

DROP TABLE IF EXISTS #FILTERED\_ICD\_2;

SELECT

ROW\_NUMBER() OVER (

PARTITION BY EncounterKey, DiagnosisKey

ORDER BY EncounterKey, DiagnosisKey

) as ICD9\_10\_Rank,

\*

INTO #FILTERED\_ICD\_2

FROM #FILTERED\_ICD\_1

-----------------------------------------------------

DROP TABLE IF EXISTS #DIAGNOSIS\_TABLE1;

SELECT

fd.DiagnosisEventKey,

fd.DiagnosisKey,

fd.PatientDurableKey,

ROW\_NUMBER() OVER (

PARTITION BY fd.EncounterKey

ORDER BY fd.EncounterKey,fd.IsPrimary desc -- puts Primary diagnosis in highest rank(s)

) AS Diagnosis\_Rank,

fd.PatientKey,

fd.PatientEpicId\_SH,

fd.EncounterKey,

fd.EncounterId\_SH,

fd.Encounter\_StartVisit,

fd.Encounter\_EndVisit,

fd.DiagnosisStartDate,

fd.DiagnosisEndDate,

fd.IsPrimary,

fd.Status as DiagnosisStatus,

fd.DiagnosisEpicId,

CASE WHEN CHARINDEX('''', fd.DiagnosisName) > 0 THEN '"' + fd.DiagnosisName + '"' ELSE fd.DiagnosisName END as 'Diagnosis\_Name',

fd.GroupType,

MAX(CASE WHEN ICD9\_10\_Rank = 1 THEN ICD9\_10\_Code END) AS Diagnosis\_1\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 1 THEN ICD\_Code\_Type END) AS Diagnosis\_1\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 1 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description END

END) AS Diagnosis\_1\_Description,

MAX(CASE WHEN ICD9\_10\_Rank = 2 THEN ICD9\_10\_Code END) AS Diagnosis\_2\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 2 THEN ICD\_Code\_Type END) AS Diagnosis\_2\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 2 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description

END

END) AS Diagnosis\_2\_Description,

MAX(CASE WHEN ICD9\_10\_Rank = 3 THEN ICD9\_10\_Code END) AS Diagnosis\_3\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 3 THEN ICD\_Code\_Type END) AS Diagnosis\_3\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 3 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description

END

END) AS Diagnosis\_3\_Description,

MAX(CASE WHEN ICD9\_10\_Rank = 4 THEN ICD9\_10\_Code END) AS Diagnosis\_4\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 4 THEN ICD\_Code\_Type END) AS Diagnosis\_4\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 4 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description

END

END) AS Diagnosis\_4\_Description,

MAX(CASE WHEN ICD9\_10\_Rank = 5 THEN ICD9\_10\_Code END) AS Diagnosis\_5\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 5 THEN ICD\_Code\_Type END) AS Diagnosis\_5\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 5 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description

END

END) AS Diagnosis\_5\_Description,

MAX(CASE WHEN ICD9\_10\_Rank = 6 THEN ICD9\_10\_Code END) AS Diagnosis\_6\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 6 THEN ICD\_Code\_Type END) AS Diagnosis\_6\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 6 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description

END

END) AS Diagnosis\_6\_Description,

MAX(CASE WHEN ICD9\_10\_Rank = 7 THEN ICD9\_10\_Code END) AS Diagnosis\_7\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 7 THEN ICD\_Code\_Type END) AS Diagnosis\_7\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 7 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description

END

END) AS Diagnosis\_7\_Description,

MAX(CASE WHEN ICD9\_10\_Rank = 8 THEN ICD9\_10\_Code END) AS Diagnosis\_8\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 8 THEN ICD\_Code\_Type END) AS Diagnosis\_8\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 8 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description

END

END) AS Diagnosis\_8\_Description,

MAX(CASE WHEN ICD9\_10\_Rank = 9 THEN ICD9\_10\_Code END) AS Diagnosis\_9\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 9 THEN ICD\_Code\_Type END) AS Diagnosis\_9\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 9 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description

END

END) AS Diagnosis\_9\_Description,

MAX(CASE WHEN ICD9\_10\_Rank = 10 THEN ICD9\_10\_Code END) AS Diagnosis\_10\_Code,

MAX(CASE WHEN ICD9\_10\_Rank = 10 THEN ICD\_Code\_Type END) AS Diagnosis\_10\_Vocab,

MAX(CASE WHEN ICD9\_10\_Rank = 10 THEN

CASE WHEN CHARINDEX('''', ICD9\_10\_Description) > 0

THEN '"' + ICD9\_10\_Description + '"'

ELSE ICD9\_10\_Description

END

END) AS Diagnosis\_10\_Description

INTO #DIAGNOSIS\_TABLE1

FROM

#FILTERED\_ICD\_2 fd

GROUP BY

fd.DiagnosisEventKey,

fd.DiagnosisKey,

fd.PatientDurableKey,

fd.PatientKey,

fd.PatientEpicId\_SH,

fd.EncounterKey,

fd.EncounterId\_SH,

fd.Encounter\_StartVisit,

fd.Encounter\_EndVisit,

fd.DiagnosisStartDate,

fd.DiagnosisEndDate,

fd.IsPrimary,

fd.Status,

fd.DiagnosisEpicId,

fd.DiagnosisName,

fd.GroupType

**Data Cleaning:**

This is the requested data cleaning by LIBR. The code replaces specific values with NULL.

DECLARE @sql NVARCHAR(MAX) = '';

SELECT @sql = @sql +

'UPDATE #DIAGNOSIS\_TABLE1 SET [' + name + '] = NULL

WHERE [' + name + '] = ''''

OR [' + name + '] LIKE ''\*Unspecified''

OR [' + name + '] LIKE ''\*Unknown''

OR [' + name + '] LIKE ''NA''

OR [' + name + '] LIKE ''N/A''

OR [' + name + '] LIKE ''\*Not Applicable'';' + CHAR(10)

FROM tempdb.sys.columns

WHERE object\_id = OBJECT\_ID('tempdb..#DIAGNOSIS\_TABLE1') -- Get columns for the temp table

AND system\_type\_id IN (167, 175, 231, 239); -- Filter for text-based columns (varchar, char, nvarchar, nchar)

EXEC sp\_executesql @sql;

**Data Export:**

Run a select query, removing patient identifying information, then save the results to a .csv file with the naming conventions specified at the beginning of this document.

SELECT

d.DiagnosisEventKey,

d.DiagnosisKey,

d.Diagnosis\_Name,

d.PatientEpicId\_SH,

d.EncounterId\_SH,

d.Encounter\_StartVisit,

d.Encounter\_EndVisit,

d.DiagnosisStartDate,

d.DiagnosisEndDate,

d.IsPrimary,

d.DiagnosisStatus,

d.Diagnosis\_1\_Code,

d.Diagnosis\_1\_Vocab,

d.Diagnosis\_1\_Description,

d.Diagnosis\_2\_Code,

d.Diagnosis\_2\_Vocab,

d.Diagnosis\_2\_Description,

d.Diagnosis\_3\_Code,

d.Diagnosis\_3\_Vocab,

d.Diagnosis\_3\_Description,

d.Diagnosis\_4\_Code,

d.Diagnosis\_4\_Vocab,

d.Diagnosis\_4\_Description,

d.Diagnosis\_5\_Code,

d.Diagnosis\_5\_Vocab,

d.Diagnosis\_5\_Description,

d.Diagnosis\_6\_Code,

d.Diagnosis\_6\_Vocab,

d.Diagnosis\_6\_Description,

d.Diagnosis\_7\_Code,

d.Diagnosis\_7\_Vocab,

d.Diagnosis\_7\_Description,

d.Diagnosis\_8\_Code,

d.Diagnosis\_8\_Vocab,

d.Diagnosis\_8\_Description,

d.Diagnosis\_9\_Code,

d.Diagnosis\_9\_Vocab,

d.Diagnosis\_9\_Description,

d.Diagnosis\_10\_Code,

d.Diagnosis\_10\_Vocab,

d.Diagnosis\_10\_Description

FROM #DIAGNOSIS\_TABLE1 d

GROUP BY

d.DiagnosisEventKey,

d.DiagnosisKey,

d.Diagnosis\_Name,

d.PatientEpicId\_SH,

d.EncounterId\_SH,

d.Encounter\_StartVisit,

d.Encounter\_EndVisit,

d.DiagnosisStartDate,

d.DiagnosisEndDate,

d.IsPrimary,

d.DiagnosisStatus,

d.Diagnosis\_1\_Code,

d.Diagnosis\_1\_Vocab,

d.Diagnosis\_1\_Description,

d.Diagnosis\_2\_Code,

d.Diagnosis\_2\_Vocab,

d.Diagnosis\_2\_Description,

d.Diagnosis\_3\_Code,

d.Diagnosis\_3\_Vocab,

d.Diagnosis\_3\_Description,

d.Diagnosis\_4\_Code,

d.Diagnosis\_4\_Vocab,

d.Diagnosis\_4\_Description,

d.Diagnosis\_5\_Code,

d.Diagnosis\_5\_Vocab,

d.Diagnosis\_5\_Description,

d.Diagnosis\_6\_Code,

d.Diagnosis\_6\_Vocab,

d.Diagnosis\_6\_Description,

d.Diagnosis\_7\_Code,

d.Diagnosis\_7\_Vocab,

d.Diagnosis\_7\_Description,

d.Diagnosis\_8\_Code,

d.Diagnosis\_8\_Vocab,

d.Diagnosis\_8\_Description,

d.Diagnosis\_9\_Code,

d.Diagnosis\_9\_Vocab,

d.Diagnosis\_9\_Description,

d.Diagnosis\_10\_Code,

d.Diagnosis\_10\_Vocab,

d.Diagnosis\_10\_Description

Medication Table

The Medication table is limited by the following statements in the initial WHERE clause. It is also limited to patients and encounters from the Patient and Encounters tables.

* The PatientKey in the MedicationEventFact table must be greater than 0. This prevents pulling in medication events with patients that may have been deleted from a specific table.
* The MedicationEvent must not have been deleted from the MedicationEventFact table.

##ENCOUNTER\_TABLE is a temporary table generated from the TRD\_Encounter\_Table-Current file. It exists in the global environment and can be run across files in a single session.

**SQL Query:**

Use Caboodle;

-- uses ##ENCOUNTER\_TABLE from global session

----------- MEDICATION BUILD START ------------------------------------------------

DROP TABLE IF EXISTS #MEDICATION\_TABLE;

SELECT

mef.MedicationEventKey,

ef.EncounterId\_SH,

ef.PatientEpicId\_SH,

mef.MedicationOrderKey as 'MedOrderNumber',

mef.MedicationKey as 'MedCode',

md.Name as 'MedName',

md.GenericName as 'MedGenericName',

md.SimpleGenericName as 'MedSimpleGenericName',

md.TherapeuticClass as 'MedTherapeuticClass',

md.PharmaceuticalClass as 'MedPharmaceuticalClass',

md.PharmaceuticalSubclass as 'MedPharmaceuticalSubclass',

md.DeaClass as 'MedDeaClass',

md.Strength as 'MedStrength',

md.Form as 'MedForm',

md.Route as 'MedRoute',

mef.DoseUnit as 'MedDoseUnit',

md.Controlled as 'ControlledMed\_YN',

md.Opioid as 'OpioidMed\_YN',

mef.StartInstant as 'MedStartInstant',

mef.EndInstant as 'MedEndInstant',

mef.AdministrationInstant as 'MedAdministrationInstant',

mof.OrderedInstant as 'MedOrderedInstant',

mef.Mode as 'MedPatientMode',

mef.Type as 'MedOrderType',

mef.Frequency as 'MedFrequency',

mef.RefillsWritten,

mef.DaysSupply,

mof.DiscontinueReason,

dd.DateValue as 'DiscontinuedDate',

ef.GroupType

INTO #MEDICATION\_TABLE

FROM

MedicationEventFact mef

INNER JOIN ##ENCOUNTER\_TABLE ef on mef.EncounterKey=ef.EncounterKey --Only patients/encounters filtered in previous query

LEFT OUTER JOIN MedicationDim md on mef.MedicationKey = md.MedicationKey

LEFT OUTER JOIN MedicationOrderFact mof on mef.MedicationOrderKey = mof.MedicationOrderKey

LEFT OUTER JOIN DateDim dd on mof.DiscontinuedDateKey = dd.DateKey

WHERE

mef.PatientKey > 0

AND mef.\_IsDeleted = 0

**Data Cleaning:**

This is the requested data cleaning by LIBR. The code replaces specific values with NULL, or ‘NULL,’ depending on if the column allows for NULL values.

DECLARE @sql NVARCHAR(MAX) = '';

SELECT @sql = @sql +

'UPDATE #MEDICATION\_TABLE SET [' + name + '] = ''NULL''

WHERE [' + name + '] = ''''

OR [' + name + '] LIKE ''\*Unspecified''

OR [' + name + '] LIKE ''\*Unknown''

OR [' + name + '] LIKE ''\*Not Applicable'';' + CHAR(10)

FROM tempdb.sys.columns

WHERE object\_id = OBJECT\_ID('tempdb..#MEDICATION\_TABLE')

AND name IN ('MedDoseUnit', 'MedFrequency'); -- Only include these two columns

EXEC sp\_executesql @sql;

SELECT @sql = @sql +

'UPDATE #MEDICATION\_TABLE SET [' + name + '] = NULL

WHERE [' + name + '] = ''''

OR [' + name + '] LIKE ''\*Unspecified''

OR [' + name + '] LIKE ''\*Unknown''

OR [' + name + '] LIKE ''\*Not Applicable'';' + CHAR(10)

FROM tempdb.sys.columns

WHERE object\_id = OBJECT\_ID('tempdb..#MEDICATION\_TABLE') -- Get columns for the temp table

AND system\_type\_id IN (167, 175, 231, 239); -- Filter for text-based columns (varchar, char, nvarchar, nchar)

EXEC sp\_executesql @sql;

**Data Export:**

Run a simple select query, then save the results to a .csv file with the naming conventions specified at the beginning of this document.

SELECT \* FROM #MEDICATION\_TABLE

Procedure Table

The Procedure table is limited by the following statements in the initial WHERE clause. It is also limited to patients and encounters from the Patient and Encounters tables.

* The ProcedureCodeSet is limited to CPT codes.
* Procedure ‘type’ is limited to General Procedures.
* The EncounterKey in the ProcedureEventFact table must be greater than 0, preventing bringing in procedures that may be from a deleted encounter.

##ENCOUNTER\_TABLE is a temporary table generated from the TRD\_Encounter\_Table-Current file. It exists in the global environment and can be run across files in a single session.

**SQL Query:**

Use Caboodle;

-- uses ##ENCOUNTER\_TABLE from global session

----------- PROCEDURE BUILD START ------------------------------------------------

DROP TABLE IF EXISTS #PROCEDURE\_TABLE;

SELECT

pef.ProcedureEventKey,

pef.ProcedureKey,

pef.EncounterKey,

ef.PatientEpicId,

ef.EncounterEpicCSN,

ef.PatientEpicId\_SH,

ef.EncounterId\_SH,

pd.Category as 'CPT\_Procedure\_Category',

pef.ProcedureCode as 'CPT\_Procedure\_Code',

pd.Name as 'CPT\_Procedure\_Description',

pef.ProcedureStartInstant,

pef.ProcedureEndInstant,

ef.GroupType

INTO #PROCEDURE\_TABLE

FROM ProcedureEventFact pef

INNER JOIN ##ENCOUNTER\_TABLE ef on pef.EncounterKey=ef.EncounterKey --Only patients/encounters filtered in previous query

INNER JOIN ProcedureDim pd on pef.ProcedureKey = pd.ProcedureKey

WHERE

pef.ProcedureCodeSet = 'CPT(R)'

AND

pef.Type = 'General Procedure'

AND pef.EncounterKey > 0 -- exclude negative keys

ORDER BY

pef.EncounterKey

**Data Cleaning:**

This is the requested data cleaning by LIBR. The code replaces specific values with NULL, or ‘NULL,’ depending on if the column allows for NULL values.

DECLARE @sql NVARCHAR(MAX) = '';

SELECT @sql = @sql +

'UPDATE #PROCEDURE\_TABLE SET [' + name + '] = ''NULL''

WHERE [' + name + '] = ''''

OR [' + name + '] LIKE ''\*Unspecified''

OR [' + name + '] LIKE ''\*Unknown''

OR [' + name + '] LIKE ''\*Not Applicable'';' + CHAR(10)

FROM tempdb.sys.columns

WHERE object\_id = OBJECT\_ID('tempdb..#PROCEDURE\_TABLE')

AND name IN ('CPT\_Code'); -- Only include this column, which doesn't allow nulls

EXEC sp\_executesql @sql;

SELECT @sql = @sql +

'UPDATE #PROCEDURE\_TABLE SET [' + name + '] = NULL

WHERE [' + name + '] = ''''

OR [' + name + '] LIKE ''\*Unspecified''

OR [' + name + '] LIKE ''\*Unknown''

OR [' + name + '] LIKE ''\*Not Applicable'';' + CHAR(10)

FROM tempdb.sys.columns

WHERE object\_id = OBJECT\_ID('tempdb..#PROCEDURE\_TABLE') -- Get columns for the temp table

AND system\_type\_id IN (167, 175, 231, 239); -- Filter for text-based columns (varchar, char, nvarchar, nchar)

EXEC sp\_executesql @sql;

**Data Export:**

Run a select query, removing patient identifying information, then save the results to a .csv file with the naming conventions specified at the beginning of this document.

SELECT

ProcedureEventKey,

PatientEpicId\_SH,

EncounterId\_SH,

CPT\_Procedure\_Category,

CPT\_Procedure\_Code,

CPT\_Procedure\_Description,

ProcedureStartInstant,

ProcedureEndInstant,

GroupType

FROM #PROCEDURE\_TABLE

GROUP BY

ProcedureEventKey,

PatientEpicId\_SH,

EncounterId\_SH,

CPT\_Procedure\_Category,

CPT\_Procedure\_Code,

CPT\_Procedure\_Description,

ProcedureStartInstant,

ProcedureEndInstant,

GroupType

RXNorm Table

The RXNorm table is limited by the following statements in the initial WHERE clause.

* RXNORM\_HISTORIC\_YN = NULL

**SQL Query:**

SELECT

RC.MEDICATION\_ID as 'MedCode',

RC.LINE,

RC.RXNORM\_CODE,

RCCL.NAME AS 'RXNORM\_LEVEL',

RCTT.NAME AS 'RXNORM\_TYPE'

FROM

RXNORM\_CODES rc

LEFT OUTER JOIN ZC\_RXNORM\_CODE\_LEVEL rccl ON rc.RXNORM\_CODE\_LEVEL\_C=rccl.RXNORM\_CODE\_LEVEL\_C

LEFT OUTER JOIN ZC\_RXNORM\_TERM\_TYPE rctt ON rc.RXNORM\_TERM\_TYPE\_C = rctt.RXNORM\_TERM\_TYPE\_C

WHERE

rc.RXNORM\_HISTORIC\_YN IS NULL

GROUP BY

RC.MEDICATION\_ID,

RC.LINE,

RC.RXNORM\_CODE,

RCCL.NAME,

RCTT.NAME

**Data Export:**

Save the results to a .csv file with the naming conventions specified at the beginning of this document. There is no patient identifying information in the dataset.

Frequently Asked Questions

* Can I compare the data in the flat files to the COSMOS data for validation?
  + No. The COSMOS dataset is more expansive than the Caboodle dataset – if you pull ‘My Organization’ in COSMOS, it will include all service areas, while the Caboodle dataset is only the Saint Francis service area (10). There are also flags and filters available in the Caboodle dataset that were used to limit the data further that are not available in COSMOS.
* Why are there more patients flagged in the Patient Table as having depression than there are in the Diagnosis Table?
  + If a patient was not diagnosed with depression at a Saint Francis location, they may come into a new department (emergency, primary care, etc) and list a historical diagnosis. The Diagnosis Table is associated with encounters – if the diagnosis is not associated with an encounter, it will not show up in the Diagnosis Table.