Visualizing Data that does not exist … aka Readmissions Dashboard

written by DaltonRuer | July 29, 2016

Many who make requests seem to have a belief that Business Intelligence is magic. They loose their ability to listen to logic and reason and simply ask you to do the impossible.

[Tweet “Many who make requests seem to have a belief that Business Intelligence is magic.”]

Pulling data from 18 different sources, many of which that you don’t even have access to. Childs play like pulling a rabbit from a hat.

Turning bad into good and interpreting the meaning of the data. A little tougher kind of like making your stunning assistant float in midair.

Creating a readmissions dashboard. Hey we aren't Houdini.

That data doesn't even really exist. Oh sure it exists in the minds of the people who want you to produce it out of thin air, but I've yet to see a single Electronic Health Record that stored readmission data. They only store admission data, not **RE**-admission data.

Patient Name	Admission Date	Discharge Date
John Doe	1/1/2016	1/4/2016
John Doe	1/7/2016	1/10/2016
John Doe	1/30/2016	2/4/2016

Those who want dashboards for Readmissions look at data like the above and talk to you like you are insane because in their minds it is clear as day that John Doe was **re**admitted on 1/7, 3 days after their first visit, and was then **re**admitted again on 1/30, 20 days after his second visit.

You try to explain to them that there is nothing in any of those rows of data that says that. They have filled in the missing data in their minds but in reality it doesn't exist in the EHR. They respond with all you need to do is have the "report" do the same thing and compare the admission date to the discharge date for subsequent visits. You respond with "Let's say I could make SQL which is a row based tool magically compare rows, what should I do about the following which is more like the real data?"

Patient Name	Admission Date	Discharge Date	Patient Type
John Doe	1/1/2016	1/4/2016	Inpatient
John Doe	1/7/2016	1/10/2016	Outpatient
John Doe	1/30/2016	2/4/2016	Inpatient

They say "Oh that's easy, when you get to the visit on 1/30 just skip the visit from 1/7 because it's an outpatient row and we don't really care about those and compare the 1/30

admission to the 1/4 discharge." To which you respond "Well that's easy enough now I'll not only somehow make SQL which can't compare rows magically try to compare rows and if it is an outpatient row I'll tell SQL to skip it and compare it to something 2 rows above, or maybe 3 rows above or 10 rows above."

Just then you remember the reality is more complicated than that. In reality you aren't just comparing all inpatient visits (other than for fun) what you really care about are if the visits were for the same core diagnosis or not.

Enc ID	Patient Name	Admission Date	Discharge Date	Patient Type	Diagnosis
1	John Doe	1/1/2016	1/4/2016	Inpatient	COPD
2	John Doe	1/7/2016	1/10/2016	Outpatient	Stubbed toe
3	John Doe	1/30/2016	2/4/2016	Inpatient	Heart Failure
4	John Doe	2/6/2016	2/10/2016	Inpatient	COPD
5	John Doe	2/11/2016	2/16/2016	Inpatient	Heart Failure

You don't want to compare the 1/30 visit to the 1/4 discharge because the diagnosis aren't the same you only want to compare the 2/6 visit to the 1/4 discharge and you need to compare the 2/11 visit with the 2/4 discharge.

If you think this is like making a 747 disappear before a crowd of people on all sides, just wait it gets worse.

Not only does the EHR not include the "readmission" flags, it doesn't really tell you what core diagnosis the visit should count as. Instead what they really store is a table of 15-25 diagnosis codes

Enc ID	ICD9_1	ICD9_2	ICD9_3	ICD9_4	ICD9_….	ICD9_25
1	491.1	023.2	33.5	V16.9		37.52

Good thing for your company you used to be a medical coder so you actually understand what the mysterious ICD9 or ICD10 codes stand for. You know for instance that the 491.1 really means “Mucopurulent chronic bronchitis.” It would be nice if that correlated directly to saying “This patient visit is for COPD.” But since we are uncovering magic why not explain the whole trick. You see if the primary diagnosis code is any of the following:

491.1, 491.20, 491.21, 491.22, 491.8, 491.9, 492.0, 492.8, 493.20, 493.21, 493.22, 494.0, 494.1, 496

Then the visit may be the result of COPD but you also have to check all of the other diagnosis codes and ensure that none of them contain any of the following other diagnosis codes:

33.51, 33.52, 37.51, 37.52, 37.53, 37.54, 37.62, 37.63′, 33.50, 33.6, 50.51, 50.59, 52.80, 52.82, 55.69′,’196.0, 196.1, 196.2, 196.3, 196.5, 196.6, 196.8, 196.9, 197.0, 197.1, 197.2, 197.3, 197.4, 197.5, 197.6, 197.7, 197.8, 198.0, 198.1, 198.2, 198.3, 198.4, 198.5, 198.6, 198.7, 198.81, 198.82, 198.89, 203.02, 203.12, 203.82, 204.02, 204.12, 204.22,

204.82, 204.92, 205.02, 205.12, 205.22, 205.82, 205.92, 206.02, 206.12, 206.22, 206.82, 206.92, 207.02, 207.12, 207.22, 207.82, 208.02, 208.12, 208.22, 208.82, 208.92, 480.3, 480.8, 996.80, 996.81, 996.82, 996.83, 996.84, 996.85, 996.86, 996.87, 996.89, V42.0, V42.1, V42.4, V42.6, V42.7, V42.81, V42.82, V42.83, V42.84, V42.89, V42.9, V43.21, V46.11

If you have ever been asked to produce a Readmissions Dashboard you probably understand why I've correlated this to magic. Every time you think you know how to grab the rabbit by the ears to accomplish the trick, the rabbit changes into an elephant.

Fortunately your assistant isn't the traditional 6 foot blonde, your assistant is Qlik. I'm going to explain how to make the 747 disappear in three easy steps that any of you will be able to reproduce:

Step 1

The heavy lifting for this trick actually involves the ICD9/10 codes. If you combine the 15-25 diagnosis codes into 1 field, then you you can use it to more easily compare the values to determine what core diagnosis you need to assign to each encounter. Qlik helps you accomplish that with simple concatenation as you are loading your encounter diagnosis data:

```
ICD9_Diagnoses_1 & ', ' & ICD9_Diagnoses_2 & ', ' &
ICD9_Diagnoses_3 & ', ' & ICD9_Diagnoses_4 & ', ' &
ICD9_Diagnoses_5 & ', ' & ICD9_Diagnoses_6 & ', '
& ICD9_Diagnoses_7 & ', ' & ICD9_Diagnoses_8 & ', ' &
```

```
ICD9_Diagnoses_9 & ‘, ‘ & ICD9_Diagnoses_10 & ‘, ‘ &
ICD9_Diagnoses_11 &’, ‘ & ICD9_Diagnoses_12 & ‘, ‘
& ICD9_Diagnoses_13 & ‘, ‘ & ICD9_Diagnoses_14 & ‘, ‘ &
ICD9_Diagnoses_15 as [All Diagnosis]
```

Step 2

One of the really nifty tricks that Qlik can perform in data loading is a preceeding load. A preceeding load simply means you have the ability to write code to refer to fields that don’t exist yet and won’t exist until the code is actually run. The following code is abbreviated slightly so that it’s easier to follow logically but the entire set of code is attached to the post so that you can download it. The “Load *” right below Encounters tells Qlik to load all of the other from the second load statement first, then come back and do the code below. This way we can construct the [All Diagnosis] field and refer to it within this code. You could repeat all of the logic for concatenating all of the fields for all 5-10 of the core diagnosis you want to track, or you could load the encounters and simply do a subsequent join load but you don’t have to. The Preceeding load makes your life easy and works super fast.

```
Encounters:

This is the preceeding load
Load *,
// If the primary matches then it’s possibly COPD and if the
none of the other 14 are one of the values listed then it
definitely is COPD
IF ( Match([ICD9 Diagnoses 1] , ‘491.1’, ‘491.20’ …
‘493.21’, ‘493.22’, ‘494.0’, ‘494.1’, ‘496’) > 0
And WildMatch([All Diagnosis], ‘*33.51*’, ‘*33.52*’,
‘*37.51*’ … ‘*V43.21*’, ‘*V46.11*’) = 0, ‘COPD’,

// If we found COPD great, otherwise we need to check for
```

```
Sepsis
IF (Match ([ICD9 Diagnoses 1] , '003.1', '027.0', … '785.52'
) > 0
And WildMatch([All Diagnosis], '*33.50*', '*33.51*' …
'*V43.21*', '*205.32*') = 0, 'Sepsis',
'Nothing')) as [Core Diagnosis];


This is the regular load from the database or file
LOAD
MRN,
EncounterID,
……
[ICD9 Diagnoses 1],
[ICD9 Diagnoses 2] …..
```

Step 3

The final step, which many believe to be the hardest is actually the easiest to do within Qlik. In fact truth be told when I was a young whipper snapper starting out on my Qlik journey I tried to do everything in SQL because I knew it so well, and did minimal ETL within Qlik itself until I found about this Qlik ETL function. The function is simply called "Previous." It does exactly what it sounds like it … it allows you to look at the previous row of data. Seriously, while you are on row 2 you can check the value of a field on row 1. In practice it works just like this:

```
IF(MRN = Previous(MRN) …..
```

How cool is that? How do I use it for solving this readmissions magic trick? Just like this:

```
IF(MRN = Previous(MRN),'Yes', 'No') as [Inpatient
IsReadmission Flag],
```

If the MRN of the row I'm on now, is the same as the MRN of

the previous row, then yes this is a readmission, otherwise no this is not a readmission it is a new patients first admission. Actually that's the simplified version of my code.

My code actually thinks through how the results would need to be visualized. Besides an easy human language Yes/No flag someone is going to want to get a count of the readmissions right? Does the Qlik Dork want to have charts or expressions that would have to use IF statements to say if the flag = Yes, of course not. I want the ability to have field that is both human readable Yes/No, but also computer readable for counting 1/0. That's where the magic of the DUAL function comes into play. It gives me a single field that can be used for both needs.

```
IF(MRN = Previous(MRN),Dual('Yes', 1),Dual('No',0)) as
[Inpatient IsReadmission Flag],
```

Using the Dual data type allows me to provide the end user with a list box while also allowing me to provide very fast performing expressions:

```
Sum([Inpatient IsReadmission Flag])
```

How does the entire Readmissions load work? After loading the encounters, and allowing the preceeding load to qualify the encounters into core diagnosis types I simply do a self-join to the encounter table referring only to the inpatient records and ordering the data by the MRN and the Admission date and time.

```
Left Join (Encounters)
LOAD
EncounterID,
IF(MRN = Previous(MRN),Dual('Yes', 1),Dual('No',0)) as
[Inpatient IsReadmission Flag],
IF(MRN = Previous(MRN),Previous([Discharge Dt/Tm])) as
[Inpatient Previous Discharge Date],
IF(MRN = Previous(MRN),Previous(EncounterID)) as [Inpatient
```

```
Previous EncounterID],
IF(MRN = Previous(MRN),NUM(Interval([Admit Dt/Tm]-
Previous([Discharge Dt/Tm])),'#,##0.00´)) as [Inpatient
Readmission Difference],
IF(MRN = Previous(MRN),IF(Interval([Admit Dt/Tm]-
Previous([Discharge Dt/Tm])) <= 30.0, Dual('Yes', 1),
 Dual('No',0)), Dual('No',0)) as [Inpatient IsReadmission
within 30]
Resident Encounters
Where [Patient Type] = 'Inpatient'
Order by MRN, [Admit Dt/Tm];
```

If you are paying attention you'll notice that the above is simply our "for fun" counts to show all inpatient readmissions and has nothing to do with any of the core diagnosis. In order to perform that trick I do the same basic steps but I enhance my where clause to only look for encounters that have a core diagnosis of COPD and I simply name my flags and other fields differently.

```
Left Join (Encounters)
LOAD
EncounterID,
IF(MRN = Previous(MRN),Dual('Yes', 1),Dual('No',0)) as [COPD
IsReadmission Flag],
IF(MRN = Previous(MRN),Previous([Discharge Dt/Tm])) as [COPD
Previous Discharge Date],
IF(MRN = Previous(MRN),Previous(EncounterID)) as [COPD
Previous EncounterID],
IF(MRN = Previous(MRN),NUM(Interval([Admit Dt/Tm]-
Previous([Discharge Dt/Tm])),'#,##0.00´)) as [COPD
Readmission Difference],
IF(MRN = Previous(MRN),IF(Interval([Admit Dt/Tm]-
Previous([Discharge Dt/Tm])) <= 30.0, Dual('Yes', 1),
Dual('No',0)), Dual('No',0)) as [COPD IsReadmission within
30],
IF(MRN = Previous(MRN),IF(Interval([Admit Dt/Tm]-
```

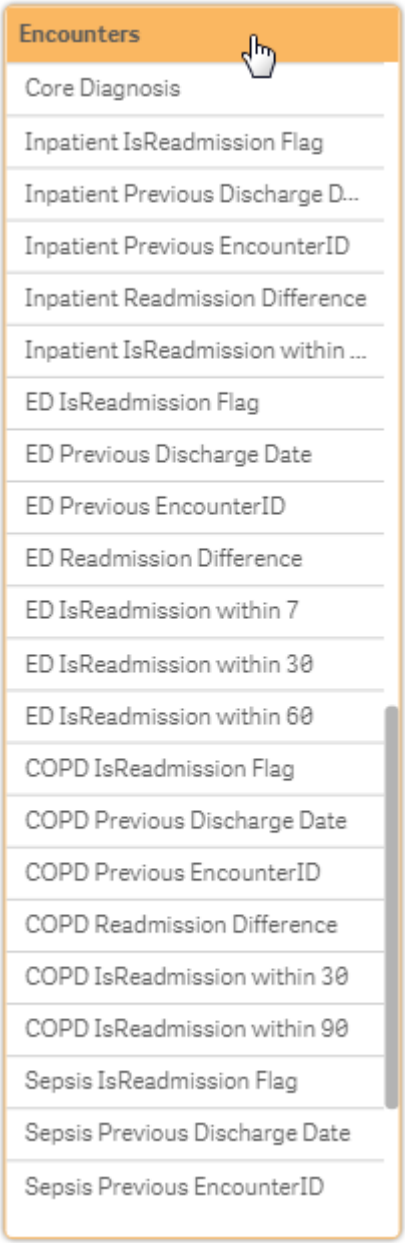
```
Previous([Discharge Dt/Tm])) <= 90.0,'Yes', 'No'), 'No') as
[COPD IsReadmission within 90]
Resident Encounters
Where [Patient Type] = 'Inpatient' and  [Core Diagnosis] =
'COPD'
Order by MRN, [Admit Dt/Tm];
```

And just when you think I've pulled as much handkerchief out of my sleeve that it can possibly I hold I do the same steps for Sepsis this time.

```
Left Join (Encounters)
LOAD
EncounterID,
IF(MRN = Previous(MRN),Dual('Yes', 1),Dual('No',0)) as
[Sepsis IsReadmission Flag],
IF(MRN = Previous(MRN),Previous([Discharge Dt/Tm])) as
[Sepsis Previous Discharge Date],
IF(MRN = Previous(MRN),Previous(EncounterID)) as [Sepsis
Previous EncounterID],
IF(MRN  =  Previous(MRN),NUM(Interval([Admit  Dt/Tm]-
Previous([Discharge  Dt/Tm])),'#,##0.00′))  as  [Sepsis
Readmission Difference],
IF(MRN  =  Previous(MRN),IF(Interval([Admit  Dt/Tm]-
Previous([Discharge  Dt/Tm]))  <=  30.0,Dual('Yes',  1),
Dual('No',0)), Dual('No',0)) as [Sepsis IsReadmission within
30],
IF(MRN  =  Previous(MRN),IF(Interval([Admit  Dt/Tm]-
Previous([Discharge Dt/Tm])) <= 90.0,'Yes', 'No'), 'No') as
[Sepsis IsReadmission within 90],
IF(MRN  =  Previous(MRN),IF(Interval([Admit  Dt/Tm]-
Previous([Discharge Dt/Tm])) <= 120.0,'Yes', 'No'), 'No') as
[Sepsis IsReadmission within 120],
IF(MRN  =  Previous(MRN),IF(Interval([Admit  Dt/Tm]-
Previous([Discharge Dt/Tm])) > 120.0,'Yes', 'No'), 'No') as
[Sepsis IsReadmission > 120]
Resident Encounters
```

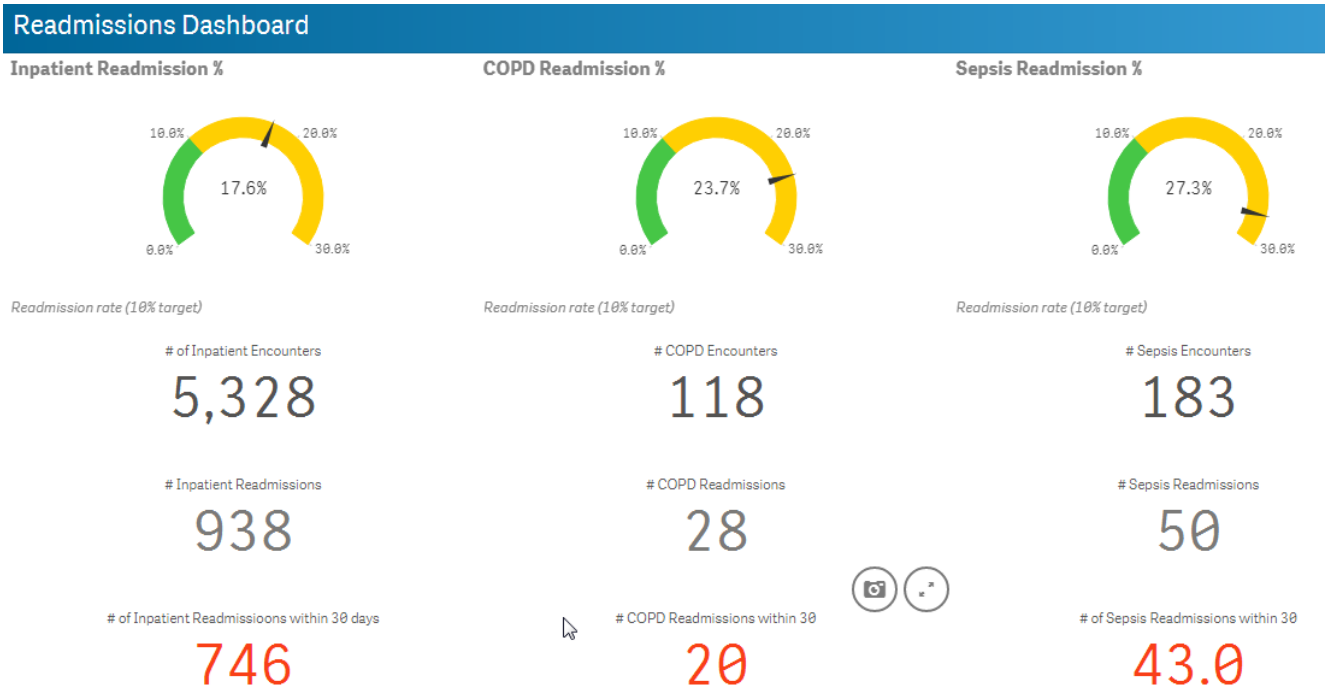
```
Where [Patient Type] = 'Inpatient' and  [Core Diagnosis] =
'Sepsis'
Order By MRN, [Admit Dt/Tm];
```

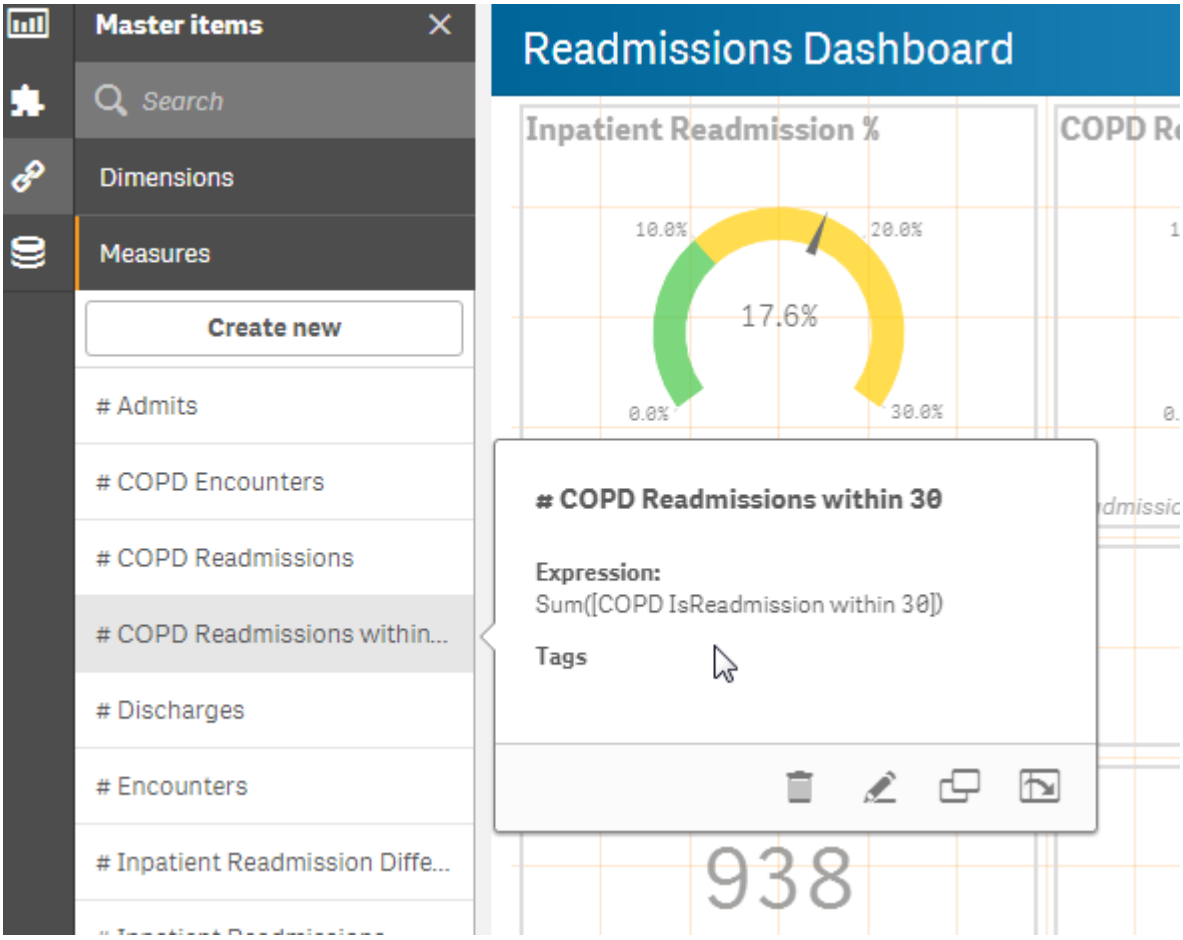
And then for AMI. And then for CHF. And then for … Oh you know the handkerchief can go on forever and eventually we end up with a data model that includes all of these awesome fields that didn't exist when we began so that we can actually do our work.



Voila a Readmissions Dashboard

Not only can we then provide a really nice looking dashboard which includes accurate statistics we can do it using very simple expressions that are incredibly fast.





Click this link to get the entire Readmissions Code start script: [ReadmissionsCodeScript]