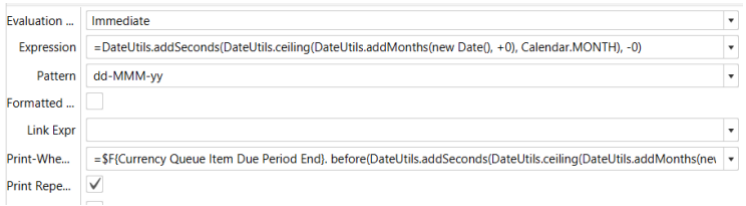
Java Scripts and Report Building Techniques

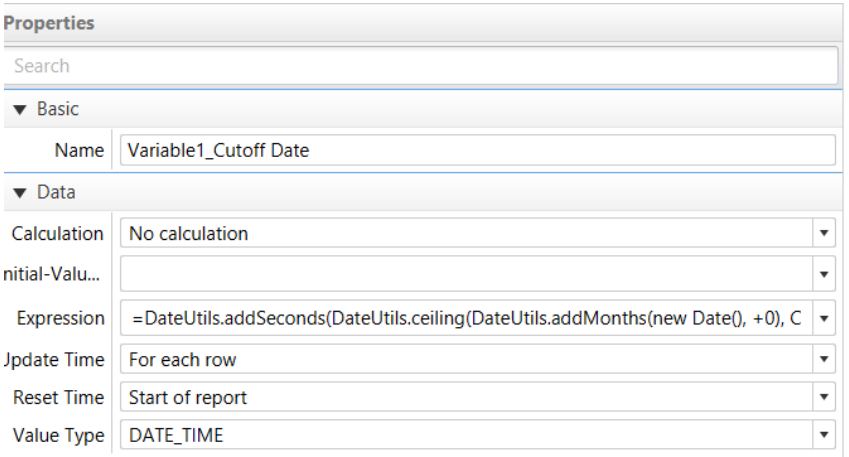
- ***When querying web, look for Jasper Reports vs. Java.***
- ***See Jasper Report Studio Guide in this folder***

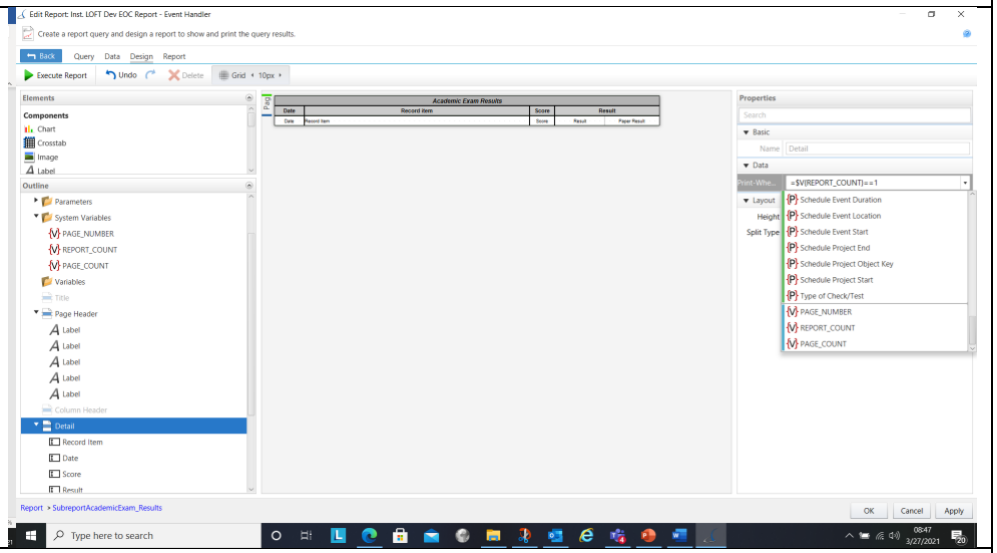
Scripting

What you want to do	How you do it
Java was the first language supported by JasperReports and is still the most commonly-used language as well as being the default. The first thing to note is that each of these expressions represents a Java Object, meaning that the result of each expression is a non-primitive value. The difference between an object and a primitive value makes sense only in Java, but it is very important: a primitive value is a pure value like the number 5 or the Boolean value true. Operations between primitive values have as a result a new primitive value, so the expression: 5+5 results in the primitive value 10. Objects are complex types that can have methods, can be null, and must be "instanced" with the keyword "new" most of the time. In the second example above, for instance (new Boolean (true)), we must wrap the primitive value true in an object that represents it.	Following are some examples of Java expressions: • "This is an expression" • new Boolean(true) • new Integer(3) • (($P{MyParam}.equals("S")) ? "Yes" : "No")
Provide an output value in a report Put this *in the expression* to return Vaccine Card Filed when the issue date for an RI is not null, else No Vaccine Card on File	=$F{Resource Record Item Issue Date} !=null ? "Vaccine Card Filed" : "No Vaccine Card on File"
Java – **Print something when field is empty.** In this case if the field is empty, you print NA Note the If then structure	= $F{Device FAA ID} == null ? "NA": $F{Device FAA ID}
Print when value is not null You can apply this to other fields on the same line and then choose Remove Blank in the properties so the entire line does not print when a particular value occurs or in this case is null	=$F{Currency Queue Item Due Period End} != null

Print when condition is true	=$F{Is Expired}==true
Print if > then	= $F{Device FAA ID} == null ? "NA": $F{Device FAA ID} = $F{Device FAA ID} == null ? $F{Assigned Date}:$F{Device FAA ID} *-brackets denote any query $F{any query term}*
Print when	You can use the formatting above or below to print when either type of case is true. If you select 'Remove Blanks' on the design tab, the blank rows are removed. You can do the same to headers/labels by placing headers inside of a sub report or in the main report with this language. =!$F{Issue Date}. before $V{Prev Month} (See below for how you might set up Prev Month. This was used in the expirations reports to trigger highlighting and in the old TCE Cert Action Obs Report).
Print when query returned date is before or after an end of month date Enter text at right in the Print When dialogue. The query returns the currency queue item due period end. This will be a 1st of the month value (expires at 00:00 on the first day of the month) Use the . before or . after command along with the date function desired. In this case, the java string finds the 1st day of the next month. The Add Months prompt allows you to add as many months as you'd like to this date. Right now it's zero. The add Days allows you to add or subtract days. Thus the first day of any month – 1 day is the last day of the preceding month. Remember quals expire at 00:00 on the first, but in reports we often look at the last day of the month as the 'due date'. In this example, today is 27 March 2021. The scripting finds a new date of 01 April 21 and prints when the currency Queue date is before this date. This is useful for printing items that are coming due – you could add 2 months as an example. See the currency expiration report for use.	=$F{Currency Queue Item Due Period End}. before(DateUtils.addSeconds(DateUtils.ceiling(DateUtils.addMonths(new Date(), +1), Calendar.MONTH), -0)) 
Negating. Adding an exclamation point negates what comes next, so this is really finding when Is Expired is NOT TRUE	=!$F{Is Expired}==true See not null example above. Location of ! may vary depending on whether a calculation is required or not.

Get date Use expression at right	=(new java.util.Date ((new java.util.Date ()) .getYear(), (new java.util.Date ()) .getMonth (), (new java.util.Date()) .getDate ()))
Get Date	mint:now
Get Last day of Last Month Today is 27 March this returns 28 FEB	=DateUtils.addSeconds(DateUtils.ceiling(DateUtils.addMonths(**new Date(), -1)**, Calendar.MONTH), **-1)**
Get Last day of Any Month from now Today is 27 March this returns 31 May The first part gets the first day of the next month. This would be 01 April. The +2 adds 2 months and gives 01 June. Then you Subtract 1 day to get the last day of May	=DateUtils.addSeconds(DateUtils.ceiling(DateUtils.addMonths(**new Date(), +2)**, Calendar.MONTH), **-1)**
Get Last day of Last Month Today is 27 March this returns 28 FEB The -1 removes days, so +1 will get to the first day of this month (last day of last month + 1 day)	=DateUtils.addDays(DateUtils.truncate(mint:now, Calendar.MONTH),-1)
Get first day of XX months from now Here we added months using -2. So on 27 March – 2 months gives me 01 Jan Changing to + 2 gets me 01 May	=DateUtils.addMonths(DateUtils.truncate(mint:now, Calendar.MONTH),-2)
Get first day of this month Today is 27 March this returns 01 Mar	=DateUtils.addSeconds(DateUtils.ceiling(DateUtils.addMonths**(new Date(), -1)**, Calendar.MONTH), **+1)**
Find Last Day of this month First part of this finds the first day of next month. The last item (-1) subtracts 1 day to get the last day of this mnth.	=DateUtils.addSeconds(DateUtils.ceiling(DateUtils.addMonths(new Date(), **-0**), Calendar.MONTH), **-1**)
Get first day of next month Today is 27 March this returns 01 APR	=DateUtils.addSeconds(DateUtils.ceiling(DateUtils.addMonths**(new Date(), -0)**, Calendar.MONTH), **+1**)
Add 1 month to today's date Today is 27 March, this returns 27 April	=(new java.util.Date ((new java.util.Date ()) .getYear(), (new java.util.Date ()) .getMonth () + 1, (new java.util.Date()) .getDate ()))
Trigger highlighting when a date is within a certain rage. Here Currency Que Item Due Period End is the Date and + 60 is the days Place a label under the data you want to highlight. One label can run across the whole page, no need for a different label under each text box.	=$F{Currency Queue Item Due Period End} .before (new java.util.Date ((new java.util.Date ()) .getYear(), (new java.util.Date ()) .getMonth (), (new java.util.Date()) .getDate () +60)) =$P{Cutoff Date}. after(DateUtils.addSeconds(DateUtils.ceiling(DateUtils.addMonths(new Date(), -2), Calendar.MONTH), -0))
Trigger highlighting on an expired date	Add a label of any color behind the text box you want to highlight. Place a label under the data you want to highlight. One label can run across the whole page, no need for a different label under each text box. Then reference the expiration date property of the RI so that the box prints when the date is expired: =$F{Is Expired}==true

Hyperlink to a form You must query for the Form Name and Object Key. They renamed the Form Object Key to 'Form Key' here. This only opens forms that you either submitted or where submitted on you. Checking with Kenny to see how you allow viewing any form. Also only works from Web, does not work from the Portal.	$F{**Form Name**}
Use a variable instead **Variable only prints in the detail band, but it calculates anywhere, so it can be used to drive print functions.**	You can use the java above to create a variable. Choose no calculation and DATE-TIME as the variable type and then you can reference the variable in place of the formulas. **Variable Syntax:** =DateUtils.addSeconds(DateUtils.ceiling(DateUtils.addMonths(new Date(), +0), Calendar.MONTH), -0) **Print When Syntax** **=$F{Currency Queue Item Due Period End}. before$V{Variable1_Cutoff Date}** **Or another example** **=$P{Cutoff Date}. after $V{Variable1_Early Warning}** **You need to set the Reset Time to 'Never' in order to get the variable to print outside of the details section. E.g. in the report header. This can be helpful for setting up an expiration date.** 
Print based on comparisons	In Print When Expression I used something like this. The text is only showing when this condition is fulfilling.

	$V{Dr_total}.intValue()<= $V{Cr_total}.intValue() ? Boolean.TRUE:Boolean.FALSE
Print a value from a list See the screenshot below. There is a built in system variable called report count. You access it form the bottom of the PRINT WHEN dialogue. Set your query so that it sorts the data you want so that you get the desired value in the top row. (Ascending/Descending) Then apply the coding at right so that only 1 row of data prints.	Apply this to the DETAIL Band Print When (Select Detail from pane at left) =$V{REPORT_COUNT}==1
	
Use HTML to Format subset of text in a label	Enter HTML in the expression. Here we will enter text ‘clicking URL opens new browser tab” in a smaller font (4 points less than selected for Label). • Enter to separate the text you want to change: • Form name (*clicking url opens new browser tab)
Using an If-Else Construct in an Expression	(($F{name}.length() > 50) ? (($F{name}.startsWidth(“A”)) ? “AAAA” : “BBB”) : $F{name}) This expression returns the String AAAA when the value of the field name is longer than 50 characters and starts with A, returns BBB if it is longer than 50 characters but does not start with A, and, finally, returns the original field value if neither of these conditions is true.
Turning Dates into Quarters for Grouping Purposes	Set Evaluation as immediate then, In the Expression use the following: ="Q"+((($F{Schedule Event Start}.getMonth()+0)/3)+1)+"-"+new SimpleDateFormat("yy").format($F{Schedule Event Start})

	This generates Q-##-YY as an output You can move the year to the front of the expression so that it sorts correctly in a cross tab YY-QQ
Grouping dates by Month/Year	Create a group of Month and then Group by the expression below. =new SimpleDateFormat("MMM.yy").format($F{Schedule Event Start})

Groovy and Java Code Samples

Table 2-4 Groovy and Java code samples

Expression	Java	Groovy
Field	`$F{field_name}`	`$F{field_name}`
Sum of two double fields	`new Double($F{f1}.doubleValue() + $F{f2}.doubleValue())`	`$F{f1} + $F{f2}`
Comparison of numbers	`new Boolean($F{f}.intValue() == 1)`	`$F{f} == 1`
Comparison of strings	`new Boolean($F{f} != null && $F{f}.equals("test"))`	`$F{f} == "test"`

. Content of the Report (bands)

1. **Title**: This band appears only on the first page of the report.
2. **Page header**: If you want your title to appear on each page, you can use this instead of the title band.
3. **Column header**.
4. **Group header**: This band does not appear by default. You have to add it by right-clicking on “Report” above the bands and selecting “Create Group”. Groups are used to group your data according to the field you choose. You can add as many as you need. Just mind the order! You might want to move them further up or down.
5. **Details**: You can have one or more details band. Here you add the detailed data gathered from your SQL.
6. **Group footer**: Usually used to sum up amounts.
7. **Column footer**.
8. **Page footer**: Usually used to show the page number and date.
9. **Summary**: Mostly used when you want to add an additional page to your report to put some separate data, like a summary.
10. **Background**: Mostly used for document watermarks. For example, if a document is not completed and you want to make this apparent to the person reading it.

Properties

1. Usually we use text fields to add data in the report (see *Palette* > *Basic elements* on the upper right).
2. Select a field from report and take a look at the “Properties” tab on the lower right.
3. There are 6 tabs: Appearance, Borders, Text Field, Inheritance, Hyperlink and Advanced.
4. “Advanced” contains all data from the others so you can make changes directly there.
5. You can change the font type, font size, styles, etc.
6. The field will show an “Expression”. It can be a field name (e.g., $F{field_name}), variable (e.g., $V{var_name}), parameter (e.g., $P{param_name}), or resource (e.g., $R{res_name}). But it can also contain an expression like $P{x}!=null ? $F{x} : $F{y}. This means if the parameter x is set then the field x shall be shown. Otherwise, field y shall be shown.
7. **Print when expression**: Probably the most used. It usually starts with “new Boolean(…)” and is used in case you want to hide a field. If you want to hide an entire line where all the fields are null, then you also have to select “Remove Line When Blank” and “Blank When NULL”.
8. **Pattern**: If you have a sum, you probably want your number to appear in a particular way (e.g., for numbers like 1’000.00 you can use #,##0.00).
9. **Evaluation time**: The default is *Now*, but there might be cases when you want to evaluate the data per group.
10. Things like size and location, you can change them directly from Design.
11. You can also set properties for the report and for bands. For example, you can use **Print when expression** for a band if you only want it to appear in certain cases.

11.

Page sizes in Pixels

Page Type	Dimensions in Pixels	Page Type	Dimensions in Pixels
Letter	612x792	ARCH_E	2592x3456
Note	540x720	ARCH_D	1728x2593
Legal	612x1008	ARCH_C	1296x1728
A0	2380x3368	ARCH_B	864x1296
A1	1684x3368	ARCH_A	648x864
A2	1190x1684	FLSA	612x936
A3	842x1190	FLSE	612x936
A4	595x842	HALFLETTER	396x612
A5	421x595	11X17	792x1224
A6	297x421	LEDGER	1224x792
A7	210x297		
A8	148x210		
A9	105X148		
A10	74X105		
B0	2836x4008		
B1	2004x2836		
B2	1418x2004		
B3	1002x1418		
B4	709x1002		
B5	501x709		