# Rush University Graph Database Loading CSV File Format Descriptions

## Entity Type Nodes:

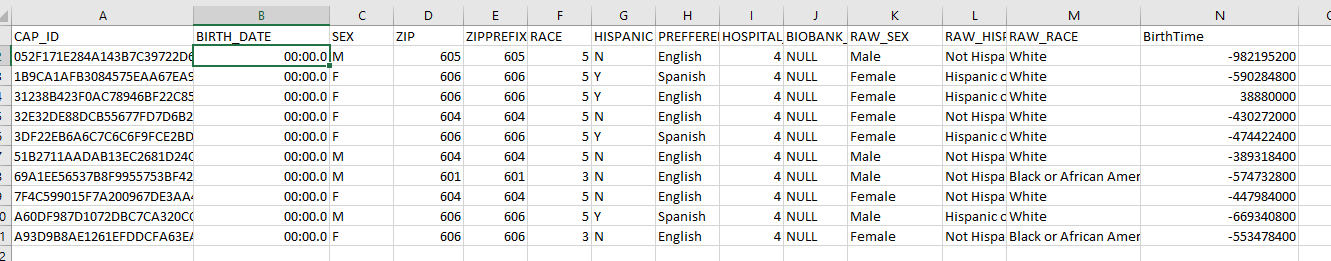
Patients

CSV File(s): UHEP10PTDEMOGRAPHICS.CSV

CQL File(s): loadPatients.cql

Cypher:

create (:Patient {patientID:line.CAP\_ID,dob:line.BIRTH\_DATE,sex:line.SEX,race:line.RAW\_RACE});



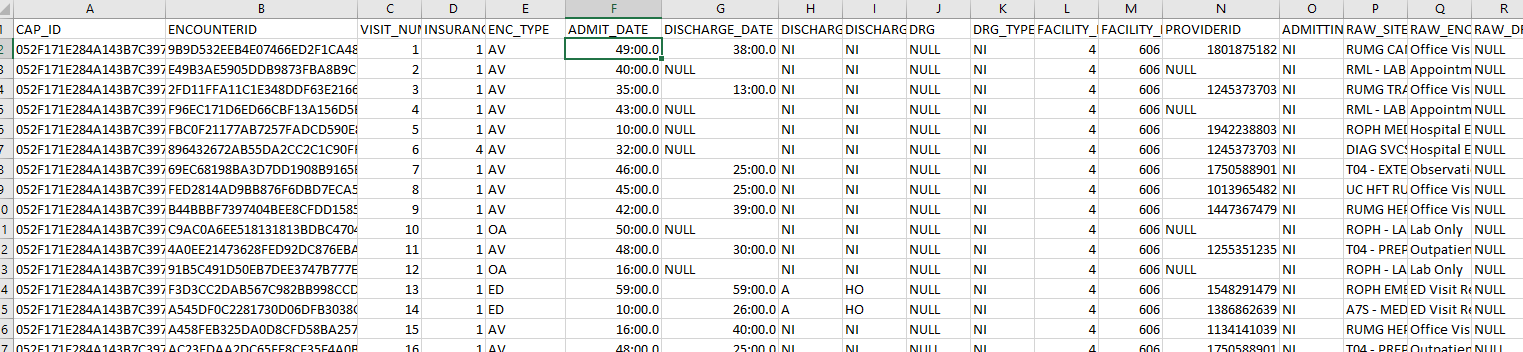
Encounters

CSV File(s): UHEP10PTENCOUNTERS.CSV

CQL File(s): loadEncounters.cql

Cypher:

create (:Encounter {encounterID:line.ENCOUNTERID, type:line.ENC\_TYPE,site:line.RAW\_SITE});



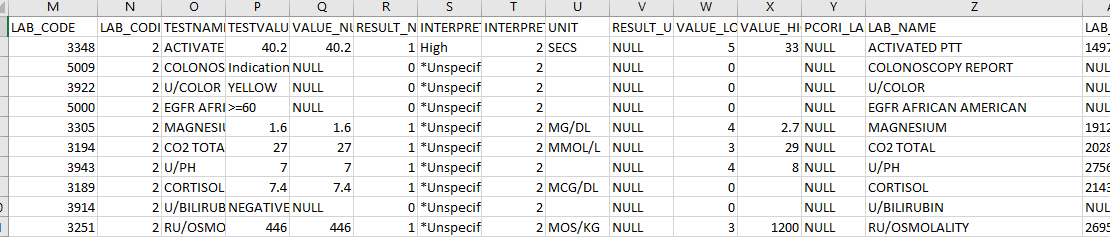
Labs:

CSV File(s): UHEP10PTLABS.CSV

CQL File(s): loadLabs.cql

Cypher:

merge (:Lab {code:line.LAB\_CODE, labName:line.LAB\_NAME});



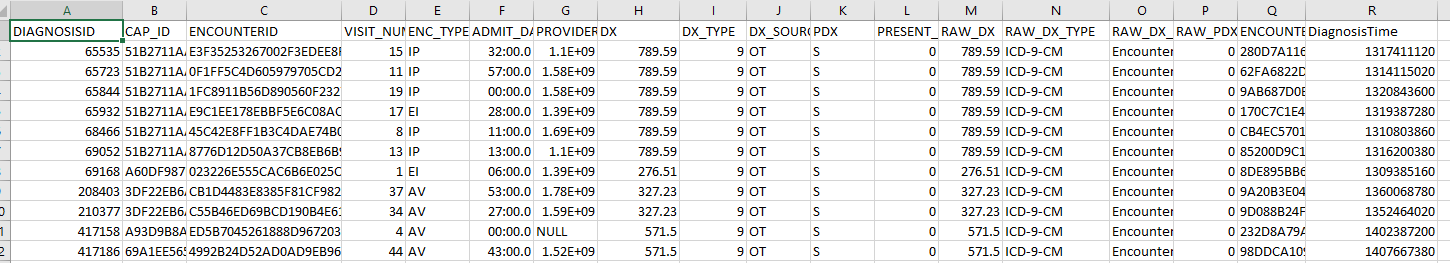
Diagnoses (DX Codes):

CSV File(s): UHEP10PTDIAGNOSES.CSV

CQL File(s): loadDX.cql

Cypher:

merge (:DX {code:line.DX, type:line.RAW\_DX\_TYPE});



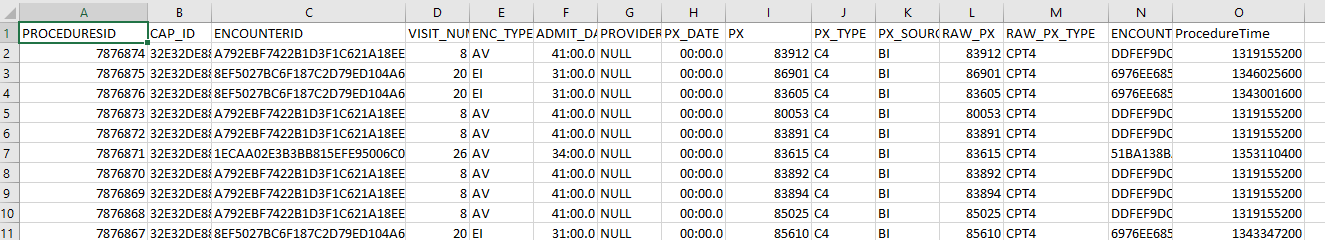
Procedures (PX Codes):

CSV File(s): UHEP10PTPROCEDURES.CSV

CQL File(s): loadPX.cql

Cypher:

merge (:PX {code:line.PX, type:line.RAW\_PX\_TYPE});



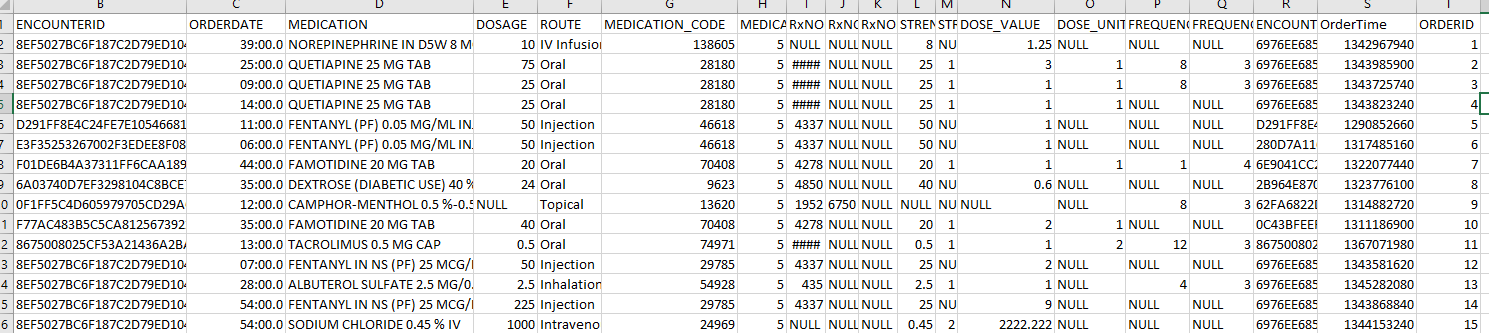
Medications:

CSV File(s): UHEP10PTMEDICATIONS.CSV

CQL File(s): loadRX.cql

Cypher:

merge (:RX {code:line.MEDICATION\_CODE, medication:line.MEDICATION});



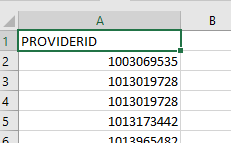
Providers

CSV File(s): UHE10PTPROVIDERS.CSV

CQL File(s): loadProviders.cql

Cypher:

merge (:Provider {ID:line.PROVIDERID});



To create the providers csv file, simply select all distinct providerid values from the source database.

## Instance Type Nodes:

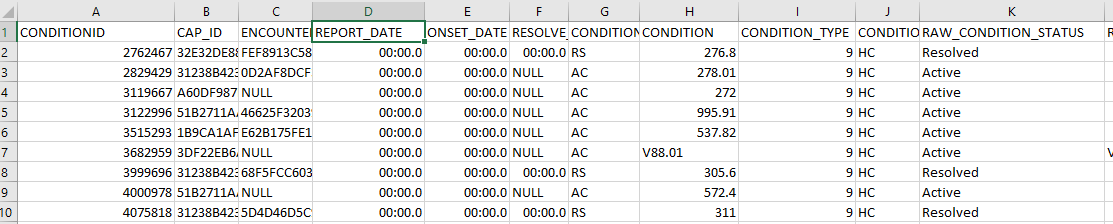
Conditions

CSV File(s): UHE10PTCONDITIONS.CSV

CQL File(s): loadConditions.cql, loadConditionDXRels.cql, loadPatientConditionRels.cql

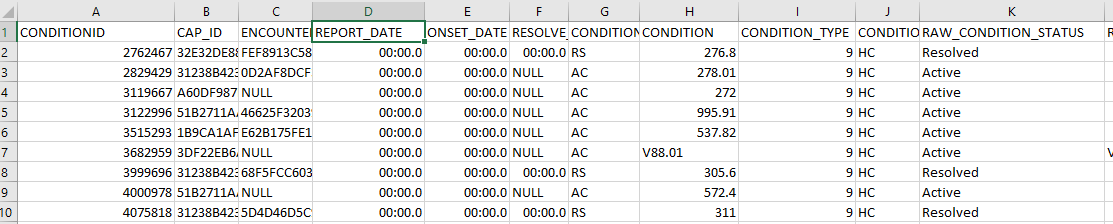
Cypher:

create (:Condition {ID:line.CONDITIONID, status:line.RAW\_CONDITION\_STATUS});



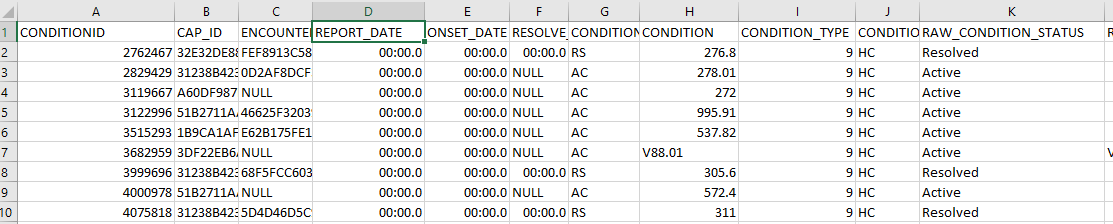
match (c:Condition {ID:line.CONDITIONID}) match (dx:DX {code:line.CONDITION})

create (dx)<-[:INDICATED]-(c);



match (c:Condition {ID:line.CONDITIONID}) match (p:Patient {patientID:line.CAP\_ID})

create (c)<-[:SUFFERED]-(p);



Lab Requests

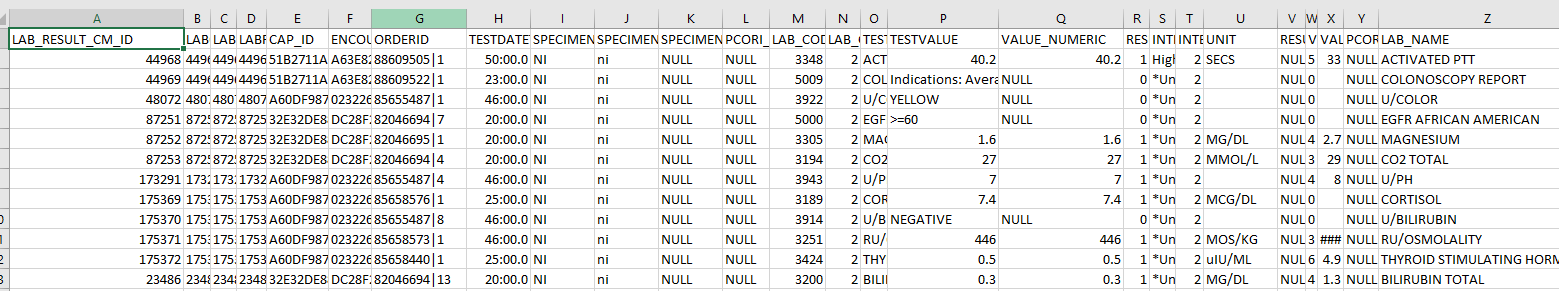
CSV File(s): UHE10PTLABS.CSV

CQL File(s): loadLabRequests.csv, loadLabRequestLabRels.cql

Cypher:

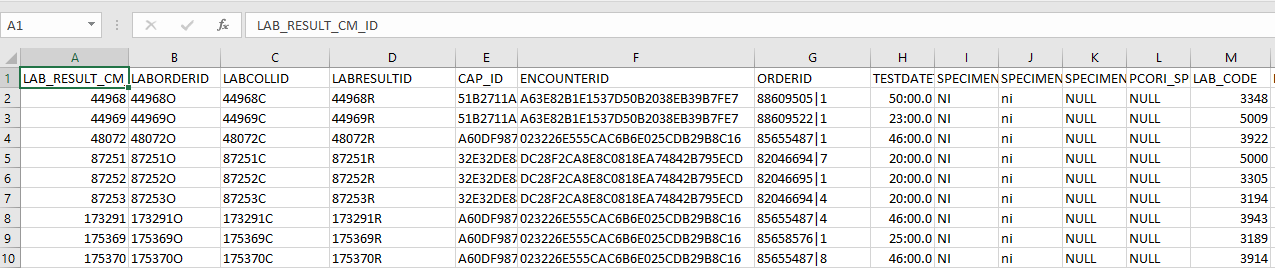
create (:LabRequest{ID:line.LAB\_RESULT\_CM\_ID, labRequest:line.ORDERID,labName:line.LAB\_NAME,

labResult:line.TESTVALUE, labNumeric:line.VALUE\_NUMERIC, labResultUnit:line.UNIT });



match (labRequest:LabRequest {ID:line.LAB\_RESULT\_CM\_ID}) match (lab:Lab {code:line.LAB\_CODE})

create (labRequest)-[:REQUESTED]->(lab);



## Event Type Nodes

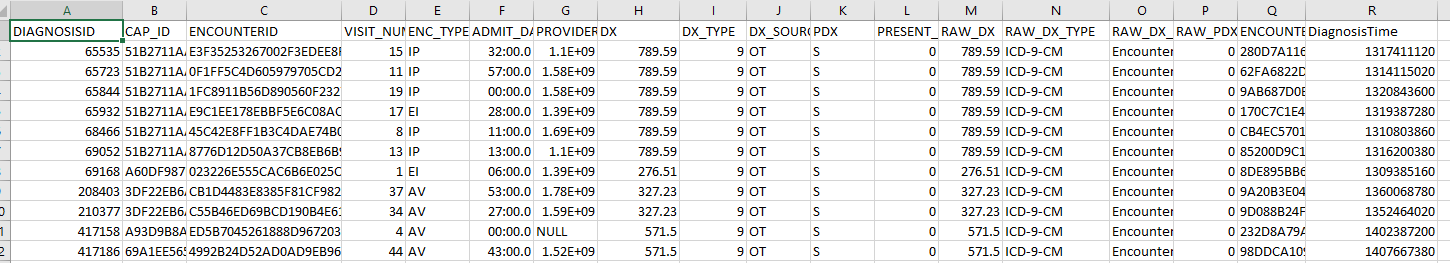
Diagnoses

CSV File(s): UHE10PTDIAGNOSES.CSV

CQL File(s): loadDiagnoses.cql, loadDiagnosisDXRels.cql, loadEncounterDiagnosisRels.cql, loadProviderDiagnosisRels.cql

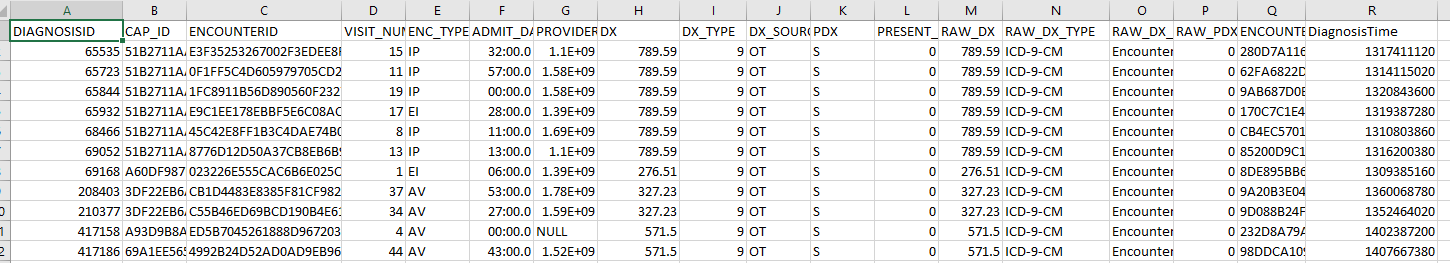
Cypher:

create (:Diagnosis {ID:line.DIAGNOSISID, time:toInt(line.DiagnosisTime)});



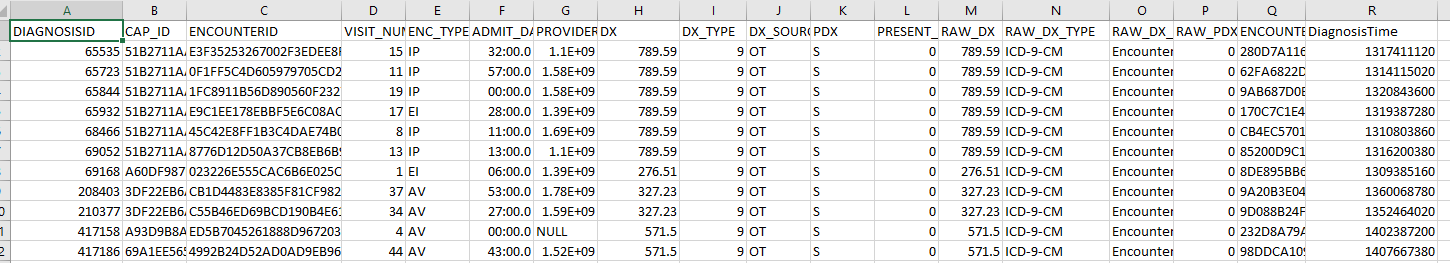
match (diagnosis:Diagnosis {ID:line.DIAGNOSISID}) match (dx:DX {code:line.DX})

create (diagnosis)-[:INDICATED]->(dx);



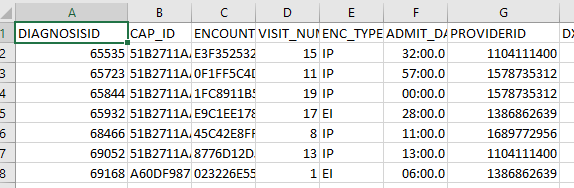
match (diagnosis:Diagnosis {ID:line.DIAGNOSISID}) match (encounter:Encounter {encounterID:line.ENCOUNTERID})

create (diagnosis)<-[:ENCOMPASSED]-(encounter);



match (diagnosis:Diagnosis {ID:line.DIAGNOSISID}) match (provider:Provider {ID:line.PROVIDERID})

create (diagnosis)<-[:DIAGNOSED]-(provider);



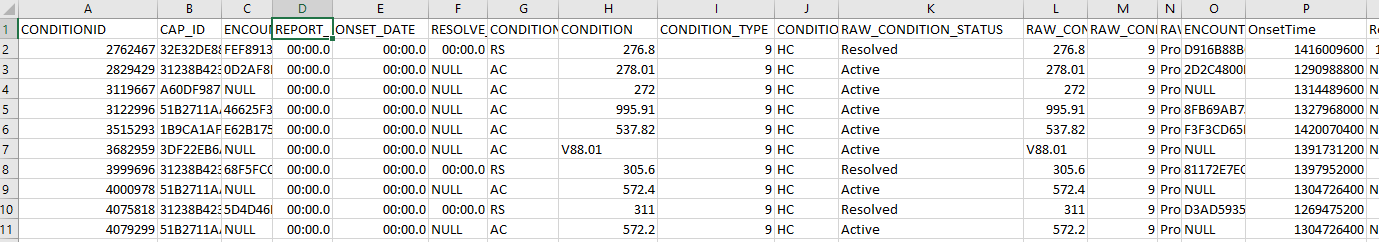
Condition Onsets

CSV File(s): UNE10PT

CQL File(s): loadConditionOnsets.cql

Cypher:

create (:ConditionOnset {ID:line.CONDITIONID + "O", onsetDate:line.ONSET\_DATE, time:toInt(line.OnsetTime)});

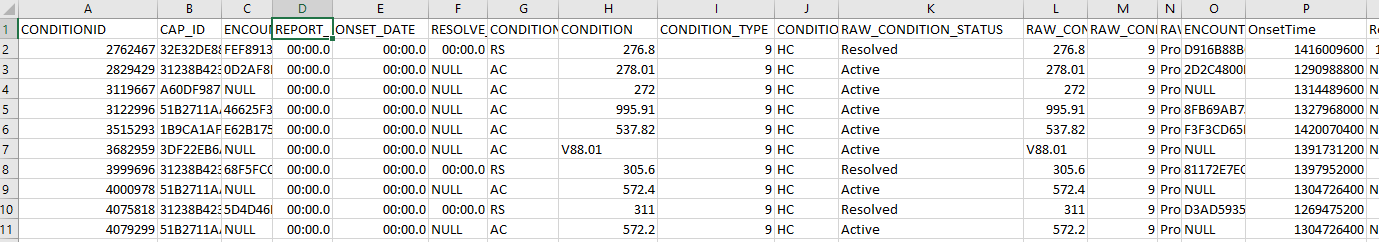


When creating the condition onset node, I ensure unique ID value for the node by appending an “O” to the conditiondid value for that onsettime row. So for every condition which has a condition ID value, if it has an onset (all do) it there will be a corresponding condition onset with an id equal to the associated condition with an “O” appended to it.

MATCH (co:ConditionOnset {ID:line.CONDITIONID + "O"})

MATCH (c:Condition {ID:line.CONDITIONID})

CREATE (c)-[:HAD]->(co);



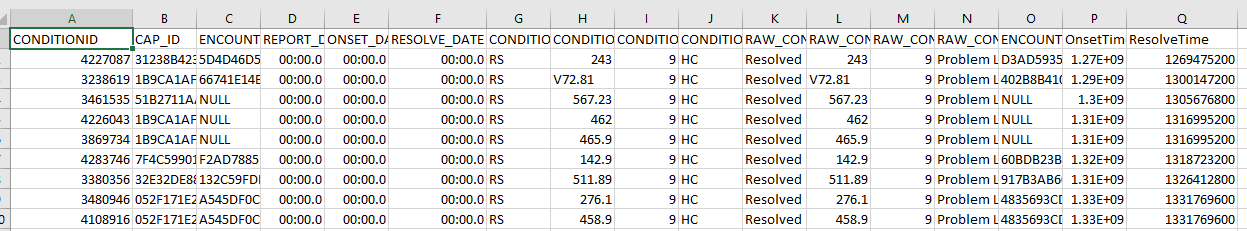
Condition Resolutions

CSV File(s): UNE10PTCONDITIONRES.CSV

CQL File(s): loadConditionResolutions.cql, loadConditionResolutionRels.cql

Cypher:

create (:ConditionResolution {ID:line.CONDITIONID + "R", onsetDate:line.RESOLVE\_DATE, time:toInt(line.ResolveTime)});



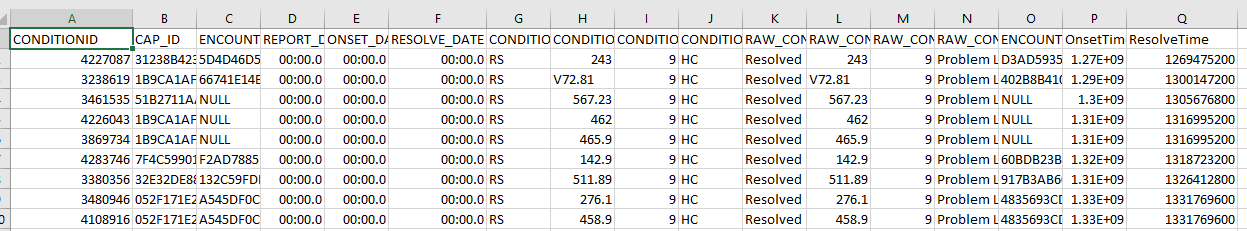
Because not all conditions have resolutions, we need to create a condition resolution csv file that selects condition rows from the source database where ResolveTime is not null.

When creating the condition resolution node, I ensure unique ID value for the node by appending an “R” to the conditiondid value for that resolvetime row. So, for every condition which has a condition ID value, if it has a resolution there will be a corresponding condition resolution node with an id equal to the associated condition but with an “R” appended to it.

MATCH (cr:ConditionResolution {ID:line.CONDITIONID + "R"})

MATCH (c:Condition {ID:line.CONDITIONID})

CREATE (c)-[:HAD]->(cr);



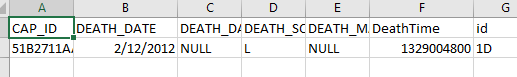
Deaths

CSV File(s): UHE10PTDEATH.CSV

CQL File(s): loadDeaths.cql, loadPatientDeathRels.cql

Cypher:

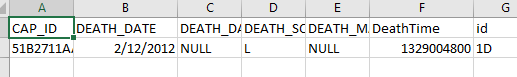
create (:Death {ID:line.id,deathDate:line.DEATH\_DATE, time:toInt(line.DeathTime)});



For this load procedure, I added a column to the death.csv file called id. This was a surrogate key that was the rownumber with a “D” appended to it.

match (d:Death {ID:line.id}) match (p:Patient {patientID:line.CAP\_ID})

create (p)-[:DIED]->(d);



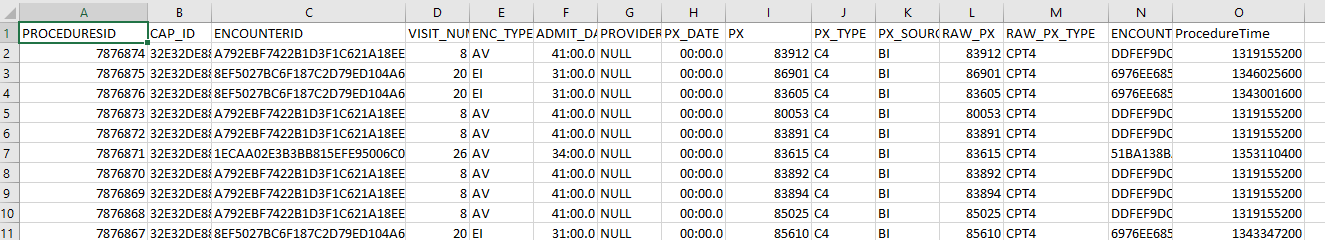
Procedures

CSV File(s): UHE10PTPROCEDURES.CSV, UHE10PTPROVPROCS.CSV

CQL File(s): loadProcedures.cql, loadProviderProcedureRels.cql, loadProcedurePXRels.cql, loadProcedureEncounterRels.cql

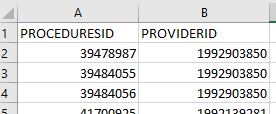
Cypher:

create (:Procedure{ID:line.PROCEDURESID, time:toInt(line.ProcedureTime) });



match (provider:Provider {ID:line.PROVIDERID}) match (procedure:Procedure {ID:line.PROCEDURESID})

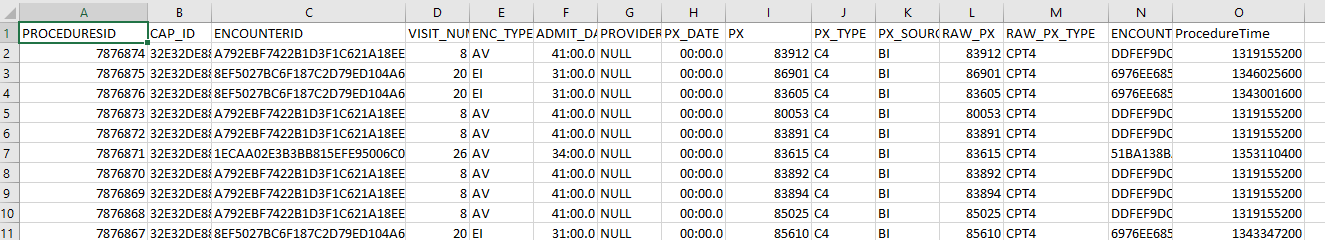
create (provider)-[:PROVIDED]->(procedure);



Because not all procedures are associated with a provider, I had to create a new csv file that contain only those procedures that had a provider. To create this csv file, query all procedures where providerid is not null.

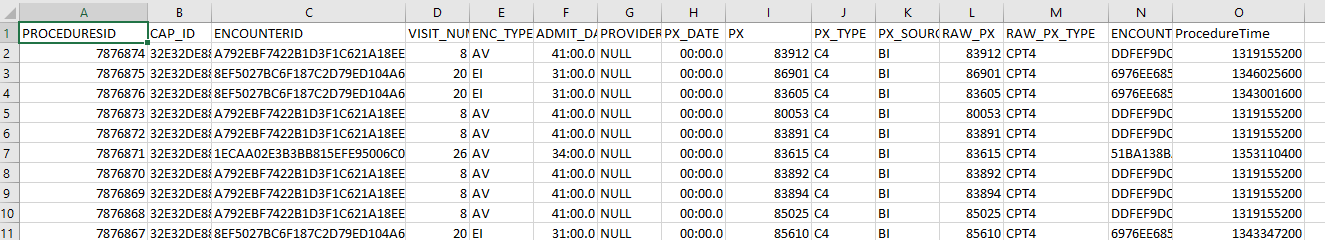
match (procedure:Procedure {ID:line.PROCEDURESID}) match (px:PX {code:line.PX})

create (procedure)-[:PERFORMED]->(px);



match (procedure:Procedure {ID:line.PROCEDURESID}) match (encounter:Encounter {encounterID:line.ENCOUNTERID})

create (procedure)<-[:ENCOMPASSED]-(encounter);



RXOrders

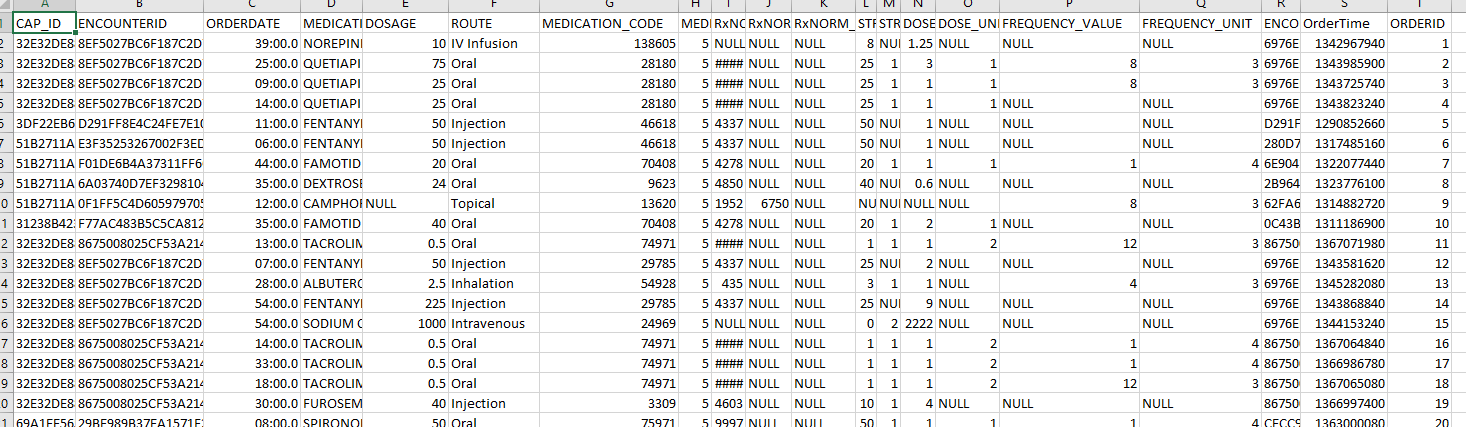
CSV File(s): UHE10PTMEDICATIONS

CQL File(s): loadRXOrders.cql, loadRXOrderRXRels.cql, loadEncounterRXOrderRels.cql

Cypher:

create (:RXOrder {ID:line.ORDERID, time:toInt(line.OrderTime), route:line.ROUTE, frequency:line.FREQUENCY\_VALUE,

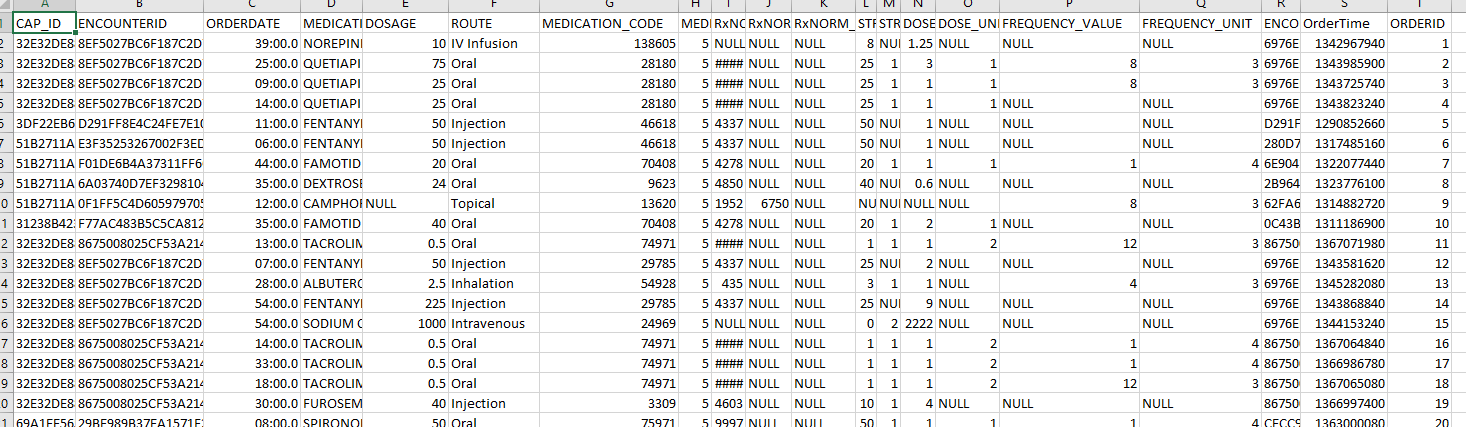
frequencyUnit:line.FREQUENCY\_UNIT });



I added a column to this csv file to provide a surrogate key for every medication order. I just used the rownumber as a primary key value, although we might consider appending a “RX” to the id to ensure overall uniqueness.

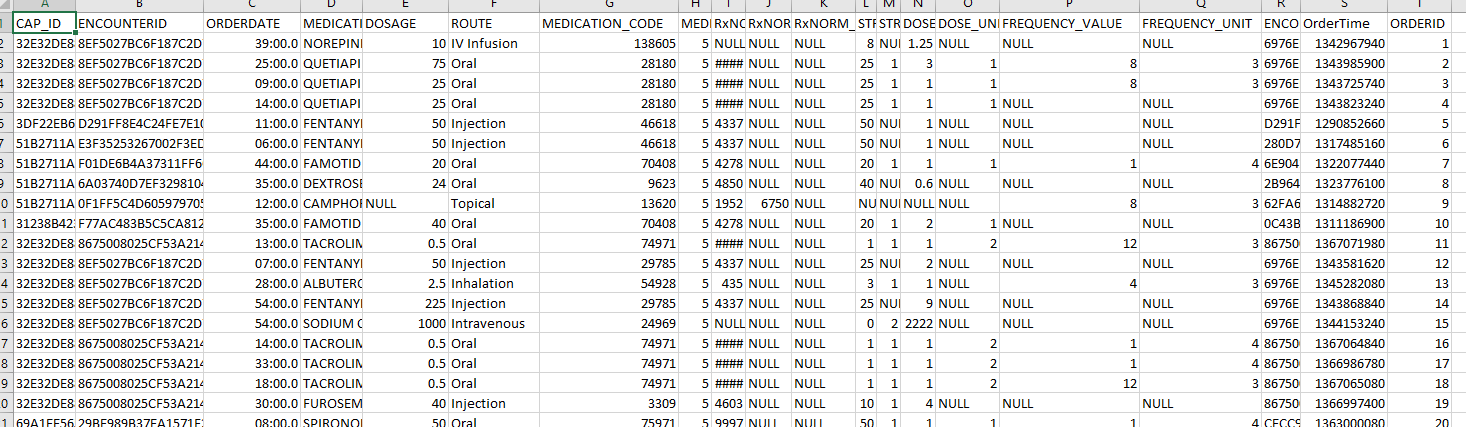
match (rxOrder:RXOrder {ID:line.ORDERID}) match (rx:RX {code:line.MEDICATION\_CODE})

create (rxOrder)-[:ORDERED]->(rx);



match (rxOrder:RXOrder {ID:line.ORDERID}) match (encounter:Encounter {encounterID:line.ENCOUNTERID})

create (rxOrder)<-[:ENCOMPASSED]-(encounter);



Vital Observations

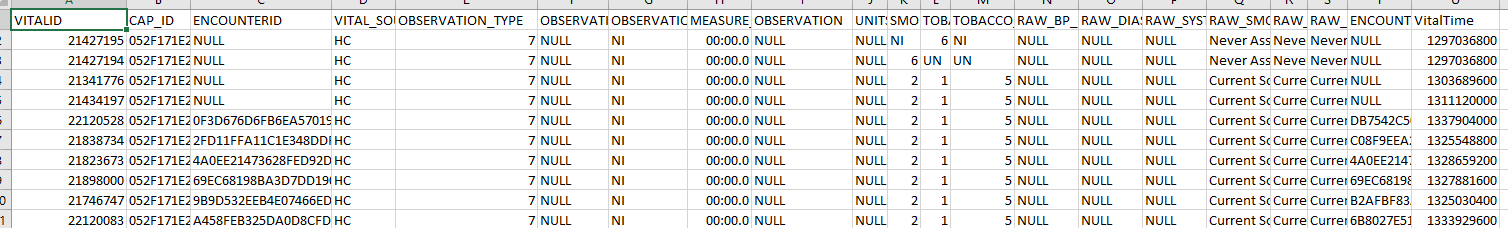
CSV File(s): UHE10PTVITALS.CSV

CQL File(s): loadVitals.cql, loadEncounterVitalRels.cql

Cypher:

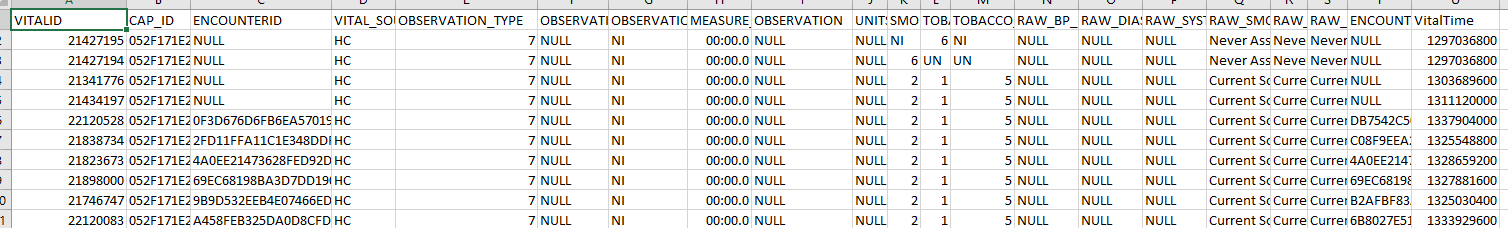
create (:VitalObservation {ID:line.VITALID, type:line.OBSERVATION\_TYPE, result:line.OBSERVATION,

time:toInt(line.VitalTime)});



match (vitalObs:VitalObservation {ID:line.VITALID}) match (encounter:Encounter {encounterID:line.ENCOUNTERID})

create (vitalObs)<-[:ENCOMPASSED]-(encounter);



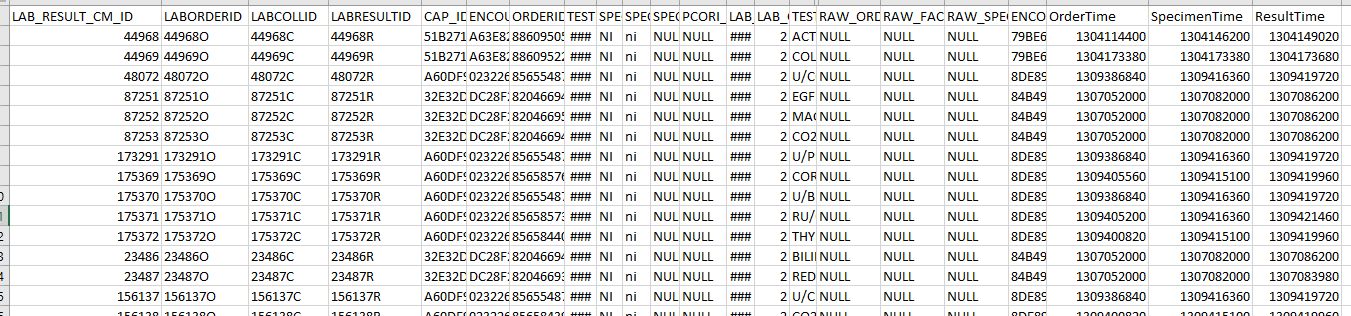
Lab Orders

CSV File(s): UHE10PTLABS

CQL File(s): loadLabOrders.cql, loadLabRequestOrderRels.cql, loadEncounterLabOrderRels.cql

Cypher:

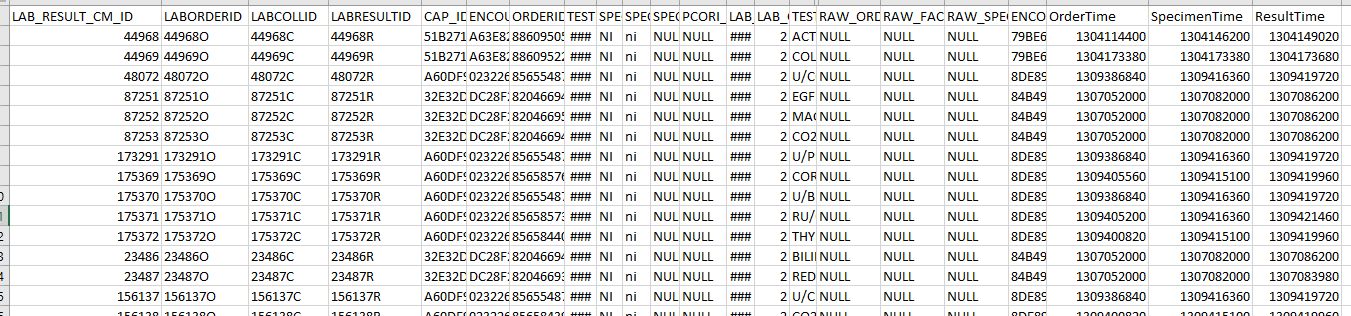
create (:LabOrder{ID:line.LABORDERID, time:toInt(line.OrderTime) });



I’ve added three columns to the labs csv file. I’ve added LABORDERID, LABCOLLID and LABRESULTID. Each of these columns are populated by selecting the LAB\_RESULT\_CM\_ID value and appending either an “O”, a “C” or an “R” (O for LABORDERID, C for LABCOLLID and R for LABRESULTID).

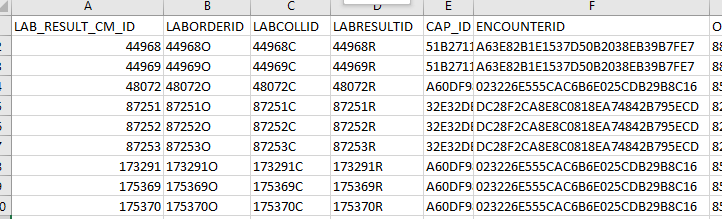
match (labRequest:LabRequest {ID:line.LAB\_RESULT\_CM\_ID}) match (labOrder:LabOrder {ID:line.LABORDERID})

create (labRequest)-[:HAD]->(labOrder);



match (labOrder:LabOrder {orderID:lineL.LABORDERID}) match (encounter:Encounter {encounterID:line.ENCOUNTERID})

create (labOrder)<-[:ENCOMPASSED]-(encounter);



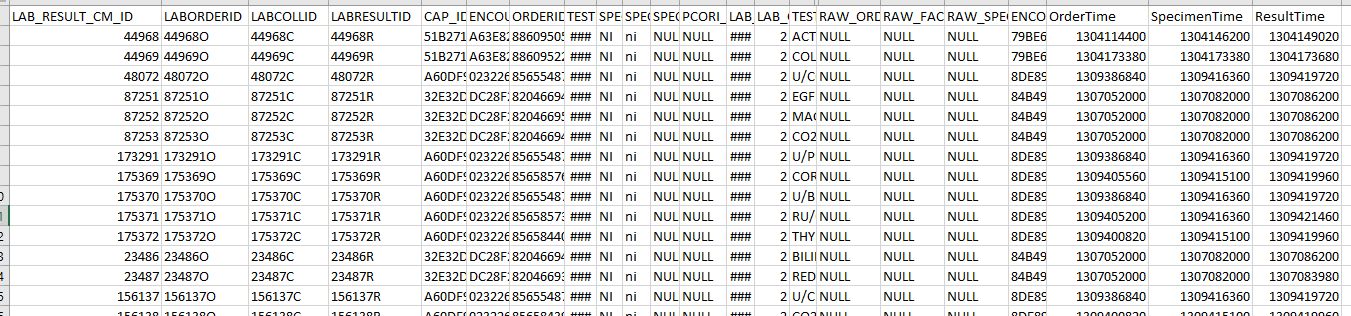
Lab Collections

CSV File(s): UHE10PTLABS

CQL File(s): loadLabCollections.cql, loadLabRequestCollectionRels.cql, loadEncounterLabColletionRels.cql

Cypher:

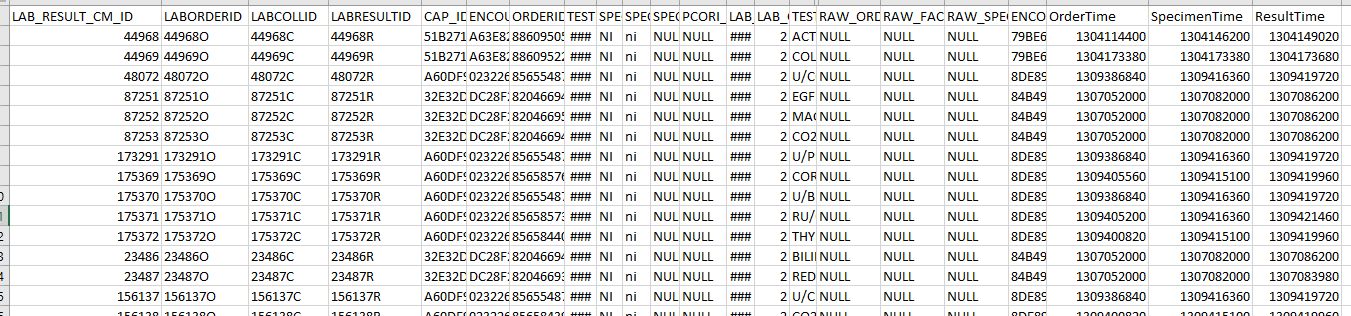
create (:LabCollection{ID:line.LABCOLLID, time:toInt(line.SpecimenTime) });



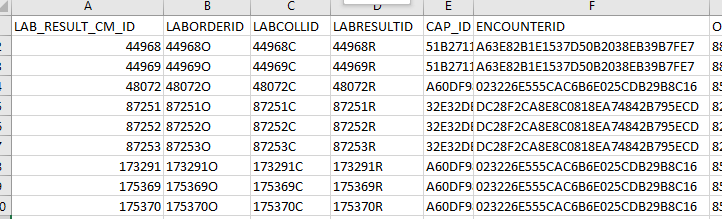
I’ve added three columns to the labs csv file. I’ve added LABORDERID, LABCOLLID and LABRESULTID. Each of these columns are populated by selecting the LAB\_RESULT\_CM\_ID value and appending either an “O”, a “C” or an “R” (O for LABORDERID, C for LABCOLLID and R for LABRESULTID).

match (labRequest:LabRequest {ID:line.LAB\_RESULT\_CM\_ID}) match (labCollection:LabCollection {ID:line.LABCOLLID})

create (labRequest)-[:HAD]->(labCollection);



match (labCollection:LabCollection {ID:line.LABCOLLID match (encounter:Encounter {encounterID:line.ENCOUNTERID})

create (labCollection)<-[:ENCOMPASSED]-(encounter);

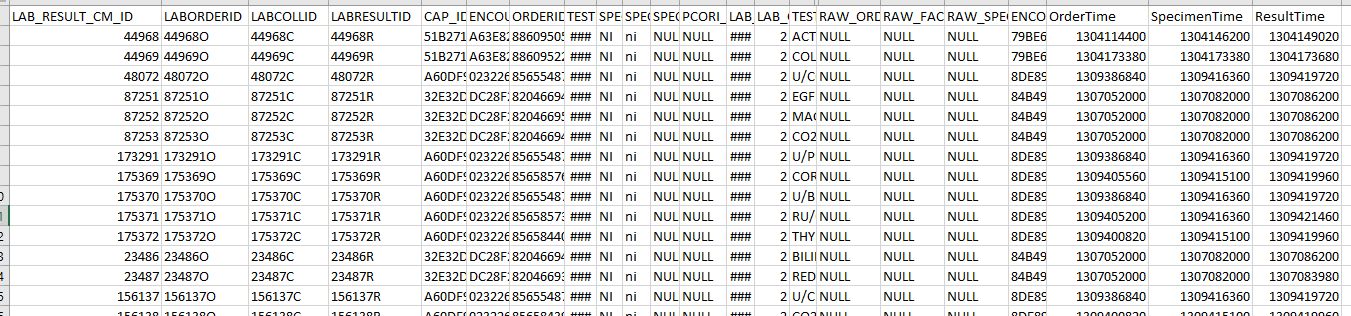
Lab Results

CSV File(s): UHE10PTLABS

CQL File(s): loadLabResults.cql, loadLabRequestResultRels.cql, loadEncounterLabResultRels.cql

Cypher:

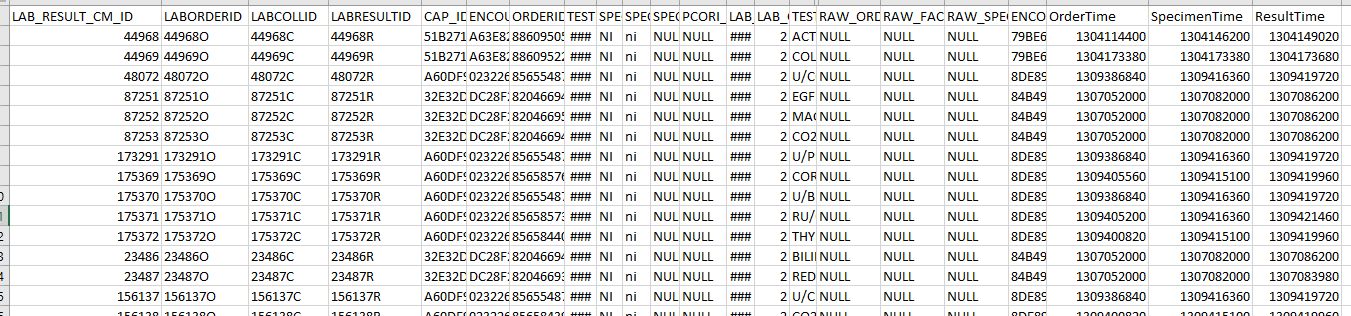
create (:LabResult{ID:line.LABRESULTID, time:toInt(line.ResultTime) });



I’ve added three columns to the labs csv file. I’ve added LABORDERID, LABCOLLID and LABRESULTID. Each of these columns are populated by selecting the LAB\_RESULT\_CM\_ID value and appending either an “O”, a “C” or an “R” (O for LABORDERID, C for LABCOLLID and R for LABRESULTID).

match (labRequest:LabRequest {ID:line.LAB\_RESULT\_CM\_ID}) match (labResult:LabResult {ID:line.LABRESULTID})

create (labRequest)-[:HAD]->(labResult);



match (labResult:LabResult {ID:line.LABRESULTID}) match (encounter:Encounter {encounterID:line.ENCOUNTERID})

create (labResult)<-[:ENCOMPASSED]-(encounter);

