Making Sense of Section Access in Qlik Sense By Dalton Ruer aka Qlik Dork

What this covers: This guide will help you understand how to limit access to rows and columns within Qlik Sense applications once a user has been authorized to access the application itself.

What this does not cover: This guide is not intended to help you understand how to understand the Security Rules that would dictate whether users are allowed to access an application itself.

Understanding the data: For the purposes of this document there are 2 key tables that were used to drive the use cases. The primary table is a very simple Encounter table. That is the Healthcare term used to define a patient is checked in for some reason. For other business it could represent a purchase or any type of customer encounter. The second table is quite simply a survey response table. As you can imagine not every encounter has a survey associated with it, but surveys are associated with the encounter that triggered the survey.

Encounters data: There are 100 records that contain an EncounterID, and a date for the encounter like the following:

EncounterID	AdmissionDate	ADMITTINGPHYSICIANID
1	1/1/2016	1
2	1/2/2016	2
3	1/3/2016	3
4	1/4/2016	4

Survey data: There are 20 records that contain a SurveyID as well as the EncounterID that triggered the survey. It also includes a few details like the score and it also contains the Supervisor ID and Name for the person responsible. In a non healthcare environment it could be the customer service manager, the sales rep or whatever. As it will be important for you to reference all 20 records are displayed

SurveyID	EncounterID	SurveyScore	SUPERVISORNAME	SUPERVISORID
1	5	7	Nurse A	1
2	10	8	Nurse B	2
3	15	6	Nurse C	3
4	20	7	Nurse D	4
5	25	5	Nurse E	5
6	30	4	Nurse A	1
7	35	7	Nurse B	2
8	40	8	Nurse C	3
9	45	9	Nurse D	4
10	50	9	Nurse E	5
11	55	9	Nurse A	1
12	60	10	Nurse B	2
13	65	5	Nurse C	3
14	70	6	Nurse D	4
15	75	4	Nurse E	5
16	80	3	Nurse A	1
17	85	2	Nurse B	2
18	90	8	Nurse C	3

19	95	7	Nurse D	4
20	100	9		

We want to allow any of the supervisors to access our application that shows the data. That access is handled outside the scope of this guide. However, as you can imagine we don't want the poor supervisor with the really low survey scores to feel bad. So we have to provide a way for Nurse A to only be able to access the data that pertains to her, but not see the data for the other nursing supervisors. That's where we start.

Step 1: Creating the "Section Access" Sections in our Load Script

There are 2 parts to the defining security within Qlik Sense. One key word to tell the system that you are defining your access and the other part is just as important … telling it that you are done declaring your access.

```
SECTION ACCESS;

{ Any authorizations would go here }

SECTION APPLICATION;
```

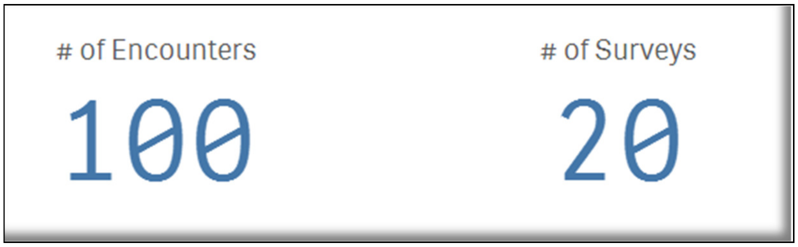
Note: It is important to note that once you declare the SECTION ACCESS element in the code you are telling Qlik Sense that nobody should be granted "ACCESS" unless you have define it. You are not limiting who gets access to what … you are defining who gets access to what.

Step 2: Granting access to rows of data

The simplest way to begin playing with Section Access is to simply use an INLINE load of data. The following code would tell Qlik Sense that you want the user "drr" who is part of the QTSEL domain to have access to values that are associated with SUPERVISORID number 1. ***The domain is required for identifying your users.***

```
AUTHORIZATION:
LOAD * INLINE [
   ACCESS,          USERID,   SUPERVISORID
   USER,   QTSEL\drr,                     1
];
```

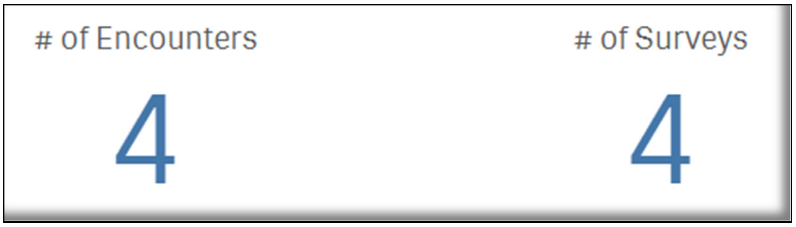
Prior to applying this restriction our user "QTSEL\drr" would have seen the following in our application:



# of Encounters 100

# of Surveys 20

But after applying the security through Section Access they would see the following:



# of Encounters 4

# of Surveys 4

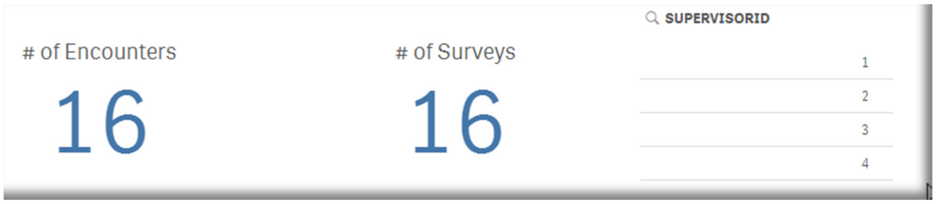
Typical examples for Section Access only refer to data in 1 table. The reason that I chose to create this guide from data for 2 tables is to ensure that you understand the concept COMPLETELY. You are granting access to users via all associations to what you are permitting. While there are 100 encounters, only 4 of them are associated with Supervisor 1 and therefore, user “QTSEL\drr” only gets to see those 4 encounters. Keep that in mind as you determine what field(s) you are going to use within your application to provide access.

Step 3: Using an Asterisk “*”

One of the things you will find in the help for Section Access is that you can use an asterisk “*” to define your security access. However, it is often misinterpreted. You would likely see a reference like below and assume that our user “QTSEL\drr” has access to ALL supervisors. However, as the image below the code illustrates that is not what the asterisk “*” means. It really means “grant the user access to all of the values that have been listed for access. In other words our user gets to see supervisors 1-4, but not supervisorid 5 because it isn’t listed.

Note: The code below demonstrates 2 things in addition to the “”. First it shows you can grant access to users from multiple different domains. Second it demonstrates that you are able to add multiple lines if you wish users to have access to more than 1 value.*

```
SECTION ACCESS;
AUTHORIZATION:
LOAD * INLINE [
   ACCESS, USERID,                     SUPERVISORID
   USER,   QTSEL\drr,                  *

   USER,    IPortal\n_one,             1
   USER,   IPortal\n_two,              2
   USER,   IPortal\n_three,            3
   USER,   IPortal\n_four,             4

   // You can define as many rows as needed to grant access to multiples
   USER,   IPortal\s_sigma,            1
   USER,   IPortal\s_sigma,            2
];

SECTION APPLICATION;
```



# of Encounters

16

# of Surveys

16

SUPERVISORID

1

2

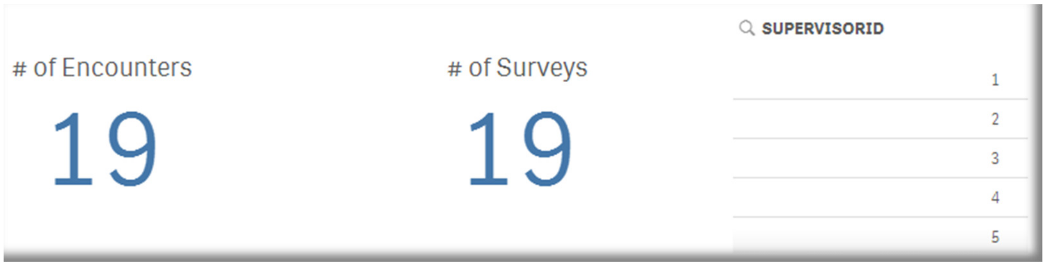
3

4

Step 4: Granting Access to ALL records

You might be thinking that shouldn’t be a problem as you would be listing all of the supervisors like the code below so our user “QTSEL\drr” will have access to all of the surveys then. Notice in the screenshot below the code that that isn’t the case. One of the surveys in our data model is missing a supervisor id, therefore our user only sees 19 of the surveys and still only gets to see the 19 encounters that are associated with those survey records.

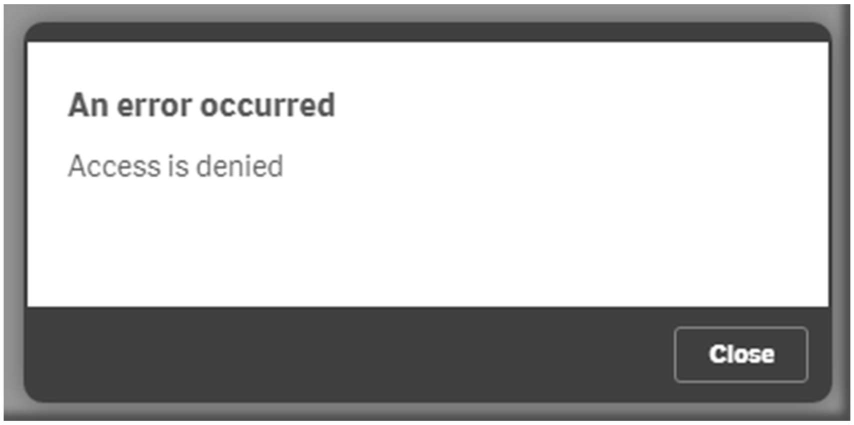
```
SECTION ACCESS;
AUTHORIZATION:
LOAD * INLINE [
   ACCESS, USERID,             SUPERVISORID
   USER,   QTSEL\drr,                    *

   USER,   IPortal\n_one,                1
   USER,   IPortal\n_two,                2
   USER,   IPortal\n_three,              3
   USER,   IPortal\n_four,               4
           USER,   IPortal\n_five,       5

   // You can define as many rows as needed to grant access to multiples
   USER,   IPortal\s_sigma,        1
   USER,   IPortal\s_sigma,        2
];

SECTION APPLICATION;
```



One of the things that I saw in posts about Section Access was that you could use put the user in the code but leave the restriction blank. In other words code like the following:

```
   ACCESS, USERID,              SUPERVISORID
   USER,   QTSEL\drr,
```

I learned really quickly that doesn't quite yield the results I was hoping for. Instead of having no limits for my user I was greeted with this error message:



An error occurred

Access is denied

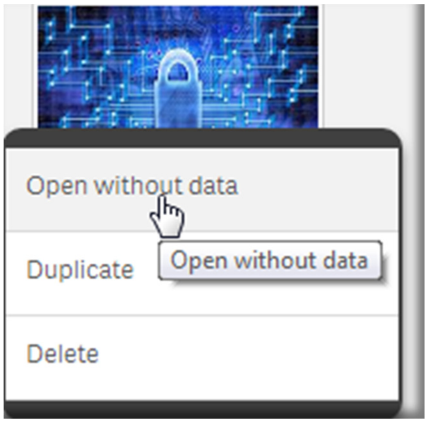
Before continuing I suggest you actually take the time to think through why that would occur. If you can figure out why you will know you really understand.

It goes back to the primary premise I shared … you are declaring access to values, and a blank for a user simply means … they get no access. Thus, Access is denied because they have no access to any data at all.

There is 1 thing I probably should mention at this point … you don't have to declare users with the USER keyword for the ACCESS type. You can use the keyword ADMIN instead. Thus the following is what I really needed to create. But whoa is me because I've now locked myself out of the application that I'm building.

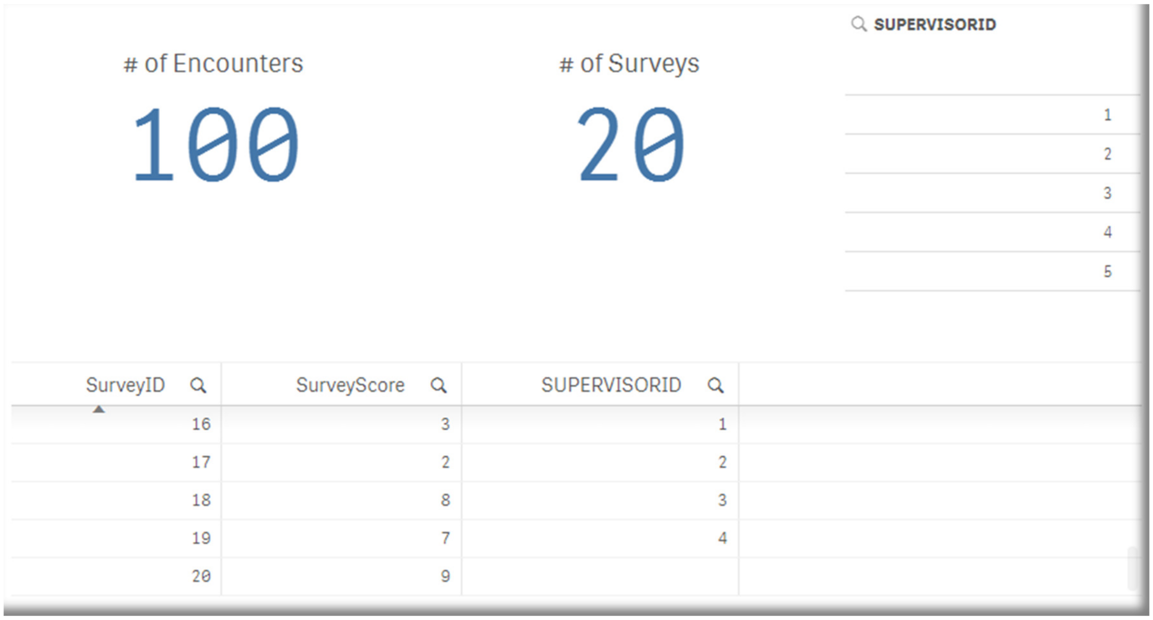
```
ACCESS,         USERID,          SUPERVISORID
ADMIN,          QTSEL\drr,
```

Fortunately the great minds on the product team knew I was a little silly and might do something like that so they provided a way for us to get back into our applications when we do that. Simply right click on the application and choose "Open without data"



The system will then let you into the code so that you can correct your access issues.

Now back to my story … I changed my account to ADMIN once I did that I was able to see all of the data. Including all 5 supervisors, the survey that has no survey associated … and in fact I'm once again able to see all of the encounters. When you declare a user as ADMIN you are saying … that user should not be restricted in any way if you leave the field value "blank".



# of Encounters

100

# of Surveys

20

SUPERVISORID

1
2
3
4
5

SurveyID	SurveyScore	SUPERVISORID
16	3	1
17	2	2
18	8	3
19	7	4
20	9	

Step 5: Loading from a file or a database

How you define the authorization for Section Access really doesn't matter to Qlik Sense. All you have to do is provide the values. So feel free to replace the INLINE code with code to read in an Excel file for your security or read the values from a database. Which then provides you the flexibility to reuse your security in multiple applications as well as modify your security without having to edit your applications themselves. Simply change your security definitions and the next time your applications are refreshed your security is refreshed.

```
SECTION ACCESS;

AUTHORIZATION:
LOAD
   "ACCESS",
   "USERID",
   SUPERVISORID
FROM [lib://Tech Thursday/Security.xlsx]
(ooxml, embedded labels, table is Sheet1);

SECTION APPLICATION;
```

Step 6: Securing ALPHA fields

More often than not your security will be on numeric fields like the cases above. But what if we chose to implement security based on the supervisor's name instead of their ID?

```
   ACCESS,           USERID,                 SUPERVISORNAME
   USER,   IPortal\n_one,                    'Nurse A'
   USER,   IPortal\n_two,                    'Nurse B'
   USER,    IPortal\n_three,                 'Nurse C'
   USER,    IPortal\n_four,                  'Nurse D'
   USER,    IPortal\n_five,                  'Nurse E'
```

I must confess I have no possible explanation for or understanding why, but Qlik Sense absolutely will not allow access to field values that are not totally UPPER cased. Which leaves us all with 2 options for implementing alpha based security. We can for the source field to be upper case when we read it in. But I frankly don't like that option as it's hard for end users to read all upper cased field values. Our other option is to create a second field in the table instead like follows so that we can still display the field value in the case it comes in as, and yet have a fully upper cased field that we can apply security for.

```
            SUPERVISORNAME,
            Upper(SUPERVISORNAME) as SUPERVISOR_SECURITY,
```

Once we have the upper cases values in our data that we can associate our security with we can simply utilize that field as part of our access instead of the original field. Notice in the following that when you are utilizing your alpha fields you can either wrap your values in single quotes, double quotes or just put the values in even when they contain spaces.

```
   ACCESS,           USERID,                 _SUPERVISOR_SECURITY
   USER,             IPortal\n_one,                  'NURSE A'
   USER,             IPortal\n_two,                  "NURSE B"
   USER,             IPortal\n_three,                NURSE C
   USER,             IPortal\n_four,                 NURSE D
   USER,             IPortal\n_five,                 NURSE E
```

Step 7: Granting Access to Groups

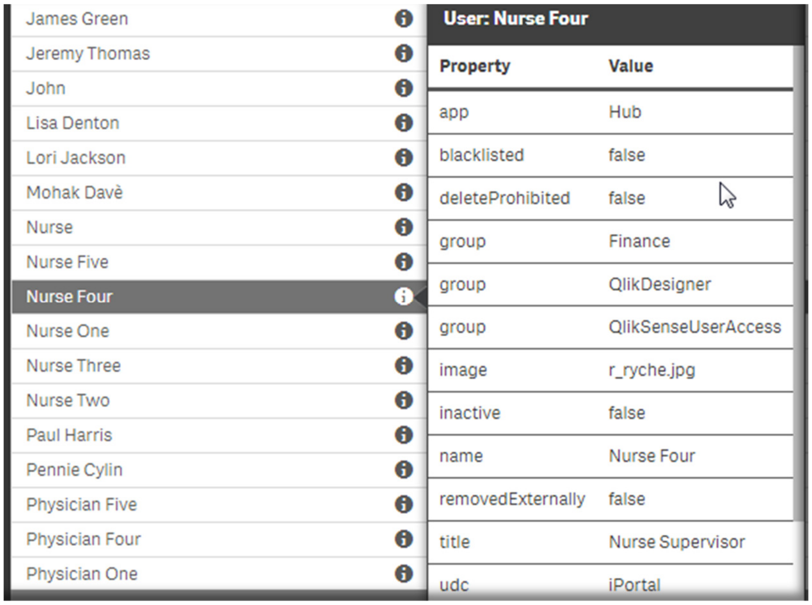
Sorry it has taken this long to get to what many of you were thinking from the first minute you opened the document … "How do you deal with GROUPS rather than single users?"

You are absolutely welcome to assign security based on groups. All you need to do is introduce a new column to your authorization table with the title GROUP. In the example code below I am telling Qlik Sense that my user id should be and ADMIN and the group is … Oh boy this gets tricky now after I've already explained how the asterisk "*" works for field values themselves. The same character is used to indicate that you actually want all Users or Groups and doesn't mean "all that are listed."

So back to my explanation … I am telling Qlik Sense that my user id is an ADMIN user, that can be part of any group at all, and that I don't want any restrictions at all for it since I've left the SUPERVISORID field "blank." I am also telling Qlik Sense that any users that are part of the group called "Clinical" are able to access the data associated with supervisors 1 and 2.

```
ACCESS,     USERID,                   GROUP,       SUPERVISORID
ADMIN,      QTSEL\DRR,                *,
USER,       *,                        Clinical,    1
USER,       *,                        CLINICAL,    2
USER,       *,                        clinical,    3
```

The Qlik Sense Management Console (QMC) provides two handy dandy things that aid you in debugging Section Access that has to do with Groups. First let's assume supervisor 4 calls and says "I can see the application but I keep getting an error that says I have no access." Within the QMC all you have to do is look at the user record itself. If you click on the little information icon the system will display all of the properties that the system is maintaining for the user, including the groups they are a part of. Notice in this case Nurse Four is not part of the Clinical group so they shouldn't have access.

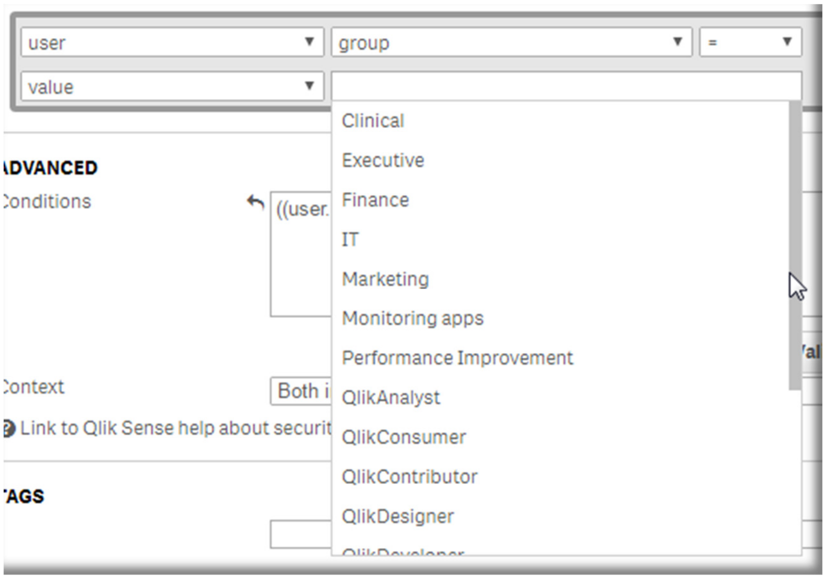


James Green
Jeremy Thomas
John
Lisa Denton
Lori Jackson
Mohak Davè
Nurse
Nurse Five
Nurse Four
Nurse One
Nurse Three
Nurse Two
Paul Harris
Pennie Cylin
Physician Five
Physician Four
Physician One

User: Nurse Four

Property	Value
app	Hub
blacklisted	false
deleteProhibited	false
group	Finance
group	QlikDesigner
group	QlikSenseUserAccess
image	r_ryche.jpg
inactive	false
name	Nurse Four
removedExternally	false
title	Nurse Supervisor
udc	iPortal

What if you are just starting to build out the security for an application and aren't quite sure what all of the groups are that are even available? The QMC comes in handy for this as well. All you have to do is pretend you are going to create a new security rule. If you choose "group" for the user value and then move the cursor to the value field, the system will provide you with the list of all of the "group" names it is aware of. If the group name isn't listed there, you can't build security for it.



user ▼ group ▼ = ▼

value ▼

Note: Unlike field values which we had to use upper case for, the user names and the group names are actually totally case insensitive. If you look back to the group code you will see that I used all 3 variations for the group name "clinical" and they all work wonderfully.

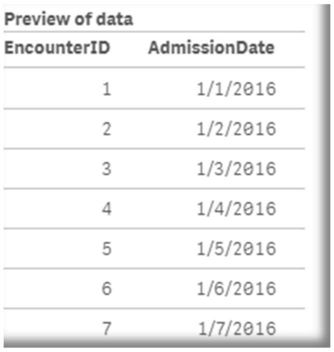
Step 7: Locking down columns

In all of the previous examples we were trying to limit the rows that end users can see. Supervisor 1 should only see “their” data. Supervisors 3 should only see “their” data. After all the goal was to protect feelings from being hurt. There is also a completely different type of security which involves the desire to hide columns of data rather than the rows of data. Meaning “I don’t care if Supervisor 3 can see all of survey data, I just don’t want them to see sensitive information like SSN, pay rates etc.

For this example we will pretend that the ADMITTINGPHYSICIANID is a highly sensitive data field that I don’t want users to see. All I need to do is add a new column to my authorization table called OMIT and then declare that I don’t want the user to see the ADMITTINGPHYSICIANID field.

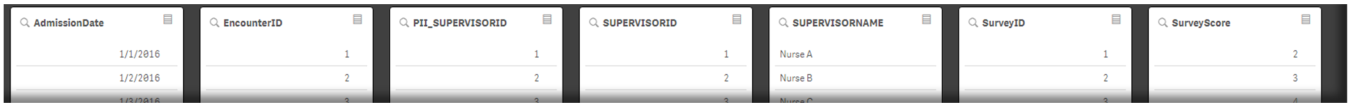
```
ACCESS,       USERID,         SUPERVISORID,          OMIT
ADMIN,        QTSEL\drr,                       ,     ADMITTINGPHYSICIANID
```

When the application is loaded even in the data manager view that field is no longer even displayed in the data manager or in the selections tool. The field simply doesn’t exist for the user.



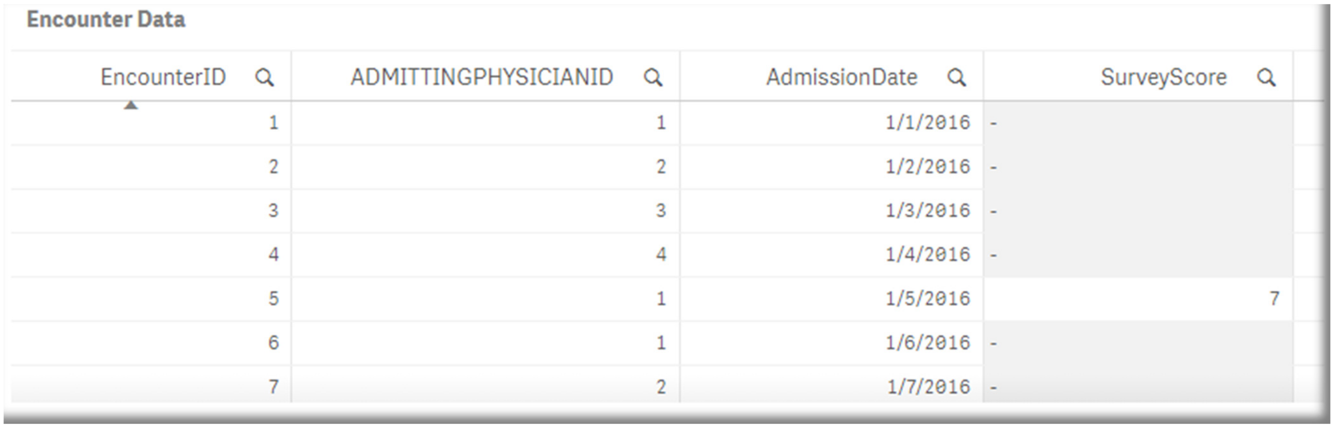
Preview of data

EncounterID	AdmissionDate
1	1/1/2016
2	1/2/2016
3	1/3/2016
4	1/4/2016
5	1/5/2016
6	1/6/2016
7	1/7/2016



AdmissionDate	EncounterID	PII_SUPERVISORID	SUPERVISORID	SUPERVISORNAME	SurveyID	SurveyScore
1/1/2016	1	1	1	Nurse A	1	2
1/2/2016	2	2	2	Nurse B	2	3
1/3/2016	3	3	3	Nurse C	3	4

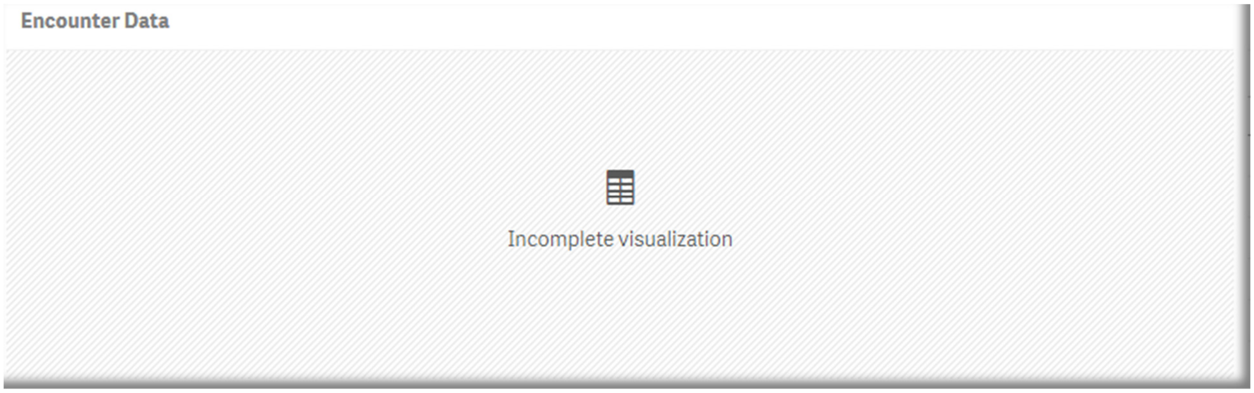
Which is a great thing in that we’ve locked that field out of view for the user. But there can also be consequences. Assume the application was built with a chart like this:



Encounter Data

EncounterID	ADMITTINGPHYSICIANID	AdmissionDate	SurveyScore
1	1	1/1/2016	-
2	2	1/2/2016	-
3	3	1/3/2016	-
4	4	1/4/2016	-
5	1	1/5/2016	7
6	1	1/6/2016	-
7	2	1/7/2016	-

Guess what happens when you apply Section Access and forcefully OMIT a field? If you guessed that the system adjusts and just doesn't include that column you are incorrect. Unfortunately, the system simply responds with "Incomplete visualization" and none of the data is shown. So consider that consequence before you begin and perhaps work through using QMC security rules to support two different sheets one that is available to users who will have access to the field(s) and one that is for users who won't have access.



Encounter Data

Incomplete visualization

Step 8: Handling combinations of row and column security

Just when you thought you had all of the details you needed to handle both row and column level security in Qlik Sense some wise guy says “That’s a good start. But here is my situation … I need you to combine them together so that a given user can see three supervisors data, but they should only see the supervisor names for 1 of them.” Or in your world it might be “This supervisor should see their own employees including salary information. They also need to see the employees and job roles for other departments but they absolutely shouldn’t be able to see the salaries for the employees in the other departments.”

Oh momma. Now we have our hands full. Because not only do we need to handle Section Access, we also have to do some nifty data modeling as well. Logically what we need to do is create a data model where the PHI/PII data fields are in a separate table with an alternative field key. That way we can identify the key values that they should see as rows, and then we also have the ability to identify the keys that they should for the associated PHI/PII.

Before I show you the data modeling that is done let’s take a look at the security model itself and simply set this use case up.

Notice in the code (next page) that in the Section Access section I’m referring to a field called USERRIGHTS which doesn’t even exist in our data. Instead I’m creating a table using an INLINE load to define that field value. We’ll get to that table in a second.

For the user “QTSEL\DRR” I’m saying they should have access to anything defined for the value “ALL.” For the user “IPortal\n_one” I’m saying they should have access to anything defined for the value “1.” For the user “IPortal\s_sigma” I’m saying they should have access to anything defined for the value “OB.” Those values themselves mean absolutely nothing, they are simply values, until I define them in my table where I load the USERRIGHTS field.

If we turn now to the table where I load the USERRIGHTS field you will notice that there are actually multiple rows for ALL and OB while the numbers have only 1 row. Nothing new there in terms of security and simple stuff.

The good stuff comes in the other columns for this table. The “ALL” permissions indicate that it should allow access to supervisor ids 1, 2, 3, 4 and 5. The fun stuff comes in the fact that when we load the data in this manner we can also define additional columns we want to limit the rows for. In this case I’m saying “the user can see supervisor id 1, 2 and 5 and they are welcome can see the PII for those same supervisors. The users associated with “ALL” are also welcome to see the data for supervisors 3 and 4, however, they are not allowed to see the rows of data of PII for those supervisors.

At this point just try and digest the logical concept. Read the following code (next page) and see if you can at least at a high level explain what data the users “IPortal\n_one” and “IPortal\s_sigma” should see:

```
SECTION ACCESS;

AUTHORIZATION:
LOAD * INLINE [
   ACCESS,          USERID,                      USERRIGHTS
   USER,            QTSEL\drr,                   ALL
   USER,            IPortal\n_one,               4
   USER,             IPortal\s_sigma,            OB
];

SECTION APPLICATION;

REDUCTION:
LOAD * INLINE [
   USERRIGHTS, SUPERVISORID,          PII_SUPERVISORID
   ALL,                  1,                    1
   ALL,                  2,                    2
   ALL,                  3,
   ALL,                  4,
   ALL,                  5,                    5
   1,                    1,                    1
   2,                    2,                    2
   3,                    3,                    3
   4,                    4,                    4
   4,                    5,
   5,                    5,                    5
   OB,                   3,
   OB,                   4,                    4
   OB,                   5,                    5
];
```

If you said that “IPortal\n_one” should see the data for supervisor id 1 and the PII for them, you fell for my trick. I actually defined that user to have the USERRIGHTS values for 4 not 1. If you said that the user would see the data for supervisors 4 and 5 but only see the PII for supervisor 4 then you are absolutely correct.

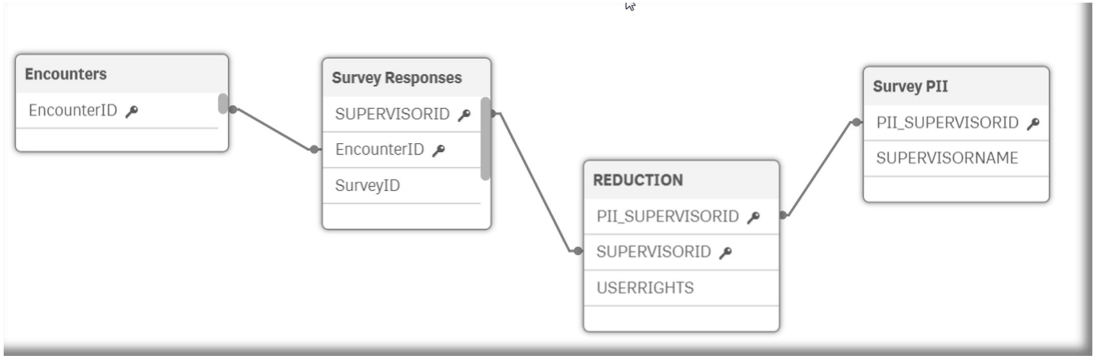
If you said that “IPortal\s_sigma” should see the data for supervisors 3, 4 and 5 but only see the PII for supervisors 4 and 5 you are absolutely correct.

Now for the data modeling that makes the magic happen. I simply separate the PHI/PII fields from the base tables and create new tables that store that data. Notice in the Survey Responses table I load the basic data, but I do not load the SUPERVISORNAME column.

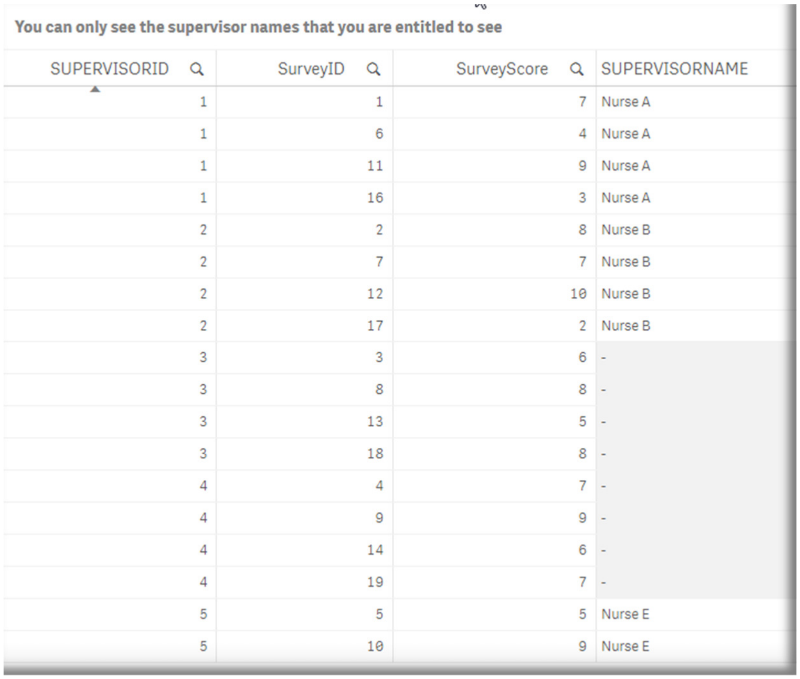
Then I simply create a new table and load only the supervisor id, but I alias it as a brand new field that can be tied to “row level” security for this new table and then I load the PHI/PII data in this new table.

```
[Survey Responses]:
LOAD [SurveyID] ,
            SUPERVISORID,
            [SurveyScore],
            [EncounterID]
 FROM [lib://AttachedFiles/SectionAccessData.xlsx]
(ooxml, embedded labels, table is [Survey Responses]);

[Survey PII]:
LOAD distinct  SUPERVISORID as PII_SUPERVISORID,
            [SUPERVISORNAME]
FROM [lib://AttachedFiles/SectionAccessData.xlsx]
(ooxml, embedded labels, table is [Survey Responses]);
```

The final data model would then appear like this:



And our user would be able to access any chart without an issue because as far as the Qlik Sense visualization engine is concerned the field SUPERVISORNAME does exist, it is simply missing values just and simply shows the missing values as "-" like it does for other values that are missing like the last survey record that had no supervisor id at all.



You can only see the supervisor names that you are entitled to see

SUPERVISORID	SurveyID	SurveyScore	SUPERVISORNAME
1	1	7	Nurse A
1	6	4	Nurse A
1	11	9	Nurse A
1	16	3	Nurse A
2	2	8	Nurse B
2	7	7	Nurse B
2	12	10	Nurse B
2	17	2	Nurse B
3	3	6	-
3	8	8	-
3	13	5	-
3	18	8	-
4	4	7	-
4	9	9	-
4	14	6	-
4	19	7	-
5	5	5	Nurse E
5	10	9	Nurse E

Hopefully this document helps you make sense of section access as it is implemented in Qlik Sense.