

**Request project specification for this version by:**October 18, 2024

**Submit project for this version by:**November 22, 2024

**Request exam for this version by:**October 11, 2024

**Materials and assessments for the next version available:**October 20, 2024

If completing Continuing Epic Education requirements, visit your [My Certificates](#) page for additional information.

# Introduction

## Introduction to Cogito SQL

Welcome to the Cogito SQL Badge! Cogito SQL allows SQL report writers like you to put your skills to use directly from Hyperspace.

In this course, you will learn the rules for what kinds of queries can be integrated into Workbench templates, and practice following these guidelines to build your own new templates or expand the functionality of existing templates. This will allow you to transform how your users use Reporting Workbench and introduce a brand-new tool into your suite of analytics solutions.

### Contact Information

The table below provides contact information you may need during your cert process:

Contact	Used For
<a href="mailto:CogitoTrainingSubmissions@epic.com">CogitoTrainingSubmissions@epic.com</a>	The majority of your questions should be directed to this email address. This address connects you with our team of Cogito trainers who can answer your questions on anything related to reporting training. All class-related questions should be directed to this email address and not to the Cert Environments or UserWeb addresses listed below. If you aren't sure about whom to contact, use this email address.
<a href="mailto:UserWebAccounts@epic.com">UserWebAccounts@epic.com</a>	Use this email address only if your access specifically to the UserWeb is broken or lost. If you have a question regarding Cogito training contact <a href="mailto:CogitoTrainingSubmissions@epic.com">CogitoTrainingSubmissions@epic.com</a> .
Direct email for your Epic representative (either the implementers or your technical services representative)	For all exam reviews, contact your Epic representative. You can find this information by asking a member of your own project team or by asking <a href="mailto:CogitoTrainingSubmissions@epic.com">CogitoTrainingSubmissions@epic.com</a> .
<a href="mailto:Exams@epic.com">Exams@epic.com</a>	Use this email address to ask questions about requesting, submitting, or taking exams, as well as proctors for those exams.
<a href="mailto:Registrations@epic.com">Registrations@epic.com</a>	Use this email address to ask questions about registering for classes at Epic.
<a href="mailto:TrainingAdminTeamSubmissions@epic.com">TrainingAdminTeamSubmissions@epic.com</a>	Use this email address to inquire about the status of your paper certification or NVT stickers.

## Practice

Reference the following documents available on Galaxy for information on how to log in and use our practice system:

Document	Used For
<a href="#">An Introduction to Epic's Cert Environments</a>	General access information; Hyperspace, Classic, and Text login information; and FAQs.
<a href="#">Cogito's Introduction to Epic's Cert Environments</a>	Cogito-specific login and usage information, specifically practicing in Study SQL Studio.

## Badge Completion

The following are requirements for successful completion of the [Cogito SQL Badge](#):

- One of the following:
  - Caboodle Data Model Certification
  - Clarity Data Model Certification
- COG200v Cogito Tools Administration attendance
  - This is a virtual class spanning two half-days
  - Attending COG200v requires COG170 attendance
- COG2030v Cogito SQL attendance
  - This is a virtual class lasting one half-day
- COG2030v Cogito SQL self-assessment
  - This is an un-proctored exam. Part of the exam will require logging into the system, and you will be answering questions about records in a project environment. The exam has true/false, multiple choice, and multiple select questions. Instructions for beginning your self-assessment can be found in [Appendix A: Cogito SQL Self-Assessment](#).
- COG2030v Cogito SQL Project
  - The project can be completed in a hosted environment at any point after completing the course. Instructions for completing the project can be found in [Appendix B: Cogito SQL Certification Project](#).

## SQL Search Engine

### Introduction

In this lesson, you will explore the possibilities introduced by the Cogito SQL search engine. Workbench templates can be expanded to retrieve data from analytical databases with much broader search bases than ever before. You will learn to explain the features and limitations of these templates.

### How does it work?

Workbench templates using an Ad Hoc search engine are the primary means of retrieving Chronicles data. Tools using the SQL language are the primary means of retrieving data from Clarity and Caboodle. Cogito SQL templates can use both a SQL query and the Ad Hoc search engine in order to retrieve data from both Chronicles and either Clarity or Caboodle.

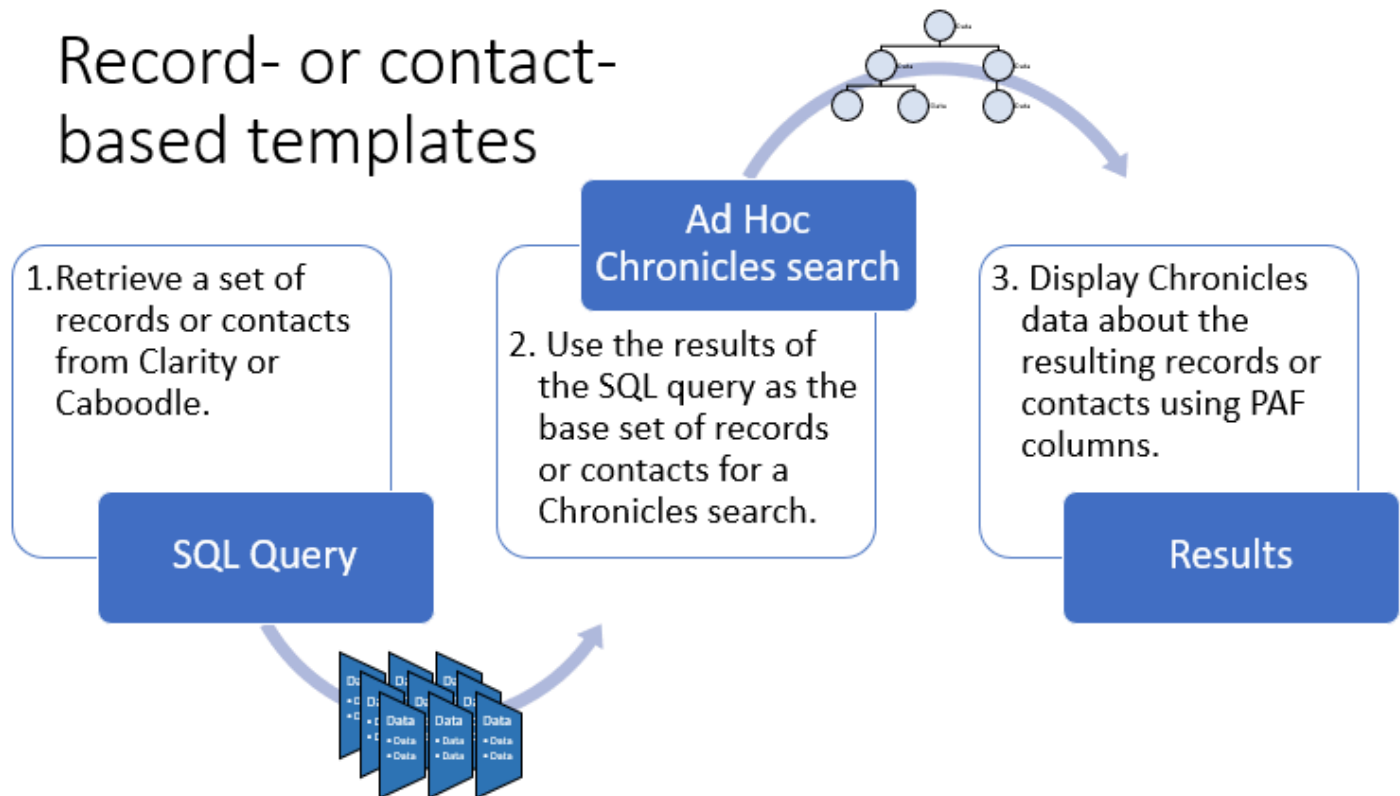
When designing a Cogito SQL template, the first thing to decide is whether or not your results can be expressed as a set of Chronicles records or contacts.

1. If yes, you can build a record- or contact-based template which uses both a SQL query and the Ad Hoc search engine to generate results.
2. If no, you can define the entire search in the Cogito SQL query, but lose the ability to evaluate the data using the Ad Hoc search engine in Chronicles.

A record- or contact-based search will always be preferable because it enables several popular Reporting Workbench features. These templates can:

- Evaluate criteria against current values in real time
- Display real-time data about your results
- Use extension-based PAF columns
- Enable follow-up actions from the results toolbar

Because of this, we will be focusing on record- or contact-based templates initially, and will address other types of template at the end of the chapter.



## The Cogito SQL query

The SQL query you write for a Cogito SQL template is actually stored in Chronicles, in the IDJ master file. When a report built from this template starts its run, the query is sent to the appropriate database and executed.

No matter which user (EMP) is logged in to Hyperspace and running the Cogito SQL report, the same SQL server security credentials are sent to Clarity or Caboodle. Some organizations may usually rely on each running user having different SQL Server security to enforce database access in other tools. That will not be the case using Cogito SQL. Cogito SQL reports will instead rely on user security in Epic to determine who can run which reports and which data to return.

**!** If you want two users to run the same Cogito SQL report and get different results, you will have to do this using dynamic parameters. You cannot rely on SQL Server security settings. For more on dynamic parameters, review [Dynamic Parameters](#) from COG200v.

The results from the SQL query are sent back to the Cogito SQL template and become the basis for the next step of the search. Because the next step is done in Chronicles, these results must uniquely identify a set of Chronicles records or contacts.

We will refer to the results returned by the SQL query as the Chronicles subset.



So far, we have been differentiating between a "Cogito SQL" template and a "Workbench template," but for people who are interested in using precise language, this classification is flawed. Technically, both templates have a type of "Workbench", and the only real difference is the search engine. It may be more precise to say, "A Workbench template using the SQL search engine" and "A Workbench template using an Ad Hoc search engine." But even this is not perfect, because most Cogito SQL templates actually use both in sequence. The most precise (and most cumbersome) way to describe a Cogito SQL template is "A Workbench template using both a SQL query and the Ad Hoc search engine." In this companion, we will refer to the two as a "Cogito SQL" template and an "Ad Hoc only" Workbench template.

## The Ad Hoc search

The Chronicles subset is now used as the basis for an Ad Hoc search. This search uses the exact same framework you used when building a Workbench template. As a reminder, this means:

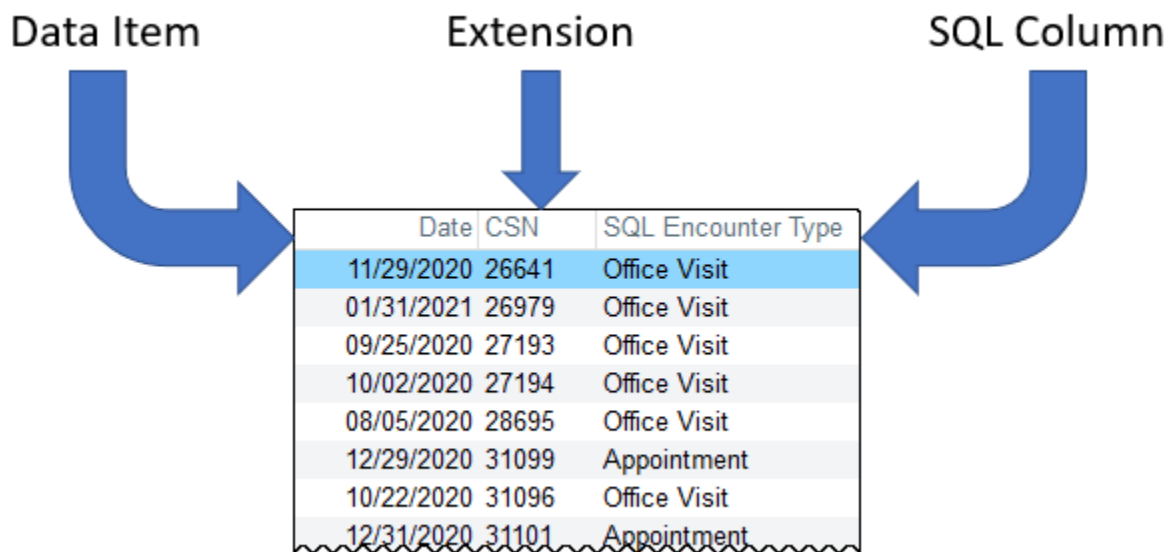
- The template is based on one Chronicles master file.
- Parameters are built in the Template Editor.
- Networked parameters use networked items to search data in other master files.
- Users define criteria using your parameters and the options you make available in the User Interface.
- At runtime, these criteria are evaluated against the values currently stored in Chronicles.

In fact, you may find that designing the Ad Hoc search portion of your Cogito SQL template is the same as designing an "Ad Hoc only" Workbench template. The only significant difference is that your Ad Hoc search can never find any new results that were not in the Chronicles subset.

## Displaying the results

In an Ad Hoc only Workbench template, we use PAF columns to control the data displayed in the report results. In a Cogito SQL template... we do the exact same thing!

The final results of your query use PAF columns to display data to the report consumer. There are three basic ways that a PAF column decides which data to display, based on the **field type** defined in the Column Editor.



*Data Columns use INI + Item # to find data in Chronicles and display it. Extension Columns use LPP ID to run M extensions which return data to display. SQL columns use column aliases to find data in SQL results and display it.*

When building a template, you will most likely use all three types of column. The most important things to remember are:

- 1. Only Data Item and Extension columns display Chronicles data
- 2. Your report authors will be able to find and add Data Item and Extension columns at the report level, but should not be adding SQL columns.
  - A. Since every SQL column is tied to an alias in a SQL query, they cannot be reused across templates.

## In the Template Editor

Cogito SQL templates are built in the **Template Editor** just like Ad Hoc only Reporting Workbench templates. While most of the settings will look familiar, there are a couple of fields that deserve special attention.

Launching Template Editor

Name:

Cogito SQL Template

Internal ID:

Auto-generated

Edit

Contact date:

5/19/2023

Record type:

Workbench

Both Ad Hoc only Workbench templates and Cogito SQL templates have a record type of Workbench.

- 1 Create a new Workbench template called "<your initials> Cogito SQL Encounters"

## Basic Info

The **Basic Info** form serves the same purpose in a Cogito SQL template as in an Ad Hoc only Workbench template.

## Search

In the **Search** form, you make the big decisions regarding the infrastructure of your template.

Query Method	To make a Cogito SQL template, set this field to SQL.
Master File	The search master file stores the base Chronicles INI for your template. Your SQL query must return data identified in this master file. This is most easily found in the dictionary documentation of the Clarity or Caboodle table from which you are returning your list of identifiers.
Search For	<p>For a record- or contact-based template, set to Records or Contacts to match the type of identifier you are returning from your SQL query. A record identifier must be the .1 of a Chronicles record. Contact identifiers may be either a CSN or a two-column key consisting of a record identifier and a Contact Date Real (CDR).</p> <p>To read more about Contact Date Real, refer to the <a href="#">Identifying Chronicles Contacts</a> section from <a href="#">CLR110 Clarity Data Model Fundamentals</a>.</p> <p>If the granularity of your results is not Records or Contacts, you will choose "Defined by Search." For a full list of the rules when using a "Defined by Search" template, refer to the <a href="#">Non-Chronicles SQL Queries</a> chapter.</p>

**Data Model**

*Choose between Clarity or Caboodle. While it is possible to build database connections to other data sources, they are not supported in a Cogito SQL template due to the limitation that all queries must return Chronicles identifiers.*

- 2 Give your Cogito SQL template a **Query Method** of "SQL"
- 3 **Master File** = "EPT"
- 4 **Search For** = "Contacts"
- 5 **Data Model** = "Caboodle"

**Criteria**

Criteria in a Cogito SQL Template are built on the **Criteria** form first. The process for building a criterion is the same for Cogito SQL and Ad Hoc only templates with one exception. Criteria that should be used by the SQL search engine are marked by a checkbox. When creating a new criterion, you will see an option for "SQL Parameter." This is used primarily in Non-Chronicles templates where the values being entered by users do not map to a Chronicles item.

**Handle in SQL Query**

*This box should be selected for every criterion built for evaluation by the SQL query. If a criterion should consider current, real time information from Chronicles, leave this box unchecked.*  
*Criteria handled in the SQL query cannot use an operator list. Only the criterion value is passed into SQL, so allowing operators would be meaningless.*

- 6 Build criteria to search on:
  - 1. Department (I EPT 7070) handled in SQL
  - 2. Provider (I EPT 7040) handled in SQL
  - 3. Encounter Status (I EPT 7020) handled by the Ad Hoc search
- 7 Set the **Appt Department** and **Appt Staff** criteria to **Default**.

**SQL Query**

This form appears on your Template Editor once you have chosen a Search engine of SQL. Here you can paste in a SQL query for use in your Cogito SQL template. While it is possible to write your query entirely in this screen, and we even provide a Script Template to help with formatting, it is not recommended. This text editor does not support any syntax or spell checking like most SQL writing applications. The typical workflow on this screen would be:  
1) Click Edit  
2) Paste your SQL query into the SQL Script field  
3) Replace any parameters with SQL placeholders  
4) Click Stop Editing  
Some other options on this screen include:

**RDBMS Platform**

*If your organization is running Clarity on an Oracle server, select the appropriate radio button.*


**Generate Columns**

*This will automatically create brand new PAF columns for values in your SELECT clause. These columns can only be used in this template.*

**Query Timeout**

*Your system has a default timeout for long running or stalled SQL queries, but this field lets you override that value with a time out limit specific to reports built from this template.*

<b>Parameters</b>	<i>Any criteria you built for use in the SQL query will be listed here. This allows you to easily paste in SQL placeholders for those criteria into your SQL query. You will see the specific syntax this creates in the <a href="#">Special Syntax</a> lesson.</i>
<b>Restore</b>	<i>If you enable your query for editing, you can freely edit the text in the SQL script field. Clicking this button will erase all changes since the last time you enabled the query for editing.</i>
<b>View Dependencies</b>	<i>This link takes you to the BI Dependency Editor. It contains metadata about the IDJ record which stores your query. This includes a list of the tables and columns your query retrieves data from. This is primarily used at runtime when we check to ensure all the required database objects are available on the server.</i>

 The text editor for SQL queries supports up to 200,000 lines of code.

**8** Open the Encounters Query from Exercise Answer Keys and paste it into SQL Script.

**9** Click **Generate Columns** and generate both columns.

We will add parameters to resolve any errors in the next chapter, so leave the parameters “hard coded” for now.

## Generate Columns

When generating columns from your SQL Query form, the PAF columns you create will usually be perfect copies of the SQL columns in your SELECT clause, but you have a couple of options for customizing them.

- 1) You can change the displayed column title if your SQL aliases are not user-friendly.
- 2) For datetime columns, you can choose to only display the date or time, or keep it as an “instant” datatype.
- 3) You can choose to create columns for some or all of the SQL columns from your SELECT clause.
- 4) Once you click to generate columns, you can choose a record ID to start from, in case you wish to control the ID ranges for PAF creation.

## Get Resolved Query

If you were to copy/paste the SQL query from a Cogito SQL template into a SQL editor, it would not run. This is because of some of the special placeholders and syntax you will learn about in the [Special Syntax](#) chapter. The **Get Resolved Query** button generates a preview of what your actual SQL query will look like once users have entered their parameters and run a report based on this template. This code can be copy/pasted into any SQL editor for you to run and troubleshoot.

## All the other forms

Everything else on the Template Editor works the same for Cogito SQL templates as it did for Ad Hoc only templates.

**10** On the **Columns** form, add columns for MRN (EPT) [367], Patient Name [1004], Appt Provider ID [1745], and DEPARTMENT/LOCATION [1031]

**11** Set all columns to Required

**12** Preview and run the report

- Dynamic Criteria
  - Dynamic criteria function the same in Cogito SQL as they do in Ad Hoc only Workbench templates.
- Viewer
  - The options in the viewer are identical to those of an Ad Hoc only Workbench template.
  - Cogito SQL templates should always show Static **Column Data** to avoid performance issues. In this case, the SQL query won't have any record limits, and the Ad Hoc search will respect the normal search and return limits specified in Hyperspace.



- In a Cogito SQL template with Dynamic **Column Data**, the SQL query is limited to only returning 10,000 rows to the Chronicles subset. This means that even if your **Search max** and **Return max** store higher values, 10,000 would be the functional maximum number of results.
- Columns
  - A Cogito SQL template uses PAF columns to display data. Data Item and Extension columns are based on the results of an Ad Hoc search, just like an Ad Hoc only Workbench template. They follow all the same rules regarding base and non-base master files.
- Actions
  - Record- and Contact-based Cogito SQL templates DO support actions in the same way as Ad Hoc only Workbench templates.
  - "Defined by Search" templates will NOT support actions.
- Access
  - Access functions the same in Cogito SQL templates as it does in Ad Hoc only Workbench templates.

## Reviewing the Chapter

### Review Questions

---

1. You run a report from a Cogito SQL template with a max # of records to return of 20,000. You can't seem to return more than 10,000 rows in the results. What can you change on the template to allow this report to return more than 10,000 rows?
  - A. Change the Data model
  - B. Change the Search engine
  - C. Change the Column Data to "Static"
2. You are designing a template for use by your clinicians. The template is meant to find patient encounters in the last 10 years with a visit diagnosis of pneumonia. Each clinician should only see visits where they were the visit provider. Which if the following correctly identifies where to build your criteria?
  - A. Visit Diagnosis on the Criteria form and "handled in the SQL Query," Visit Provider in the SQL query only
  - B. Both criteria in the SQL query only
  - C. Visit Diagnosis in the SQL query only, Visit Provider on the Criteria form and the Dynamic Criteria form
  - D. Both criteria on the Criteria form and "Handled in the SQL Query," Visit Provider also in the Dynamic Criteria form
3. Describe the order of operations in a record-based Cogito SQL report. For each step, list whether the data is real-time or historical (not including today's data).

### Study Checklist

Make sure you can define the following key terms:

Cogito SQL template

Query Method

CSN

CDR

Record ID

Chronicles subset

"Defined by Search" template



Make sure you can perform the following tasks:

Create a blank Cogito SQL template

Paste a SQL query into the template

Identify which criteria will be handled in the SQL query

Generate SQL-based PAF columns

Make sure you fully understand and can explain the following concepts:

A Cogito SQL template uses both a SQL query and an Ad Hoc search engine

The Ad Hoc search engine uses Chronicles criteria and PAF columns to search and display real-time data

The difference between historical data and real-time data

## Special Syntax

### Introduction

In this lesson, you will write SQL code for use in a Cogito SQL template. You will practice in both Clarity and Caboodle to find appropriate record and contact identifiers; and apply Workbench date ranges to your SQL queries. With these skills, you will be able to take an existing SQL query and convert it into a Cogito SQL template.

### Record and Contact Identifiers

Templates that search on records or contacts must have a SELECT clause that identifies the Chronicles records or contacts. That means it must include one of the following:

- A single column identifying a record
- A single column identifying a contact by its CSN
- Two columns identifying a contact by its record ID and CDR

It is your job to make sure you use the proper database columns to return these specific data points from either Clarity or Caboodle.

You will never use a datetime column in either database to identify a contact. Even if you are confident that each contact has a unique date in your data set, Chronicles will not recognize a datetime as a unique identifier. It is not possible to generate a Chronicles subset using a datetime column.

### Caboodle

Caboodle stores both Epic and non-Epic data. Because of this, the Chronicles identifier is never the primary key of a table. Instead, you must find a column which specifically stores an EpicCsn or an EpicID.

Address for the patient's residence
▶ <b>PatientEpicId</b> : string Epic ID of the patient. This column identifies patient (EPT) records.
▶ <b>Name</b> : string

*PatientDim.PatientEpicId identifies a Chronicles record in the EPT master file.*

Type of service provided for the encounter derived from encounter data. For Epic claims derived data, this can be Short-te...
▶ <b>EncounterEpicCsn</b> : number Epic contact serial number (CSN) of the encounter. This column identifies patient (EPT) contacts.
▶ <b>DerivedOutpatientVisitType</b> : string

*EncounterFact.EncounterEpicCsn identifies a Chronicles contact in the EPT master file.*

Caboodle doesn't store CDRs as contact identifiers. This means for certain reports, you will have to use Clarity as your data source for a Cogito SQL template.



### Write in Workbook

Medication dispenses are contacts in the ORD master file. These contacts can only be identified by CDRs in Chronicles. To write a Cogito SQL template that returns a list of medication dispenses, you must use \_\_\_\_\_.

You could not complete this report using \_\_\_\_\_.

## Clarity

Clarity contains only Epic data, and Clarity tables often use a Chronicles identifier as the primary key.

Column Information			
	Name	INI	Item
1	PAT_ID	EPT	.1

*PATIENT.PAT\_ID extracts EPT .1 which is a Chronicles record identifier.*

In Clarity, you may find tables which extract both a CSN and a CDR. In this case, you could use either in your query, but using CSN will make your query slightly easier to write.

#### 1 Use Database Object Search to open the table ALT\_HISTORY

This table obviously has a record identifier: ALT\_ID which extracts ALT .1. But how can you tell if a given column stores a CSN or a CDR? For this you often have to rely on naming conventions.

CDR columns will often contain the string **DATE\_REAL** or the abbreviation **CDR**. They will also always be stored with a data type of **float**. Sometimes the dictionary will list them as extracting an item called the DTE. Other times they will show as extracting no item at all.

CSN columns will often contain the string **CSN** or **SERIAL\_NUM**. CSN can be stored with a different item number in different master files. If you are not confident that a column in the dictionary is definitely the CSN, use **Item Editor** to look up the Chronicles item it extracts and read the item description.



### Write in Workbook

To write a SQL query which returns a list of contacts from ALT\_HISTORY, you could SELECT two columns:

\_\_\_\_\_

and

\_\_\_\_\_

or a single column:

## Multiple response data in Clarity

Many Clarity tables extract multiple response and related group items from Chronicles. These items are stored in tables with a LINE column in the primary key. At their core, these tables are about Chronicles records or Chronicles contacts, so Cogito SQL templates can incorporate the line details about those records or contacts.

OR_LOG_ALL_STAFF					
Type:	Extracted Table	Load Type:	REQ	Deprecated?	No
Chronicles INI:	ORL	Load Frequency:	INCREMENTAL	Extracted?	Yes
Release Version:	Rel 2014	Filename:	OR_LOG_ALL_STAFF	Preserve Record?	No
Description:	The OR_LOG_ALL_STAFF table contains information about all staff members associated with a procedural case that has been performed. This includes physicians, procedural staff, anesthesia staff, pre-op nurses, and recovery nurses.				
Primary Key					
Column Name	Ordinal Position				
LOG_ID	1				
LINE	2				

OR\_LOG\_STAFF has one row per staff member on a surgical log. A report listing each staff member would need to identify each LINE on the record as a separate row in the results.

A Cogito SQL template can accept line-level results in the Chronicles subset. In this case, each line of data will get its own row in the results, with the following considerations for your display columns:

- For columns displaying single response Chronicles data, the data will be repeated on each row in the results.
  - For columns displaying multiple response or related group Chronicles data, the PAF will respect the settings defined in the Column Editor in the Lines field.
    - (recommended) Matching lines only means that the column will display the value whose line matches the value in the LINE column of your SQL results. One value will display for each row in the results.
    - All lines means that every value in the item will be displayed for each row in the results. This would lead to a lot of repetition and confusion.
    - Last line means the column will display the value stored in the last line of the item. (The highest LINE value, not the most recently updated)
- 2 From the Analytics Catalog, run the COG2030v Line-Level Columns report.

## Aliases

In your SQL query, each column used as an identifier in your SELECT clause must have an alias. These aliases are all case sensitive and must be exactly as written here.

- Record IDs must have an alias of ID
- CSNs must have an alias of CSN
- CDRs must have an alias of DATE
  - Note, that even though the alias is DATE, this is not a datetime column. It is a float.
- LINE columns must have an alias of LINE.



If you use any of these reserved aliases on other SELECT columns, your report will have unexpected results. Consider ID, CSN, DATE, and LINE to be reserved aliases in a Cogito SQL query.

## In-Class Activity: Aliasing

Based on the name of each of the following Clarity or Caboodle columns, how would you alias it in a Cogito SQL query?

Column Name	Alias
VisitEpicCSN	
TX_ID	
ORDER_DATE_REAL	

Column Name	Alias
NDC_CSN_ID	
MedicationEpicId	
GuarantorEpicId	
ALT_DATE_REAL	
CONT_DAT_REAL	
PAT_ENC_CSN_ID	
CNCT_SERIAL_NUM	
ACCOUNT_ID	
LINE	

## SQL Parameters

Cogito SQL templates allow for a great deal of flexibility by accepting user-entered values for the SQL query. Passing these user-entered values into the SQL query at runtime is fairly simple to set up in the **Template Editor**. Each SQL Parameter is mapped to one of your template's criteria. The values entered in those criteria on the HRX are pasted into your SQL code wherever you have the mapped SQL parameter.

- 3 Open the SQL Query of your <your initials> Cogito SQL template for editing.
- 4 Highlight the Department ID in your WHERE clause.
- 5 Select the **Appt Department** parameter and click **Insert Placeholder** to copy the SQL placeholder into your query.
- 6 Highlight the Provider ID in your WHERE clause.
- 7 Select the **Appt Staff** parameter and click **Insert Placeholder** again.
- 8 Click **Stop Editing** to save your changes.
- 9 Preview the report.
- 10 Add an **Appt Department** of "EMC Family Medicine" and an **Appt Staff** of "Rob Marino".

However, there are some rules to follow when building SQL parameters to ensure that your end users are happy with the results and don't find themselves facing an unfamiliar error screen.

## I Want to Search on Multiple Values

The equals (=) operator in SQL can only compare one value to one value. When a user enters multiple values into a Workbench Report, they are pasted into the SQL query as a comma-delimited list.

### Appt department

Appt department	
1	EMC FAMILY MEDICINE [10501101]
2	EMH MED SURG [10101102]
3	EMC PHYSICAL THERAPY [10501140]
4	<input type="text"/>

Criterion Logic
OR

These values would be passed to the SQL query as 10501101,10101102,10501140

In order to handle this, your SQL syntax would have to use the IN operator instead of equals (=) and place parenthesis on either side of the SQL placeholder.

```
PATIENT.ZIP in ({{C_ZIP_CODE}})
```

- 11 Edit the SQL query in your template to use IN operators instead of equals
- 12 Add parentheses around the SQL placeholders
- 13 Click Preview
- 14 Run a report for multiple departments and providers

Now your template can handle multiple values from the user. Not every criterion will allow multiple values, but if a user should ever be able to enter multiple values into a criterion, your SQL query should use this syntax to handle that eventuality.

## I Don't Want to Use that Criterion

What happens if someone builds a report from your template and decides to leave one of the criteria blank, or remove it from the Criteria list? The SQL query will evaluate the report as though the SQL placeholder were a NULL. This usually means that no results will be returned, because the equals (=) and IN operators do not handle NULLs well. There are two ways to handle this dilemma:

### Make the criterion required

This is done on the Criteria form. This option ensures that a NULL cannot be passed to your SQL query. However, it also means your template is less flexible.

- 15 Make your **Appt Staff** criterion required
- 16 Click Preview

Since a user can neither remove a required criterion nor leave it blank, you can be sure your query will work as intended.

### Add NULL handling to your SQL query

This requires some savvy SQL skills, but can allow your template to handle NULL values in any way you want. The most typical case would be to treat a NULL as "all." This means that if a user doesn't enter a value for a given criterion, the report should not filter on that parameter in the SQL query.

- 17 Edit the SQL query in your template
- 18 In the WHERE clause, add the expression "OR NULLIF(COALESCE({{C\_APPT\_DEPARTMENT}},N''),N'') IS NULL" to your department filter



### Beyond the Basics

This code is fairly complex. To break it down, you have to understand a couple of functions.

CAST changes the datatype of a given value to another.

COALESCE looks at a list of values and returns the first non-null value.

NULLIF compares 2 values, and returns NULL if they are the same.

All together, this code checks to see if there are any values in the parameter. If yes, then NULLIF will not return NULL. The value 'N' was chosen because it can never be a valid entry for a criterion. However, if the user entered no values, then the COALESCE returns 'N', and the NULLIF returns NULL, so the entire expression returns a value of 'true' and the OR logic means your filter will be ignored in the larger query.



If the query runs against Clarity on an Oracle server, replace both instances of 'N' with TO\_CHAR('').

### 19 Run the report again without a department filter

Epic also has a property which may be used instead of the above SQL code.



This syntax will only work in cases where your parameter uses the IN operator, and should not be used for any expressions that rely on inequalities, between, or other complex logic.

```
{{IN_PROPERTY_OPT [[table.COLUMN]] [[PARAMETER_NAME]]}}
```

If the user enters a value in the criterion, this code will compare the value in table.COLUMN to the list of values entered.

If the user enters no value in this criterion, then at runtime it will simple change into

```
1=1
```

Which always returns true.

For more elaborate logic, you may have to build CASE statements into your query which check the placeholder for a NULL to determine how the query should execute.

## I Want to Use a Different Operator

Criteria are built assuming that your query uses an equals (=) or IN operator. There is no way for a user to select an operator like "Less Than" (<) or "Contains" when building a report from this template, because there is no way for the template to pass that operator into the SQL query. The operators in the SQL query are not affected by the template in any way.

If you need to build a criterion which allows these other operators (usually a string, numeric, or date parameter) you should build that criterion as an Ad Hoc only criterion, and you should not add it to the SQL query.



There may be creative ways to get around the need for complex operators. For example, instead of using a "between" operator on a single "Admission Date" criterion, you could build an "Admission date start" and "Admission date end" criteria and use an equals (=) operator for both of them.

## I Want the Parameter Hard-Coded in the Query

While it is certainly possible to just enter a filter into the SQL query itself, and never even show it to the report builders in Hyperspace, it is recommended that even for hard-coded filters, you build a criterion in the template as usual. You

can lock down such a criterion in the **Criteria** form in the **Template Editor**.

The settings that must be in place to enforce a hard-coded parameter are as follows:

1. The parameter must have a default value or values in the **Criteria** form.
2. The parameter must be either **Hidden** or **Required**.

**!** It is certainly possible to hard code your query by just filtering in the SQL query and leaving out the parameter build in the **Template Editor**. However, building the parameter in both places means you will have much greater flexibility in modifying the template in the future.

## Category parameters in Caboodle

Caboodle usually stores the display names for categories instead of their numeric code. For example, in Clarity the TRANSACTION\_TYPE\_C column may store the number 1, but in Caboodle the TransactionType column would store the word Charge.

Cogito SQL templates anticipate this, so when you create a category criterion for user entry, and the user-entered value is copied into your SQL placeholder at runtime, the template will check which database is being used.

- If Clarity, the parameter passes in the numeric codes for the chosen categories.
- If Caboodle, the parameter passes in the display names for the chosen categories.

This means that your SQL query should compare parameters to \*\_C columns in Clarity, but you do not need to find numeric columns in Caboodle to evaluate your category parameters.

## Date Ranges

Almost every report will somehow filter on a date range. In a Cogito SQL template, you can either build a hard-coded date range in the SQL query which will always return data from the same range of dates, or you can allow the report-level (HRX) date range to apply to your SQL query at run time.

```
SELECT
PAT_ENC_CSN_ID      "CSN"

FROM
V_SCHED_APPT      appt

WHERE
appt.APPT_STATUS_NAME = 'cancelled'
and
appt.APPT_MADE_DATE < ({{REPORT_END_DT}} + 1)
and
appt.APPT_MADE_DATE >= {{REPORT_START_DT}}
```

**Find Criteria** Enter a search term, or click the search icon to browse available criteria

🕒 **Date Range** From: M-1 (2/4/2024) To: T (3/4/2024)

**!** In order to use these placeholders, your template MUST have a date range. On the **Search** form of the **Template Editor**, you may not use a "Search all contacts (date option not available)" setting for your date range.



**20** On the **Search** form, allow users to “Edit date range”

**21** Go to the **SQL Query** form and look at the parameters list

The placeholders `{{REPORT_START_DT}}` and `{{REPORT_END_DT}}` in the query of your Cogito SQL template will pull the datetime values from your report's date range at the time a user runs a report.

## Inclusive date ranges

In Ad hoc only Workbench templates, the date range is always inclusive, meaning data from both the start and end date are included in the results. To make sure your Cogito SQL templates work the same way, follow these two rules:

1. To make sure that an event occurred after the start of the report's date range, the date column should be `>=` the `{{REPORT_START_DT}}` placeholder.
2. To make sure to include events that occurred up until (and including) the report's end date, the date column should be `< ({{REPORT_END_DT}} + 1)`

Because SQL assumes all dates occur at exactly midnight, finding all datetimes `< ({{REPORT_END_DT}}+1)` will find everything up to, but not including, midnight the day after the report's end date.



This syntax is preferable to several other options because:

- The syntax works in both SQL Server and Oracle databases
- It works the same for date and datetime columns
- It doesn't transform any date columns using a function, thereby preserving the efficiency gain when filtering on indexed columns.

## Caboodle dates

Remember that in Caboodle, many dates are referenced by DateKey columns. Do not directly compare a DateKey column to a date range placeholder. Instead, you should join each DateKey to DateDim and use the DateValue column.

```
SELECT
ord.MedicationOrderEpicId      "ID"

FROM
MedicationOrderFact ord
  join DateDim start on
    ord.StartDateKey = start.DateKey

WHERE
start.DateValue >= {{REPORT_START_DT}}
and
start.DateValue < ({{REPORT_END_DT}} + 1)
```

Use `DateDim.DateValue` to compare Caboodle dates to the report's date range.



In this class, we will only be using the date range as a filter in the WHERE clause, but these date range placeholders can be used anywhere in a SQL query.

If you wish to use the report date range in a subquery or a join, they will function the same as they do in the WHERE clause.

**22** Edit your template's SQL query to join to DateDim on `enc.DateKey`

**23** Add date filters on DateValue using `{{REPORT_START_DT}}` and `{{REPORT_END_DT}}`

**24** Click **Preview** to see the date range

## Exercise 1: Writing a Query for a Cogito SQL Template

In this exercise, you will open an existing, functioning SQL query about patients with penicillin allergies. You will convert this into a Cogito SQL template to allow end-users to run the report on demand.

1. Open Study SQL Studio.
2. Open the file "Exercise 3-1 Allergies Start" from the COG2030v folder.
3. Save a copy of this file with your initials in the title. This will ensure that if you make any mistakes, you can always start over from the base query.
4. Run the query.

## SQL Query

This query currently returns a list of all penicillin allergies documented in Clarity. To model this data in Cogito SQL, you will have to first find your Chronicles identifier.

1. Consider the columns in the **SELECT** clause. Which of them is the Chronicles identifier for a single allergy?
2. What should this column's alias be?
3. Edit your **SELECT** clause to correct this alias.
4. Highlight the entire query and use **CTRL+C** to copy the text.
5. Log in to Hyperspace as Lorena, your Business Intelligence Developer.
6. Create a new Workbench template called "<your initials> Allergies" in **Template Editor**.
7. On the **Search** form, modify your template so that it:
  - A. Returns a list of LPL records. (LPL is the master file which stores all patient allergies)
  - B. Uses a SQL query based on Clarity data to generate the Chronicles subset
8. On the **SQL Query** form, enable your **SQL Script** for editing.
9. Use **CTRL+V** to paste your query into the **SQL Script** field.
10. Delete "Use CLARITY\_MAY" from the top of your query.
11. **Stop Editing** the script to save your query.
12. Click **Generate Columns**.
13. Flag the columns for PAF creation.
14. As the "ID" column is ambiguous, change the name of the PAF column to be "Allergy ID".
15. Change the **Data Type** of your Date Noted column to "Date".
16. Click "Generate 7 Column(s)".
17. Accept the system-generated ID numbering.
18. On the **Columns** form, make all the columns **Default**.
19. Click **Preview**.
20. Run your report.
21. Compare the results of your Cogito SQL report to the results of your SQL query in SSMS to validate your work.
  - A. You may get different numbers because our training environment is not kept in perfect sync with our Clarity database.
  - B. If you want to find specific rows to spot check, try sorting the Workbench results by Date Noted, and add an "ORDER BY [Date Noted]" clause to your SQL query in SQL Studio.

## NULL handling

1. Some of your allergies do not have a severity noted. To avoid blank space on your report, you can add some SQL NULL handling to the query. Replace your LPL.ALLERGY\_SEVERITY\_C column in the SELECT clause with the following code:
2. COALESCE(CAST(LPL.ALLERGY\_SEVERITY\_C as varchar), '\*No severity noted')



If you copy/paste, the single quotes may be replaced with an invalid character. Try manually typing in the single quotes.

3. Make sure you didn't delete the alias for your severity column when you added NULL handling.
4. Now that your SQL column is a string, you'll have to update your PAF column as well.
5. Go to the **Columns** form.
6. Double click the SEVERITY CATEGORY column to edit it.
7. Change your column's **Data type** to String.
8. Click **Accept**.
9. Preview and run your report again to see the NULL handling in action.

## Date Ranges

1. To allow your report authors to enter a date range, go to the **Search** form.
2. Change the **Date Range Option** to "Edit date range".
3. Modify your SQL query so that the date range in reports built from your template applies to the DATE\_NOTED field in your SQL query.
  - A. Hint: This will involve using the special SQL placeholders covered in the [Date Ranges](#) section.

## Criteria

1. Add a criterion to your template allowing it to search "I LPL 3000".
2. This item stores the allergen ID, but the parameter's default display title is Allergy Name. You can edit the name if you would like it to be more accurate. Change the caption and Internal Name of your criterion to Allergen ID.
3. Mark this parameter as "Handle in SQL Query".
4. On the **SQL Query** form, edit the query.
5. Since your new parameter filters on ID instead of name, you can compare it to lpl.ALLERGEN\_ID. Add an expression to your WHERE clause that finds allergies with an ALLERGEN\_ID in the list of values entered by your end user.
  - A. HINT: Use the Insert Placeholder button to get the right syntax and remember to put parentheses around the placeholder.
6. Remove the **ALLERGEN\_NAME = 'Penicillins'** expression from your query's WHERE clause.
7. **Stop Editing** your query to save your changes.
8. Click **Preview**.
9. Change the date range to "1/1/2019 "to "12/31/2020".
10. Add the **Allergen ID** criterion.
11. Search for allergies to "Penicillins [25]"(the drug class)
12. To validate your report, go back to SSMS to run the report and compare the results.
13. If you have time:
  - A. Try some different allergens and date ranges to test your parameter.
  - B. Does your report run if you filter on multiple allergy names?
  - C. What if you don't filter on allergy name at all?
  - D. Try adding NULL handling to your template or hard-coding your filter.

# Reviewing the Chapter

## Review Questions

1. You are troubleshooting a Cogito SQL query which isn't returning results. The date range filter is using the following placeholder {{REPORT\_START\_DATE}}.  
You're pretty sure that part of the syntax is wrong, but which part(s)?

- A. REPORT should be RPT
- B. START should be ST
- C. DATE should be DT
- D. the curly braces {{...}} should be square brackets [...]

---

2. In a Cogito SQL query, you find a column in the SELECT clause with an alias of DATE. The query is working correctly.

What must be true about this column?

- A. It stores a Chronicles Contact Date Real
  - B. It has a datatype of datetime
  - C. The query must be in Caboodle
  - D. It had a datatype of float
  - E. There is at least one more column in the SELECT clause
- 

## Study Checklist

Make sure you can define the following key terms:

Record identifier

Contact identifier

CSN

CDR

SQL placeholder

Cogito SQL parameter

Equals (=) operator

In operator

{{REPORT\_START\_DT}}

{{REPORT\_END\_DT}}

Make sure you can perform the following tasks:

Correctly alias record and contact identifiers

Insert SQL placeholders into a Cogito SQL template's query

Build a Cogito SQL parameter

Handle multiple values in a Cogito SQL parameter

Handle NULLs in a Cogito SQL parameter

Hard code a Cogito SQL parameter

Filter a Cogito SQL query using a report's date range

Make sure you fully understand and can explain the following concepts:

The SELECT clause of your SQL query will always contain one or two identifiers

## Non-Chronicles SQL Queries

There may be times where a record- or contact-based search just won't cut it. You may have 3rd party data in Caboodle that doesn't have Chronicles identifiers. Some derived tables or views may not have the right granularity. There are even specific Clarity and Caboodle tables in some data models that don't use Chronicles identifiers at all. In these cases, you can still use Cogito SQL, with a few requirements and restrictions.

## Requirements

In the Search form, set the Search For field to Defined by Search.

Any criteria you build will follow these specific steps:

1. Create new criterion
2. Choose a **Criterion type** of SQL Parameter

3. Manually change the data type of the criterion to meet the needs of your query
4. If the criterion calls for a user to pick a category value or a record ID from Chronicles, set a prompt master file and prompt item.



If the criterion calls for a user to pick a value from a custom list that you want to design, the build is more complex, and beyond the basics of this course. You can read more on how to create your own custom lists here: [Add a Custom List Parameter to a Cogito SQL Template](#).

## Restrictions

The results of a "Defined by Search" template are not defined by Chronicles identifiers, which means a number of features of other Cogito SQL templates will simply not function.

1. The results of the SQL query cannot be evaluated by the Ad Hoc search engine, so all parameters MUST be handled by the SQL query
2. All PAF columns must be SQL columns. This means you're not seeing "real time" information about the results.
3. No detailed view is available for more row-level information in the Results Viewer.
4. No toolbar actions will function in the Results Viewer.

## In-Class Activity: Build a "Defined by Search" Template



In Clarity, there is a table called CLARITY\_TDL\_TRAN used frequently for financial reporting. One ETR record could generate many rows in this table, each of which is called a "transaction detail." But these details do not have a unique Chronicles identifier, so they cannot be returned by a record- or contact-based template.

- Create a new template called <Your initials> Transaction Details
- Basic Info
  - Return Max = 1000000
- Search
  - Query Method = SQL
  - Search For = Defined by Search
  - Data Model = Clarity
  - Search Source Name: Transaction Details
  - Date Range Option: Edit date range
  - Default Date & Time = Y-1 to T
- Criteria
  - Create 2 SQL criteria.
    - Current Payer
      - Record Select
      - EPM .1
    - Transaction Type
      - Category Select
      - ETR 50
- SQL Query
  - Open the Transaction Details query from the Exercise Answer Keys folder
  - Enable the SQL Script for editing
  - Paste in the Transaction Details query
  - Stop editing
- Columns

- Generate 5 SQL Columns that use the same names and aliases as defined in the SQL query
- Set all columns to default
- Preview and run your report
  - If you wish to test your parameters, try the following:
    - Transaction Type = Charge [1]
    - Current Payer = Cigna [100009]

## Reviewing the Chapter

### Review Questions

---

1. When would you build a "defined by search" Cogito SQL template?
    - A. Any time you don't need real-time data
    - B. Only when retrieving data from a non-Clarity or non-Caboodle database
    - C. When your results can't be expressed as a set of Chronicles records or contacts
- 
2. What features of a Workbench template can you still use with a "defined by search" Cogito SQL template?
    - A. Users can still use actions from the results viewer to act on results.
    - B. You can still use epic-released PAF columns to display data about results.
    - C. You can still add tags and report groups to your template to allow end users to find and use reports built from the template in their Analytics Catalog
    - D. You can still audit usage of the report using the same HRX and HRN based templates used to report on Ad Hoc only Workbench templates
- 

### Study Checklist

Make sure you can define the following key terms:

"Defined by Search" template

SQL criterion

Make sure you can perform the following tasks:

Create a "Defined by Search" template

Configure useful parameters on a "Defined by Search" template

Make sure you fully understand and can explain the following concepts:

The requirements and restrictions when reporting on non-Chronicles data

## Practice Scenarios

### Scenario 1: Clarity

Build a Cogito SQL template according to the following specifications. For each of the items in your query, we have provided both the Chronicles location and the Clarity location of the data. All you have to do is build the correct parameters and/or PAF columns in your template, and then assemble your SQL query out of the given joins and SQL columns to appropriately satisfy the specifications.

### Report Request

Build a Cogito SQL template that will be used to generate lists of implants. Implants are things placed inside a patient during surgery, and each implant is an IMP record in Chronicles. The requirements for this template are as follows.

1. Name your template <your initials> Clarity Implants Lab
2. The template should use the Clarity database.
3. Users should be able to search by implant type when building a report from the template.
  - A. By default, new reports built from this template should search for implants of the types:
    - i. Screw
    - ii. Mesh
    - iii. Implantable Wire
    - iv. Prosthetic Valve
4. Users should also be able to filter the report by the implant area (which part of the body the implant was placed into).
  - A. By default, new reports built from this template should search for implants in:
    - i. Ankle
    - ii. Heart
    - iii. Abdomen
5. The date range on the report should apply to the implanted date.
  - A. Use the date range 1/1/2017 - 12/31/2020 to test your report.
6. Reports built from this template should always display the following PAF columns:
  - A. Implant ID
    - a. use a Chronicles-based PAF
  - B. Implant type
  - C. Implanted date
  - D. Implant area

## Data points

In the real world, you would need to spend a lot of time finding all of the data points required by this report in Chronicles and Clarity before you begin. We have done that research for you. For every item needed by this report, we have located the data in both databases. It is up to you to decide when you need the Chronicles location or the Clarity location for this template, and to build your query and template appropriately.

Implant ID

in Chronicles	in Clarity
IMP .1	OR_IMP.IMPLANT_ID

Implant type

in Chronicles	in Clarity
IMP 70	Join to ZC_OR_IMPLANT_TYPE on OR_IMP.IMPLANT_TYPE_C = ZC_OR_IMPLANT_TYPE.IMPLANT_TYPE_C Filter on OR_IMP.IMPLANT_TYPE_C Display ZC_OR_IMPLANT_TYPE.TITLE

Implant area



in Chronicles	in Clarity
IMP 500	Join to ZC_IMPLANT_AREA on OR_IMP.IMPLANT_AREA_C = ZC_IMPLANT_AREA.IMPLANT_AREA_C Filter on OR_IMP.IMPLANT_AREA_C Display ZC_IMPLANT_AREA.TITLE

Implanted date

in Chronicles	in Clarity
IMP 220	join to OR_IMP_IMPLANT on OR_IMP.IMPLANT_ID = OR_IMP_IMPLANT.IMPLANT_ID display/filter on OR_IMP_IMPLANT.IMPLANTED_DATE

Implant name

in Chronicles	in Clarity
IMP .2	OR_IMP.IMPLANT_NAME

## Scenario 2: Caboodle

Build a Cogito SQL template according to the following specifications. For each of the items in your query, we have provided both the Chronicles location and the Caboodle location of the data. All you have to do build the correct parameters and/or PAF columns in your template, and then assemble your SQL query out of the given joins and SQL columns to appropriately satisfy the specifications.

### Report Request

Build a Cogito SQL template that will be used to generate lists of hospital accounts. Hospital accounts are used to track all of the charges for a single patient encounter, and they are records in the HAR master file. The requirements for this template are as follows.

1. Name your template <your initials> Caboodle HAR Lab.
2. The template should use the Caboodle database.
3. Reports built from this template should only ever return accounts with an account source of Emergency (This means the account was for an ED encounter).
  - A. Hint: Remember that when filtering on a category value in Caboodle, your parameter will copy over the display name, not the numeric category value.
4. Users should only see accounts from their authorized service areas as determined by the Service Area org level.
  - A. If the user has no authorized service areas as determined by the Service Area org level, they should not get any results.
5. Users should also be able to filter their reports by the discharge department of the account.
6. Reports built from this template should only return accounts with an admission date later than the report's start date, and a discharge date prior to the report's end date.
  - A. Test your template with the date range from 1/1/2019 to 12/31/2020.

7. Reports built from this template should always display the following columns:

- A. Account name
  - i. Use a Chronicles-based column
- B. Admission date
- C. Discharge date
- D. Discharge department
- E. The home phone number of the account's guarantor (A guarantor is the person responsible for paying the account balance).
  - i. Build a Chronicles-based column using a Linkage record.

## Data Points

In the real world, you would need to spend a lot of time finding all of the data points required by this report in Chronicles and Caboodle before you begin. We have done that research for you. For every item needed by this report, we have located the data in both databases. It is up to you to decide whether you need the Chronicles location or the Caboodle location for this template, and to build your query and template appropriately.

Hospital account ID

In Chronicles	In Caboodle
HAR .1	BillingAccountFact.AccountEpicId

Account Source

In Chronicles	In Caboodle
HAR 55	BillingAccountFact.AccountSource

Account service area

In Chronicles	In Caboodle
HAR 350	join to BillingServiceAreaMappingDim on BillingAccountFact.BillingServiceAreaMappingKey = BillingServiceAreaMappingDim.BillingServiceAreaMappingKey display/filter on BillingServiceAreaMappingDim.ServiceAreaId

Account discharge department

In Chronicles	In Caboodle
HAR 390	join to DepartmentDim on BillingAccountFact.DischargedFromDepartmentKey = DepartmentDim.DepartmentKey display/filter on DepartmentDim.DepartmentName

Account admission date

In Chronicles	In Caboodle
HAR 400	join to DateDim on BillingAccountFact.AdmissionDateKey =

In Chronicles	In Caboodle
	DateDim.DateKey display/filter on DateDim.DateValue

Account discharge date

In Chronicles	In Caboodle
HAR 425	join to DateDim on BillingAccountFact.DischargeDateKey = DateDim.DateKey display/filter on DateDim.DateValue

Account name HAR .2

In Chronicles	In Caboodle
HAR .2	BillingAccountFact.BillingAccountName

Guarantor home phone HAR 110 > EAR 265

In Chronicles	In Caboodle
Network through HAR 110 to EAR. Use EAR 265	join to GuarantorDim on BillingAccountFact.GuarantorDurableKey = GuarantorDim.DurableKey and GuarantorDim.IsCurrent = 1 display/filter on GuarantorDim.HomePhoneNumber

## Appendix A: Cogito SQL Self-Assessment

### Introduction



The user and template IDs your need for the Cogito SQL self-assessment will be emailed to you once you complete the steps listed in the [Record Creation - Before you Begin](#) section of this appendix.

### Record Creation - Before you Begin

In order to complete your self-assessment, you need a unique user to log in as and a broken Cogito SQL template to fix. Because creating or duplicating a user is taught in classes outside of the courses required for this badge, complete the following steps to have the system create a user record automatically, as well as your broken template:

1. Log in to [access.epic.com](https://access.epic.com).
  - If you are located in Europe or the Middle East, log in to [euaccess.epic.com](https://euaccess.epic.com) instead.
2. Log in using the same credentials you use to log in to the UserWeb.

- If prompted, provide your 6-digit authentication code. (If you haven't configured multi-factor authentication, click the [Learn More](#) link.)
- 3. Find and click the Study <month> Training Magic icon (where month is the month of the version of Epic in which you will get your certification).
  - Note: For the Cogito SQL Self-Assessment, make sure you are accessing a **Study** Training Magic icon instead of a Project Training Magic icon. The Self-Assessment can only be completed in a Study Classic environment.
- 4. Select the following from the drop down menus:
  - **Application:** Cogito
  - **Project:** BEGIN Cogito SQL Self-Assessment
- 5. Put your initials in the **Initials** field and click **Next**.
- 6. Wait a moment (as the system generates your records), then check your email.

A message will be sent to you containing the names and IDs of the records needed for your self-assessment:

- **User ID:** #####
- **Password:** train
- Template (HGR) record created: <your initials> Self-Assessment Visits [#####]
  - Your template has an HGR record ID of '#####'

Your records are now built. To begin your self-assessment complete the following steps:

1. Go to the [Training Home](#)
2. From **Your In-Progress Certificates**, select **Cogito SQL Badge**
3. From the list of requirements, find **COG2030: Self-Assessment** and click **Take Assessment**

Alternatively, you can access the self-assessment from the UserWeb Course Catalog using the following steps:

1. Open the Userweb Course Catalog record for the [Cogito SQL Badge](#)
2. Under the **COG2030v** Cogito SQL entry, expand the option for **1 required self-assessment**
3. Click the link for the **COG2030v Self-Assessment (Required)** to begin your self-assessment.

## Appendix B: Cogito SQL Certification Project

### Introduction



The complete project specification (steps you will need to complete the project) for the Cogito SQL project will be emailed to you along with records you will need to complete the project once you complete the steps listed in the [Record Creation - Before you Begin](#) section of this appendix.

### Record Creation - Before You Begin

In order to complete your project, you need a unique user you will log in as and modify. Because creating or duplicating a user is taught in classes outside of Cogito Fundamentals, complete the following steps to have the system create a user record automatically. You will also be emailed the project specification after you complete these steps:

1. Request certification environment access at [training.epic.com/CertEnvRegistration](https://training.epic.com/CertEnvRegistration).
2. Go to Epic Access.
  - Most learners: Go to [access.epic.com](https://access.epic.com).
  - Learners in Europe or the Middle East: Go to [euaccess.epic.com](https://euaccess.epic.com).
  - Epic staff: Follow the steps on the [Training FAQ](#) wiki
3. Log in, using the same credentials you use to log in to the UserWeb.
  - If prompted, provide your 6-digit authentication code. (If you haven't configured multi-factor authentication, click the [Learn More](#) link.)

4. Find and click the Project <month> Training Magic icon (where month is the month of the version of Epic in which you will get your certification).
5. Select the following from the drop down menus:
  - **Application:** Cogito
  - **Project:** BEGIN Cogito SQL Project
6. Put your initials in the **Initials** field and click **Next**.
  - A. When choosing your initials, do not use any numeric characters. If you use numbers in your initials, you will not be able to begin your project.
7. Wait a moment (as the system generates your records), then check your email.

A message will be sent to you containing the IDs of the records needed for your project.

**Error Messages:**

Recall that these project environments are shared environments used by many trainees. Another trainee may also share your same initials. If someone else has already created records with the initials you are trying to use, the system will notify you to try a different set of initials. Whatever initials you use to begin your project should also be used as you build your records.

**Login Issues:**

If you get an error when trying to log in to the Project environment, check the following:

- Are you logging in with the right user ID and password? Your user ID should be JUST the numbers in the EMP record that was emailed to you.
- Are you logging in to the correct environment? The environment you should be logging in to is listed in the Epic Training Magic Complete email you received.

Your records are now built and linked appropriately. You may now begin your project. If you'd like, print out the project specification that was emailed to you.

© 2024 Epic Systems Corporation. Confidential.