ESRBI Advanced Report Writing Workshop

Script

Introduction (00:00 – 01:02)

Hello, welcome to the ESR Business Intelligence webinar workshop on advanced report writing, my name is Christopher Holroyd and this webinar follows on from the basic report writing and the intermediate report writing webinars we hosted in autumn 2020. We will run through some of the more advanced techniques and functions available in ESRBI and also why you may use some of these functions including problems that can be solved using some of the functions covered today. Hopefully you may hey get some ideas when designing your own analyses and dashboards

Agenda (01:02 – 02:28)

Here is the agenda we are going to cover today starting with commonly used functions in ESRBI including aggregate functions, calendar date functions and expressions. We will then cover dynamic date filtering which is one of the questions we have been asked in advance of the workshop and then lastly, we will cover filtering based on the results of another analysis. With all these topics we will try to cover some of the problems you may solve using these techniques as well as using the techniques themselves.

Aggregate Functions
SUM BY (02:28 – 10:08)

Ok we are in ESR Business Intelligence, we are using the BI Administration URP and we are using a training environment so all the data is fictitious. The first thing we're going to look at is a couple of aggregate functions including SUM and RANK however we going to take the functions to step further by including the BY clause. We're going to build a basic analysis here and filter the date as should always be done when creating an analysis. We are going to enter some assignment details including assignment number and assignment FTE. We run it and look at the data, we return a list of assignments and FTE values. When we look at the sum function that we going to look at introducing the BY clause so we will be grouping by a specific level. It's something we get asked about quite a lot and in this example we will be by grouping by staff group, occupation code and job role however it's the same principle when grouping by multiple organisation levels. You may wish your

data to be summed by a specific level but include levels with a lower granularity, in our example we are going to sum by staff group but include items of lower granularity such as occupation code and job role. It's the same principle if you are looking to sum by an organisation level higher up the hierarchy but also include organisation levels lower down the hierarchy. The values we are going to sum are assignment fte values. Including staff group within the analysis you can see assignment FTE values are still returned per assignment because there is no aggregation on the FTE values. We now going to put some aggregation on the FTE values and whenever we using a function in ESRBI we always try to highlight the function button in the lower left corner when editing a formula. Clicking on the button gives you a list of all the available functions, we are looking at aggregate functions so we will open up the aggregate folder. If we click on a function you are given a description of what it does which is helpful. You are also presented with how to set up the function within the formula. We will put the sum into the formula and test the results. Looking at the results we now return a sum of FTE by staff group. The sum function always groups to the lowest level of granularity within the analysis, in our case staff group. At this stage we could just use the Total FTE measure which would do exactly the same thing as our sum function. What we are going to do is fix the grouping level within our sum function so where the Total FTE fact will always group to the lowest level of granularity, our sum function will group to our choice of granularity. We will set the group by within the sum function. What we've done here is added job role to the analysis and you can see the sum function sums to the job role field which is lower granularity than the staff group. If we put in occupation code the sum will group to occupation code field which is lower granularity than both staff group and job role. We want to group by staff group but leave occupation code and job role within the analysis and to do that we enter a by clause. To do that we add BY and the field we would like to group by. We run and test the report, you will see we return repeating values but the values are summed by the staff group field correctly. We are introducing the BY clause because we are going to use it in further examples throughout the webinar.

RANK BY (10:08 – 17:50)

The second function we are going to look at is the rank function. We are going to use it to resolve a problem we get asked about quite regularly which is how do I remove multiple rows per employee or assignment etc. Using aggregation is a way we can remove multiple rows. We are building a new report based on the absence subject area which enables us to return multiple absences per assignment. The method is exactly the same were we to use EIT information such as professional registrations, DBS or appraisal information. We will set up a basic absence analysis using a date period and enter some basic assignment and absence information. We will test the results and we can see a number of assignments are returned with single absences and towards the bottom some are returned with multiple absences. the problem we are going to see how we return the latest absence records. Again, the method is the same whether we were returning the latest absence record or the latest created date record or the latest last updated date record. We will add absence start date into the report again so we can set up a

custom formula. We are back in the functions folder within the aggregate folder, we will find the rank function and we will have a look at the description. In summary the rank function is designed to return a numerical value for the field you are ranking, in our case we are ranking a date field so the latest date will return 1, the second latest will return 2 etc. We can update the formula, set the column heading and test the results. And we can see the results returned look a little strange, we have some very large numbers but when we think about it we have asked BI to rank all the absence start dates rather than start date per assignment number. So we need to use the by clause again to ensure that the aggregation is grouped by the item as required, in this case assignment number. To update the formula it's the same method as updating the sum formula, before the close bracket we simply add in BY and the field we would like to group by. We then test the results and we can see the data looks as required. Looking at the example with seven absence records we can now see we return 1 to 7 for the single assignment. The last step in the process is we can filter for the bottom row which returns 1 in the rank column and that will ensure that we only return the latest absence start date row per assignment across the entire dataset. We add a filter based on the column and set the filter up to only return rows where the value is 1. Now checking the results we can see per assignment only one row is returned which is the latest absence start date row. Lastly we could remove our workings if required because we have a filter in place and so the column is no longer required.

Date Functions
TIMESTAMPADD (17:50 – 25:00)

The next topic we are going to cover is date functions. The two date functions we are going to look at are Timestampadd and Timestampdiff. On their own these functions are quite straightforward, but we are going to look at how we can use them together to return more complex results. As before we can check the functions folder and if we look at the date folder towards the bottom we can see Timestampadd and TImestampdiff. To start with we are going to look that how we enter a date into a formula within ESRBI. There is a specific way to enter a date which we will enter now. TIMESTAMP ‘2020-01-01 00:00:00’. We will now enter a date filter which is different to entering a date in the column, the two are separate. We will filter for the date the video was taken which is 05/08/2021. Ok the next thing we are going to look at is using the Timestampadd function. If we go back into the folder and have a look at the setup we can see the function is in three parts the first part is the interval and we can set an interval to whatever we like, day, month, year or second etc and it is always entered as SQL_TSI_DAY or SQL_TSI_MONTH etc. The second part of the function is how many intervals we would like to move e.g -7, +7, -10 or +10 as required and the final part of the function is the starting timestamp (date). If we check the results you can see we have now moved the date back 7 days. The next thing we will look at is changing timestamp from a fixed date to a more dynamic date, to do this we will use a function called current_date which simply returns the current date. If we check the results we should now be moving back 7 days from the current date rather than the fixed date we entered. The

current date of this report is 05/08/2021 so we can see the formula is working correctly.

TIMESTAMPDIFF (25:00 – 28:00)

The next thing we will look at is using the Timestampdiff function. If we have a look in the function folder we can see the setup which is similar to timestampadd. We need to add an interval and then a from and a to date. The function will then return the number of intervals between the two dates. We will enter the formula now, setting our interval, our from date and our to date. If we test the results we can see the formula is working as expected. We can go back to the criteria page and update the formula, changing the interval and test the results again.

Time.Date (28:00 – 32:00)

The next item we are going to look at is using time.date. We can see that we are filtering time.date for 05/08/2021 so we know time.date will always equal that date and we can use this within our formulas to make them more dynamic. Rather than including a fixed date in the formula we can use time.date which will then change depending on how we filter the time.date (effective date) of the analysis. Where we may use dashboard prompts and set up filters to work with dashboard prompts we are then able to use the date set in the dashboard prompt within our formulas which makes the formula even more dynamic. So we will add time.date to our formula, check the results and we can see that update is returned as per the filter we have setup. The next thing we can do is add time.date to our timestampadd formula and we return time.date -7 days in our results. Lastly we can add time.date to our timestampdiff formula. Now imagine we have set the filter to Is Prompted and we are using this analysis within a dashboard. We have set up a prompt so our users can change the effective date of the analysis, by including time.date within our formulas they are dynamically updated depending on what our users have entered into the dashboard prompt, rather than having fixed dates within the formula.

Expression Function
CASE (32:00 – 44:55)

Now we are going to look at using an expression function. The function we are going to look at is the CASE statement. If anyone has used the IF function in Excel it is very similar. It is a straightforward function but we will look at how we can use it embedded within other case functions. Lastly we will then look at how we have used it within a formula in the NHS standard dashboards. We will set up a quick case statement here, CASE WHEN 1=1 THEN….. If we look within the functions folder, we get a description of how to use a case statement. In our example we are going to use the case statement within a case statement so Case When 1 = 1 then… here we can use this second case statement Case When 2 = 3 then... and build up a more complex case statement. As long as each Case statement is ended properly then the formula will work correctly. We're now going to look at how we have used some of the examples we have gone through within a formula in the NHS

Standard Dashboards. The formula we going to look at is the average hours worked formula within the working time regulation report. Rather than being based on a calendar date (time.date) it is based on a payroll period or an accounting period and this is important because this is used this within the formula we are going to look at. So if we open up the average hours worked formula we will attempt to simplify and understand the formula. This is something we get asked about quite a lot, "I have found a formula in an NHS Standard Dashboard but I'm not sure where to start with understanding how it works". When we actually look at the formula we can see that it is actually a series of SUMs, Case Statements and other simple functions but used together to create a more complex formula. So we will have a look at how we can break this formula down. If we look at the bottom of the formula we can see there is a group by clause 'BY assignment number' so we can split the formula out to start with by placing the sum on the top row and the by assignment number on the bottom. We know everything in the middle is being summed by assignment number. If we start splitting out the case statements we can see the first case statement relates to weekly payrolls, we can then split out the next sum and next Case statement and we find the next By clause which relates to the Sum and so we can see the Case statement is being summed itself by assignment number. The next function we can see is a Count of Period End Dates and that makes sense because we need to divide the units worked by the number of Payroll Periods the report is running for. We then come to an ELSE which relates to the original Case statement and if we look over the next few rows we can see they are the same as the earlier rows, the only difference being the count of payroll periods is ÷ 12 x 52.1428. This is because the second half of the case statement relates to monthly payrolls which need to be annualised and divided by the number of weeks in the year. We can then come back to the original Sum which ensures we get average hours worked grouped per assignment.

Dynamic Dates (44:55 – 01:11:38)

We are now going to look at some dynamic dates following on from the date functions we looked at earlier. We will cover using dynamic dates within formulas and also filtering using dynamic dates and dynamic date periods. If we set up a basic report the most simple of all dynamic dates is the current_date function and we can see that here. The current date for this recording is 09/08/2021. We are also going to filter the report using current date, this ensures that dynamically, whenever the report is run the data is correct as at the current date. To do this we select add more options in the filter setup up and select SQL expression, then we enter current date as the value. If we add time.date into the report we can prove the filtering is working correctly. This is useful when you want to test the date dimension filtering. Ok so what do we mean when we say dynamic dates, well some of the examples we are going to look at now are first day of previous month, last day of the previous month and last full 12-month period (where we are reporting over a period for things like staff movements or absence reporting). Often the easiest way to do this is to use formulas that have already been written. The NHS Central Team has produced a quick reference guide which is available on the ESR Hub and contains a number of pre-written formulas for dynamic dates. The easiest way to use these

formulas is to copy them straight out of the document and put them into ESRBI and you can see that is being done here. Now if we have a look at this formula you can see it is made up of basic functions, nothing we haven't covered already today. If we test the results you can see that the formula is returning the last day of previous month based on the current date. If we have a look at how this formula is put together, we have an initial timestampadd, an interval which is day, then we have a comma and then we have minus 1 so we are going back one day from whatever comes next. And what comes next is another timestampadd so just as we did with the Case statement we have embedded a timestampadd within another timestampadd. If we ignore the first row and look at the second timestampadd, we have our day interval then we have our integer which is actually day of month function which simply returns the day of the month, in our case 9. If we look at the next part it's -1 which simply turns 9 into -9 then we have plus 1 which gives us -8, so we know the whole row evaluates to -8. If we use -8 within the second timestampadd we can see that we are moving back 8 days from the current date which will always return the first of the month given the current date is the 9th. So we know the lower three rows of this formula will always return the first of the month. If we then use this within the first timestampadd, all we are doing his moving back one day from the first of the month to give us the last day of the previous month. We are using simple functions in interesting ways to provide more complex results. If we now update the time.date filtering to 01/01/2020 we can see that our last day of previous month formula is working independently of the time dimension filter. It is working based on current date. If we now look at adding time.date into the formula our last day of previous month updates in line with our time dimension filter and becomes more dynamic. We simply need to replace any instance of current_date with time.date. If we don't change the time dimension filtering we can see our formula updates in line with the filter. We will now pick another formula from the document, first day of previous month and we will build up to a dynamic period rather than single date. Currently this is based on current date but if we want it to work with our time dimension filtering we replace current_date with time.date and the results return as we require. We will now look at using the last day of previous month formula within a filter. To do that we need to ensure time.date is not included in the formula because we are trying to filter time.date so we set the formula back to current_date. We copy the formula from the column and paste it into the SQL expression box. We now know whenever this report is run it will always be correct as at the last day of the previous month in terms of the effective date. We can test it removing all columns except time.date which shows that the report is filtering correctly. If we now look at filtering for a period we need to set up the filter with an operator Is Between and ensure we add a second SQL expression option, this then becomes the from and the to date. In our example we will leave the from date as last day of previous month and add the to date as current date. The results show we are now returning a date period rather than a single effective date. If we pick one more example and put in last day of current month we can test the results. Finally we are going to look at dynamic dates within dashboard prompts and setting up prompts so there is no reason for our users to change the date, by default have the required date as the default date. We are setting up an example dashboard prompt based on time.date and we will set a simple fixed default date and then a more dynamic

default date. In the example on-screen we have setup a basic prompt with no default value. To set a default value open up the options and select the type of default value we wish to set. We can set a fixed date and test it in the example prompt. If we select reset to default values, we can see the default is displayed. Where we want to set default values for dynamic dates we can select SQL results. We need to enter SELECT then we can add a fixed date if we want to and then we need to add FROM and the subject area in this format. We are entering a fixed date to start with to ensure the SQL works, we can then enter our dynamic date formula. We can test it and we can see it is working as required. The last thing we are going to look at is a dynamic period, we need to set the operator to Is Between and we are presented with two SQL results boxes. These are now our from and to date. We will copy the last day of previous month formula into the to date box and change the from date SQL. We are going to set the period to the last full 12 months so to achieve this we need to change the from date to 364 days prior to the to date. We can simply add another timestampadd function around our to date moving it back 364 days and it becomes our dynamic from date. If we check the default values we can see our prompt will now always return a full 12-month period meaning there is less chance of user error when setting date values.

Filtering based on the results of another analysis (01:11:38 – 01:21:15)

We will now look at filtering based on the results of another analysis and some of the reasons you may use this function. If we set up a basic analysis with a time period of 31 days, we save this analysis and call it the parent. We will now update the date filter to return just the first week of the month. We will save this and call it the child. If we now go into the catalog and open up the parent analysis, we can set up a new filter with an operator of based on the results of another analysis. We need to select the child analysis in this instance, change the operator to ‘is not equal to’ and there is only one field so we can leave that as date. If we test the results we can now see the values from the child analysis are removed or filtered out of the parent analysis. That is a quick run through of how the functionality works, we will now look at an example of how we have used it within in the NHS Standard Dashboards. The NHS Standard Dashboard analysis we are going to look at is the NHS Position Detail report. The problem to be solved with this report was that we are looking to return all positions, and assignment information where it is present. To do that we need to use two different subject areas, the Position Analysis subject area and the Workforce Profile subject area. The Position Analysis subject area returns all positions including vacant positions but returns no assignment information, the Workforce Profile subject area returns populated positions along with assignment information. We use a Union to combine the results but we needed to remove populated positions from the position analysis side of the query because these are returned in the workforce profile side. To do this we created a sub-analysis which is a copy of the workforce profile side of the Union query. We used this to remove any positions from the position analysis side of the query that appeared in the workforce profile side. That way we are left with a distinct list of positions along with assignment information where it is present. Other reasons to use this style of filtering might be because you can't naturally join two subject areas together so to

filter an analysis based on a subject area using the results of another analysis based on a different subject area can be achieved using this method. You may have two analyses returning datasets based on the same subject area but you would like to filter the datasets differently, then use one dataset as a filter for the other dataset, this will be another reason to use the filtering based on the results of another analysis. That is the end of the workshop, we hope you found that useful, thank you and goodbye.