O'REILLY®
Technical Guide

Accelerating Data Pipeline Development

Deliver Data Projects Faster Without Creating Tech Debt

Early Release
RAW & UNEDITED

Compliments of

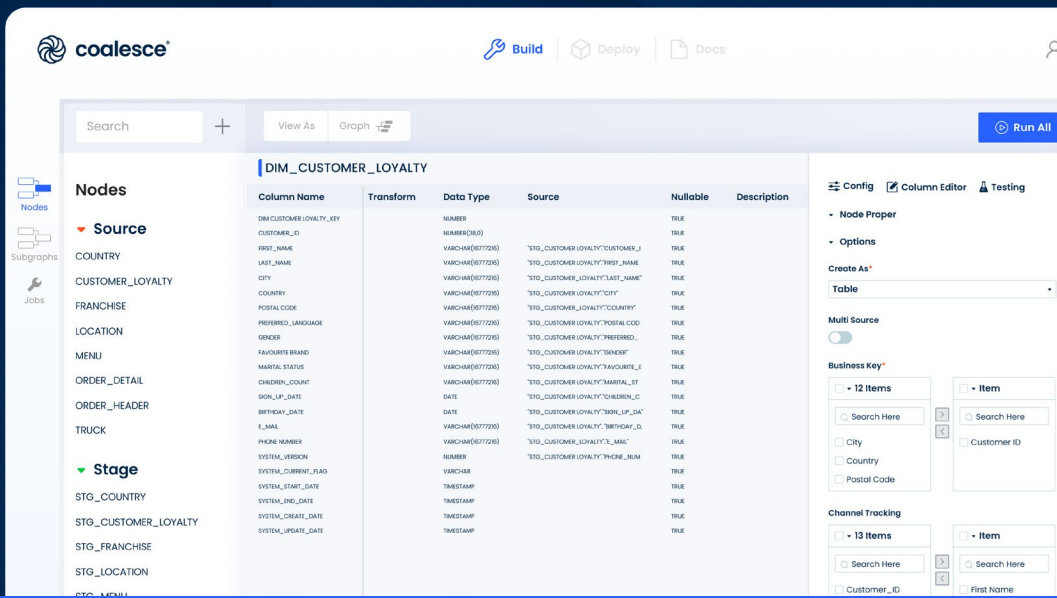
coalesce®

Josh Hall

coalesce®

Deliver data projects 10x faster – without creating tech debt

Build a supercharged data foundation and cut through the noise of data development with Coalesce.



Discover the future of data transformations at Coalesce.io

Accelerating Data Pipeline Development

Deliver Data Projects Faster Without Creating Tech Debt

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

Josh Hall

O'REILLY®

Accelerating Data Pipeline Development
by Josh Hall

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Aaron Black
Development Editor: Melissa Potter
Production Editor: Kristen Brown
Interior Designer: David Futato
Cover Designer: Ellie Volckhausen
Illustrator: Kate Dullea

September 2025: First Edition

Revision History for the Early Release

2025-03-11: First Release
2025-04-02: Second Release
2025-04-24: Third Release
2025-05-22: Fourth Release
2025-06-06: Fifth Release

See *http://oreilly.com/catalog/errata.csp?isbn=9798341608740* for release details.

979-8-341-60874-0

[LSI]

Table of Contents

Brief Table of Contents (*Not Yet Final*)

CHAPTER 1
Getting Started in Coalesce

A Note for Early Release Readers

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 1st chapter of the final book.

If you'd like to be actively involved in reviewing and commenting on this draft, please reach out to the editor at *mpotter@oreilly.com*.

Just like the foundation that supports a house, Coalesce has foundational components that support your development of data products. This chapter will lay the groundwork for the rest of this guide, ensuring you have the tools you need to build end-to-end data pipelines. In it, you will learn about projects and workspaces, which are the core of the development experience in Coalesce. You will discover how to connect Coalesce to your data platform and make your data ready for development. You'll also see how you can work with the rest of your data team to build nodes for all your data transformation needs.

Equipped with this knowledge, you'll be able to get started developing your data in Coalesce and have a foundational understanding for the rest of topics discussed in this guide. Let's start by learning about the user interface.

The User Interface

Coalesce provides a graphical user interface (GUI) that enables you to develop your data while giving you the flexibility to write SQL. The interface is divided into different segments which provide support for different functions throughout the data development lifecycle. These segments include:

- The Projects Page
- The Build Interface
- The Deploy Interface
- The Docs Interface

You also have the ability to manage your Coalesce organization and users from the user menu interface using Org and User Settings. In this section, you'll explore each of the segments of the Coalesce interface.

The Projects Page

The projects page is the default landing page when logging into Coalesce. This page will display any of your organization's projects that you have access to. Projects in Coalesce give you the flexibility to organize your data initiatives for a specific purpose of team goal, as shown in Figure 1-1. Don't worry if you can't make out all the details here, we'll dig into these screens in more detail shortly.

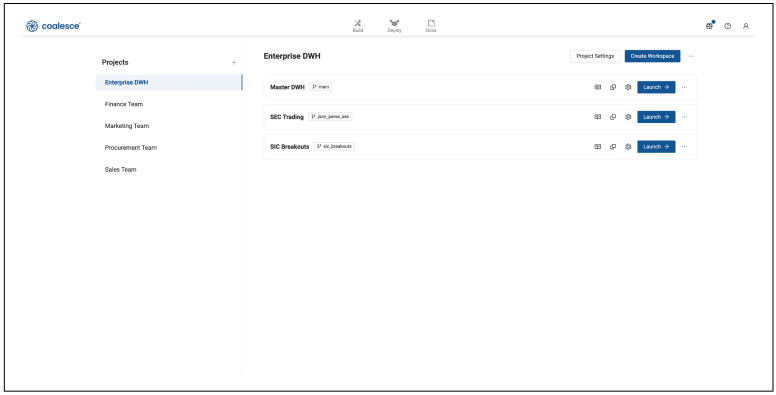


Figure 1-1. The projects page of Coalesce.

The Build Interface

The build interface is where you will spend time developing your data products and building node graphs and pipelines as shown in Figure 1-2. It can be accessed by launching any workspace from the projects page. Users can easily manage each aspect of a data pipeline from the build interface, including creating jobs and subgraphs.

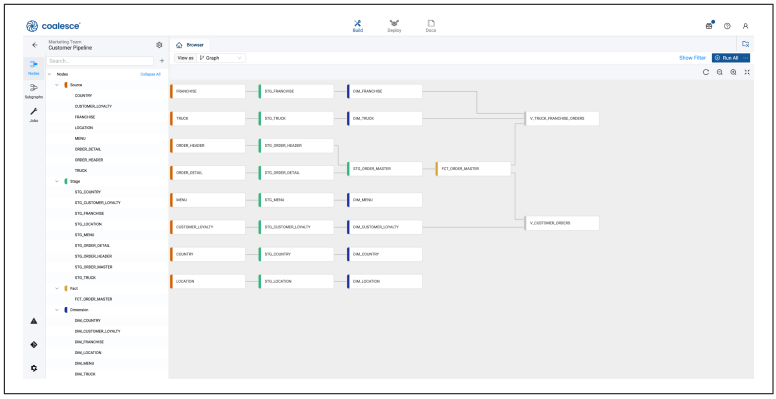


Figure 1-2. The build interface in Coalesce displaying various nodes. Don't worry if you can't make out all the details here, we'll dig into these screens in more detail shortly

The Deploy Interface

The deploy interface is where you can deploy your data projects from the desired state in your git repository, and see a history of your pipeline's deployments and refreshes from the feed on the right side of the screen as shown in Figure 1-3. This interface contains a dashboard for any environment you create, allowing you to see the current status of job runs as well as allowing you to schedule job refreshes.

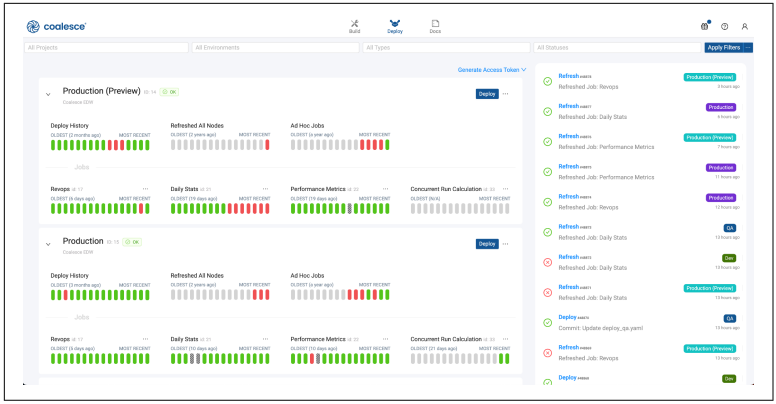


Figure 1-3. The deploy interface where you can manage and monitor your data projects. Don't worry if you can't make out all the details here, we'll dig into these screens in more detail shortly

The Docs Interface

The docs interface captures automatic, real-time documentation about each project and environment in your Coalesce organization. You can find information about each project created, such as the database and schema, column names and descriptions, and even data definition language (DDL) and data manipulation language (DML) as shown in Figure 1-4.

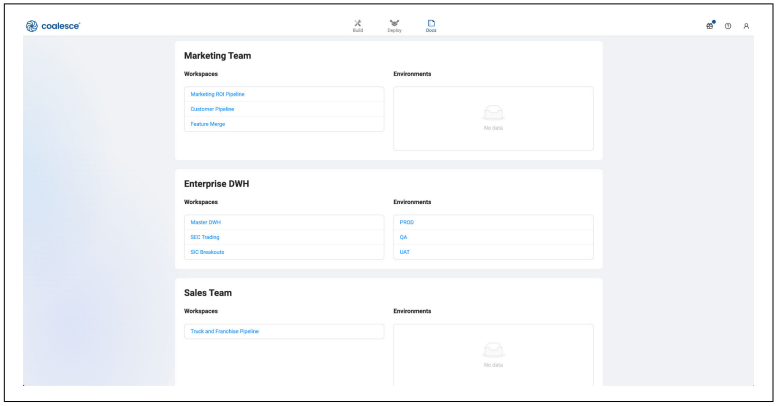


Figure 1-4. The docs interface displaying all of the workspaces and environments that have been documented

The User Menu

As a user within Coalesce, you will have access to the user menu. The user menu can be accessed in the upper right hand corner, denoted by the user icon as shown in Figure 1-5. Within the user menu are the Organization and User settings of your organization.

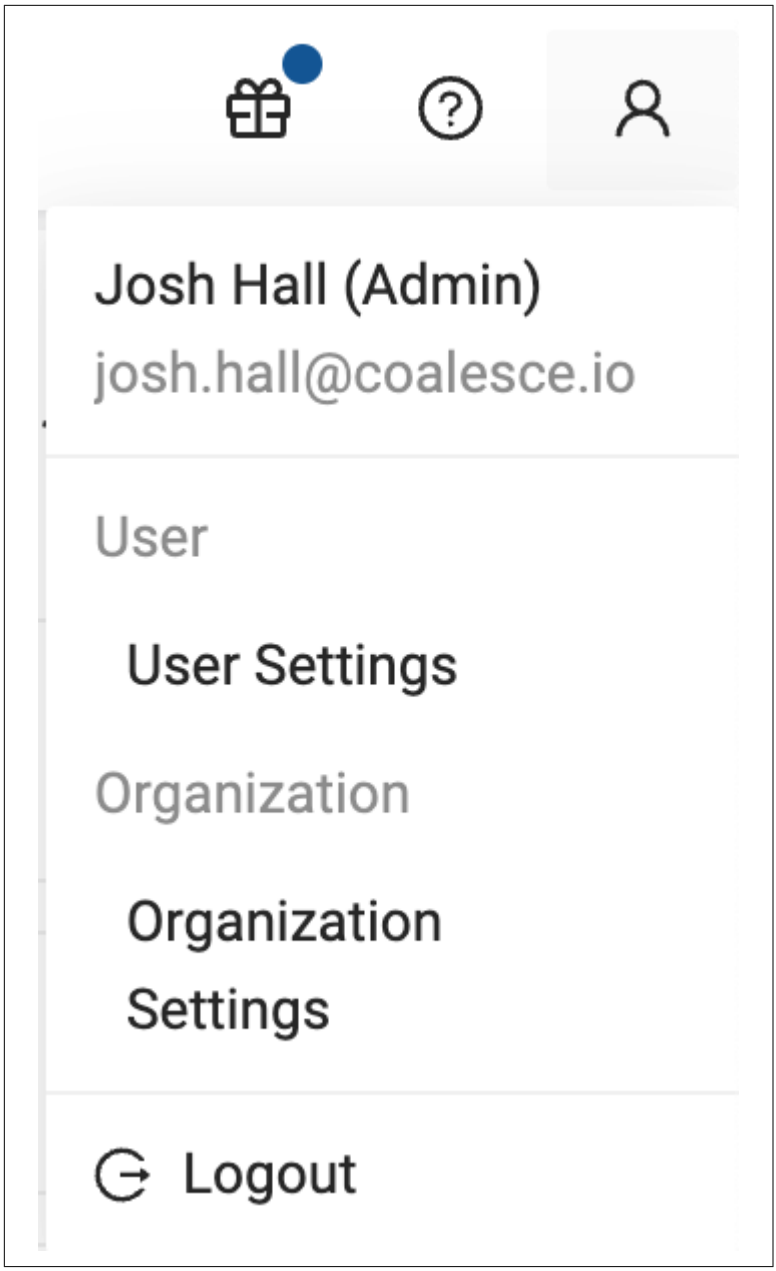


Figure 1-5. The user menu opened to display User and Organization settings

Organization settings provide management access to the following controls for your Coalesce organization:

- User management – adding, removing, or modifying users
- Single Sign-On
- Preferences – such as the Coalesce parser sample size

User settings provide you with the ability to manage your individual user with the following controls:

- Configuration of your Git settings
- Support information about your Coalesce account
- Changing your password

With the ability to navigate the Coalesce interface, you can now dive into setting up your first Coalesce project for developing your data.

Projects

You learned about the projects page in the previous section, but now it's time to take a step further to learn more about projects. Projects give you the ability to organize your data development in a structured way, similar to how folders allow you to organize documents in Google Drive. In this section, you will learn how to use projects, as well as how to set up a project for your own data development.

The Purpose of Projects

Projects provide multiple advantages for data teams developing their data. The first of these advantages is architectural. By utilizing projects, you can decide how your data development processes should be architected.

For some data teams, this means organizing projects by the initiatives that the data team is working on. Others may want to implement a data mesh pattern and use projects to separate each domain of the business. Still others may want different teams within the organization to be managed through separate projects as shown in Figure 1-6. Regardless of the organization pattern, you must have at least one project in order to develop your data.

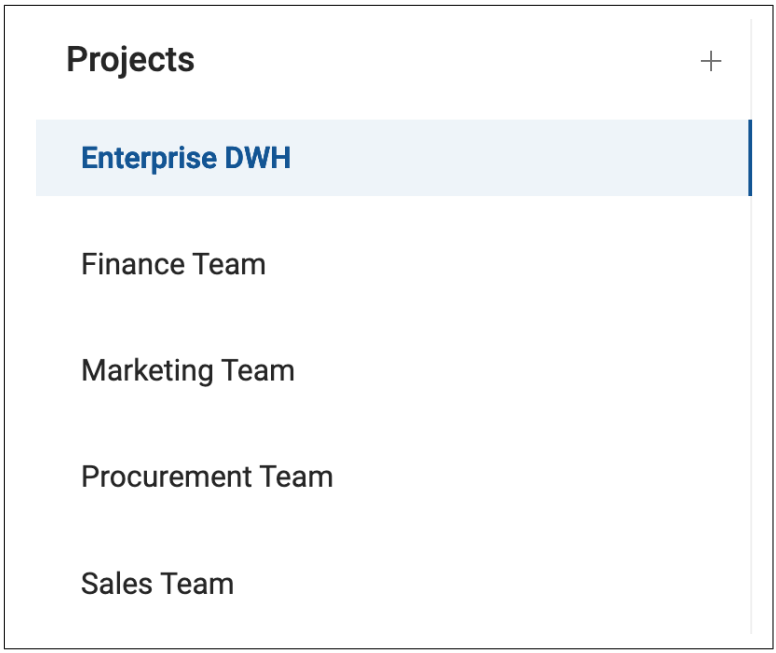


Figure 1-6. Projects in Coalesce managed by the team working on the project

Another advantage of projects is version control. Each project in Coalesce is integrated with your version control system such as GitHub. You will integrate a git repository for each project that you create. This allows each project to be managed separately from the others while providing version control for the specific purpose of the project.

While it is possible to skip this integration, your development experience will be limited to a singular workspace. Coalesce does not recommend developing this way.

The last advantage we'll discuss is role based access control, or RBAC. RBAC allows your Coalesce administrators to determine which users should have access to which projects, as well as determining which level of access each user should have. This provides granular control for all of your data development initiatives.

How to Set Up a Project

Now that you know when to use a project, let's talk about how to set one up. If you have never configured a Git account in Coalesce before, you will need to do this first. To configure a Git account in Coalesce, navigate to the user settings. Within the version control section, select Add New Account. You can follow the instructions on the setup modal to provide the information necessary to configure your Git account as shown in Figure 1-7.

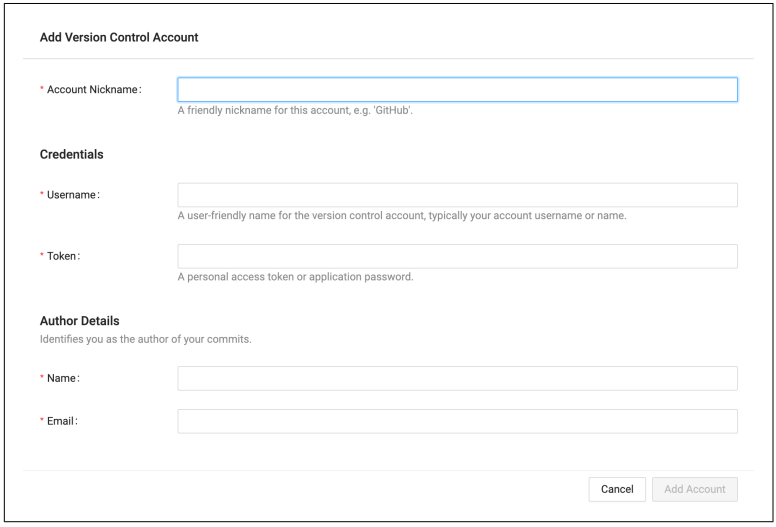


Figure 1-7. The Git setup modal in Coalesce

From the projects page, click on the plus + button next to the Projects header. This will open the project configuration workflow. Here you'll give your project a meaningful name, select the data platform you want to connect Coalesce to, and provide any descriptive information as shown in Figure 1-8.

Figure 1-8. The Project set up workflow where you select your data platform and supply a name and description

Next, you'll need to provide the Git repository URL from your version control system. You can skip this step to create a project without version control, but I don't recommend this. Once you supply the Git repository URL, you can select a Git account configuration from Coalesce. Once you have selected your Git account, your setup is complete and you can complete the project setup workflow.

Workspaces

With your new knowledge of projects, you can now move on to setting up a workspace for your data development projects. A workspace is a sandbox environment where you can complete the development of your data. Each workspace has its own graph, storage locations, macros, node types, data platform connection configuration, and Git branch – all of which you will learn about in subsequent sections. You can create multiple workspaces to work on different tasks and merge them into your codebase. In order to begin building your data pipelines, you will need to connect your workspace to your data platform. Let's go over how to do that now.

Within any project, select the Create Workspace button in the upper right corner of the project – this will open the workspace creation workflow. You will be asked for a workspace name and description. Next, you will select the branch and commit you want to create your new workspace from. For example, you may want to create a workspace from your main branch in order to design a new forecasting

pipeline in your data warehouse. Once you have selected the branch and commit, you will supply the name of the new branch you wish to create and finish the creation of the workspace.

With a workspace created, you can now connect it to your data platform. By clicking on the gear cog icon next to the workspace Launch button, you can provide the information about your data platform connection. In the case of the Figure 1-9, this workspace is connected to Snowflake, but Coalesce supports multiple different data platforms.

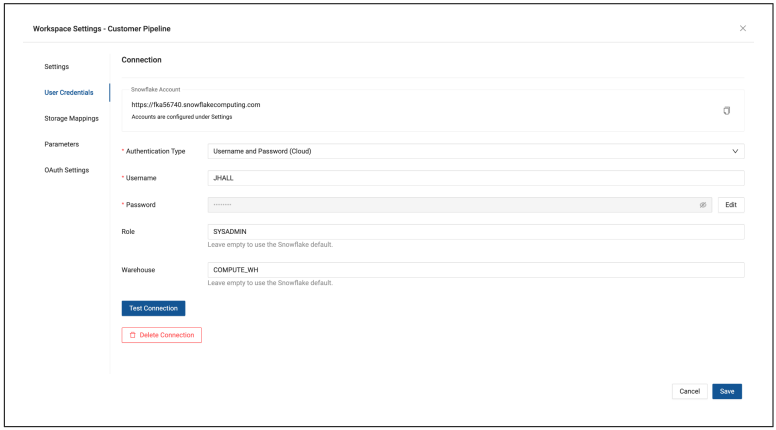


Figure 1-9. Connecting your workspace to your data platform. In this case, Snowflake

Coalesce supports multiple different authentication types. The two most common are OAuth and Username and Password. Once you have supplied Coalesce with your account information and connection criteria, you can test the connection to your data platform. Once successful, you are ready to map your workspace to the data that exists in your data platform.

Once you have connected your workspace to your data platform, you can select the blue Launch button to enter the build interface. Before you can begin your data development in the build interface, there are a few more configuration items to complete. You can find these items in the Build and Workspace settings. We'll quickly explore each of these.

Build Settings

The build settings page can be accessed by clicking the gear cog icon in the lower left corner of the build interface. Within the build settings you can manage all aspects related to development within your workspace. This includes:

- Storage Locations and Storage Mappings
- Development Workspaces
- Environments
- Macros
- Node Types
- Packages

Each of these items has their own settings which can be configured by selecting each item.

Workspace Settings

Your workspace settings provide management for the connection to your data platform. These workspace settings can be accessed in one of two ways. You can either select the gear icon next to the workspace name, or select the same icon next to the workspace in the Workspace selection from the Build Settings as shown in Figure 1-10.

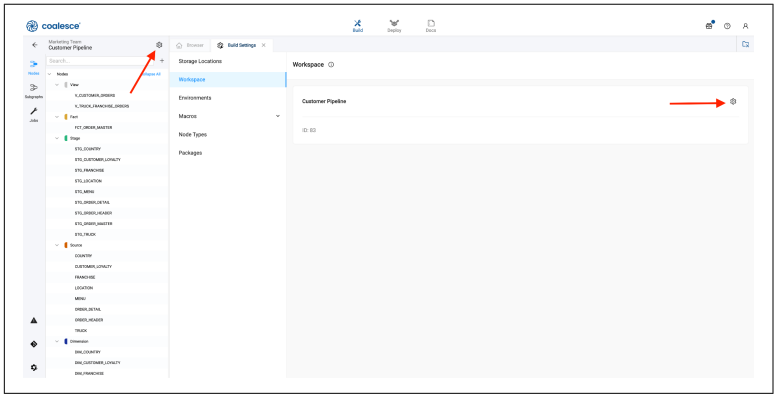


Figure 1-10. Where to open the workspace settings inside of a workspace

Now that you know where all of the settings are available in your workspace, you can finish the final configuration of your workspace to begin building your data pipelines.

Storage

So far you have created a project and workspace and connected the workspace to your data platform. But now you need to tell your workspace what data in your platform you want to develop with. This is where storage locations and storage mappings come into play.

Storage Locations

Storage locations are a logical representation of a database and schema in your data platform. You can think of them as the glossary that points to the chapters in a book. Storage locations themselves don't contain any data or perform any action on their own. Instead, they act as logical containers for the databases and schemas you want to use in your data pipeline development.

For example, you may have a database and two schemas within that database which contain source data for a pipeline you wish to develop. You want the results from your data pipeline to be output in a different database and schema. In this case, you could use three storage locations, two for the source data and one for the output or target destination as shown in Figure 1-11.

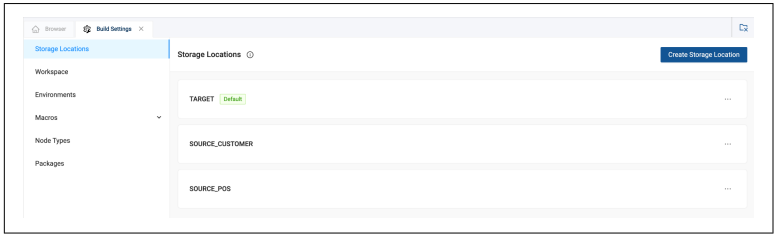


Figure 1-11. Storage locations created to be mapped to physical destinations in your data platform

Storage locations can also be used for other use cases as well. For example, if your organization uses a medallion architecture, you can have a storage location for each level of your architecture i.e. bronze, silver, gold. Or if you leverage a staging layer in your data pipelines,

you could have a staging storage location where all staging tables are created in the same location.

You will notice that there is always a *default* storage location. This is the location that, unless configured otherwise, all tables in your data pipeline will be created within by default.

It's important to note that *storage locations cannot be renamed once they are created*.

Once your storage locations are created, you can map each storage location to a physical destination in your data platform using storage mappings.

Storage Mappings

As you just learned, storage locations are just logical containers that can point to physical locations in your data platform. Storage mappings are what tie your storage locations to a database and schema in your data platform. To configure storage mappings for your workspace, you will need to access your workspace settings and select Storage Mappings.

Each of the storage locations you have created will show up as an item to be mapped to your data platform. For each storage location, you can select the database and schema that you want each to point to as shown in Figure 1-12. Ensure that your default storage location is mapped to a database and schema where you expect your tables to be output.

Figure 1-12. Mapping physical databases and schemas to the storage locations previously created

Storage mappings can be updated at any time from the workspace settings. You can also add more storage locations and map them in the same way described here. For this guide, we will be using a fictional foodtruck company dataset for our illustrations,where our data sources are the customer and POS data and our target storage location is a development database and schema. We can deploy this to a production environment later on.

Adding Users

You now have a fully configured and ready for development project and workspace. With your data ready for development, you can add your team into your Coalesce account to collaborate alongside them. To add users, you will need to access the Org settings from the User Menu. As a Coalesce administrator, you will be able to add users to your Coalesce account and assign them appropriate roles.

Table 1-1 provides guide to user roles in Coalesce and the permissions that they contain.

Table 1-1. Coalesce User Roles and Associated Permissions

Role	Permissions Summary	Recommended For
Organization Administrator	The creator of the Coalesce App is automatically assigned as organization administrator. Only organization administrators can add other users, including other organization administrators. They have full access to all functionality in Coalesce.	Full administrative control
Organization Contributor	They can't add new users to the organization. They have access to read documentation, create API tokens, user settings, and Git account information. They will be able to set up a project, configure Git, add members to projects,and oversee work. They'll only have access to the projects they create themselves. If there are multiple organization contributors, they will need to share access with the organization contributor.	Managers who decide how each person will contribute to a project.
Organization Member	This is the default role. They can edit Git account information, create API tokens, and read documentation.	Default Role

Now that your users are added into your workspace with the proper permissions, your team can begin building data products by adding data sources

Adding Data Sources

With your team ready to collaborate in Coalesce and your workspace ready to develop your data, you can begin adding data sources to your graph. You can do this by launching your workspace and selecting the plus + button in the upper left corner of the build interface. This will open the add data sources modal, which will display all of the storage locations you mapped your data to, and the objects available in those locations as shown in Figure 1-13.

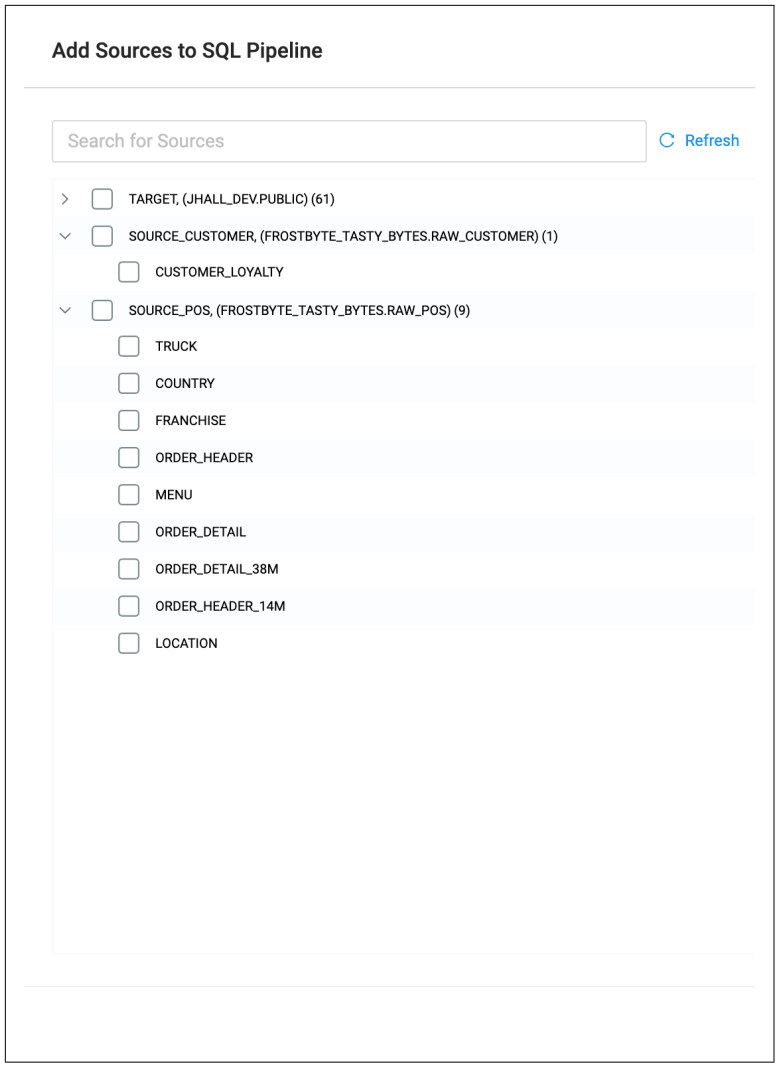


Figure 1-13. Data sources modal showing all of the objects available from each of the storage locations configured.

You can select as many or few data objects as you want to add to your data pipeline. Once you have selected the data sources you want to work with, you can add them to our graph. Coalesce will add each object into your graph as a node. In the next chapter of this guide, we will discuss what a node is in detail, but for now, all you

need to know is that it's a visual representation of the objects you have added into your graph.

Building on Your Foundation

This chapter laid the foundation for all the components you will be using in Coalesce. In it you learned how to navigate the user interface and how each segment of the interface is used. You learned about the purpose of projects and how to set one up. You also learned how to create a workspace and connect to your data platform. At the end of the chapter, you saw how to bring data sources into your graph. At this point, you are ready to begin developing your data.

However, before you begin building a data pipeline, it's important to understand the core concepts of how to develop your data in Coalesce. In the next chapter, we'll explore everything you need to know to begin your journey building data pipelines and applying all of the knowledge from this chapter, to the concepts of pipeline development in Coalesce.

CHAPTER 2
Coalesce Core Concepts

A Note for Early Release Readers

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 2nd chapter of the final book.

If you'd like to be actively involved in reviewing and commenting on this draft, please reach out to the editor at *mpotter@oreilly.com*.

In the previous chapter we laid the foundation for your understanding of Coalesce components, such as projects, workspaces, and storage locations. In this chapter, we will lay the building blocks on top of your foundation, by providing you with the core concepts that empower Coalesce data development. You will learn why column-aware architecture is important within data transformations, what nodes are and how they harness column-aware architecture, the data pipeline approach, and managing data development.

At the end of the chapter, you'll walk away with the knowledge that will provide you with the framework you need to understand and develop your data on the platform. And there is no better place to start than understanding column-aware architecture.

Column-aware Architecture

Data processing workloads in modern data platforms don't just operate on the scale of thousands of database tables; they run on hundreds of thousands—even millions—of columns. Coalesce is built from the ground up to automate data transformations while keeping a code-first approach and flexible interface. This all starts with a column-aware architecture.

But what is column-aware architecture? Column-aware architecture, or being column-aware, is an approach to managing data transformation with an understanding of columns and how they are connected. With this understanding, Coalesce provides automated column-level lineage, while enabling the creation and maintenance of database objects at scale. This column-aware architecture captures metadata for each object created, allowing you to leverage incredible development speed and agility in your data transformation workloads.

By using column-aware architecture, Coalesce shifts the paradigm of traditional data transformation processes to an automated, reusable, and scalable approach. We'll see how this translates to the foundational building blocks of Coalesce (nodes), but first, let's dig deeper into the benefits of building a data transformation platform using column-aware architecture.

Data Patterns

Column-aware architecture is the key to standardizing how transformations are applied, how tables are structured, and how columns are logically connected. This standardization can be referred to as a pattern, which is a reusable step in a data transformation process that represents a logical transformation. This can include:

- Incremental & processing logic
- Materialization logic
- Deployment logic

These data patterns are essential for the creation, management, and accessibility of data, especially at enterprise scale. Coalesce provides a platform to rapidly implement these data patterns by leveraging metadata at the column level. With column-aware metadata, you

can build a single reusable data pattern that can be applied across any of your data transformations.

Take, for example, a simple type 2 slowly changing dimension–an industry standard for tracking historical data, such as a customer's current address and what it was six years ago, and every change in between. Writing the complex SQL to deliver this functionality could take hundreds of lines of code. Because Coalesce is column-aware, this can be defined once and then reused as many times as required without the need for writing the code each time.

In Coalesce, column-aware data patterns streamline complex, time-consuming manual coding tasks. This speeds up data processing and lets you focus on broader strategy while applying reusable logic with confidence..

Impact Analysis and Lineage

Having data patterns is powerful, but if those patterns aren't coupled with the ability to understand and manage your entire pipeline in one place, you may end up spending all of the time you saved building patterns having to understand how changes impact your pipeline. This is a second powerful benefit of using column-aware architecture: this column-awareness enables complete impact analysis and lineage at the column level.

Imagine you've just deployed a pipeline that powers critical stakeholder dashboards. It's Monday morning, and you wake up to an email from your SaaS ETL provider: they've changed the schemas of several tables in your pipeline. Panic sets in. Which columns in my pipeline are impacted? Are dashboards already broken? How many transformations are affected?

With column-awareness, you can quickly see how changes to your data affect everything downstream as seen in Figure 2-1. You'll know exactly what's been impacted and can fix issues before they become problems. You can see exactly how each object and column affects your pipeline at any moment, no guesswork or managing dozens of SQL tabs.

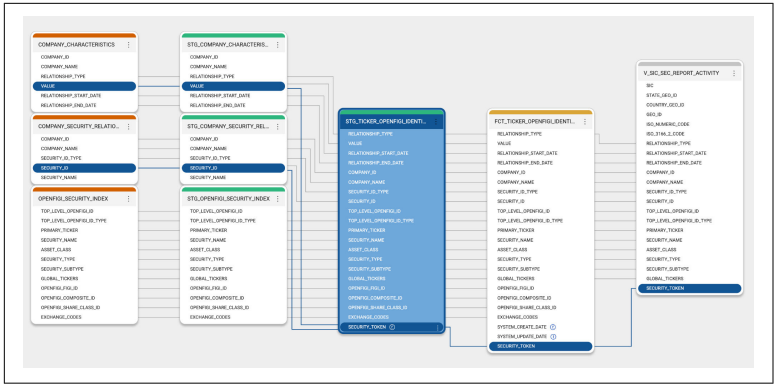


Figure 2-1. Column level lineage showing the impact of every column throughout a pipeline.

Additionally, with column-awareness, you can perform bulk operations directly at column granularity and across your data platform, making source data changes or business logic changes easy and straightforward to implement. For a data consumer, Coalesce provides clear documentation at the column level, showing where the data came from and how it was calculated. This transparency helps build trust across the organization.

Scale and Governance

Column-awareness drives efficiency and accuracy in data management, even at scale—across hundreds of thousands of columns, multiple teams, environments, and the entire business.

Column-aware state management minimizes the risk of data loss and errors by enabling in-place, column-level modifications instead of requiring full table re-creations. It also offers detailed column-level visibility into changes over time, a critical aspect of effective DataOps and governance practices.

For instance, in a deployment with thousands of columns, Coalesce eliminates the need to build complex architectures to handle changes. Instead, you can manage in-place edits out of the box This approach reduces costs, enhances deployment visibility and planning, and fosters trust among data consumers throughout the organization.

This column awareness enables column propagation, allowing you to add or remove columns across the entire pipeline with ease

through the user interface. This functionality streamlines column management across your project, making column awareness a core advantage.

Now that you have an understanding of column-aware architecture in Coalesce, let's dive into the foundation of data development: nodes.

Nodes

Nodes are the core components used to build data pipelines in Coalesce. In Chapter 1 we defined a node as a visual representation of an object (table, view). Now that you have an understanding of pattern based development, we can provide a more precise definition.

A node is a visual representation of an object within your data platform as seen in Figure 2-2. It serves as a building block for constructing data pipelines and leverages data patterns to automate your data transformations. Nodes are classified as node types, such as a stage or fact node type–more on this shortly!

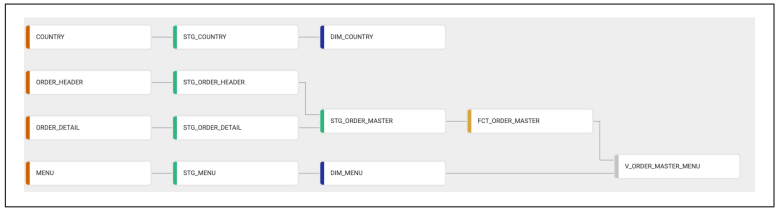


Figure 2-2. Nodes in the build interface representing a data pipeline.

Nodes enable pattern-based development by allowing you to define a transformation once and repeatedly apply it, streamlining automation and ensuring consistency across your pipeline. You can begin to see how leveraging nodes can accelerate development while ensuring quality and consistency across your pipelines, as everyone works from a shared pattern or standard. To build effectively with nodes, it's essential to understand how node types are created and function. Let's take a closer look at what makes up a node.

Node Architecture

Node types consist of three components: a node definition, a create template, and a run template. Each of these components performs a

specific role in the representation and execution of a node type. Let's start with the node definition.

Node definition

The node definition defines the attributes available within a node type, such as naming conventions and node type colors. It also specifies the UI elements used to configure each individual instance of the node type. For example, if you want to build a stage node type for creating a consistent staging layer in your pipeline, the node definition would define the naming convention for the node each time it's used. However, each instance of the stage node can be configured differently, based on the options provided through the UI elements in the node's configuration. Such as one stage node type being materialized as a view and another being materialized as a table.

The node definition is defined using YAML as seen in Figure 2-3.

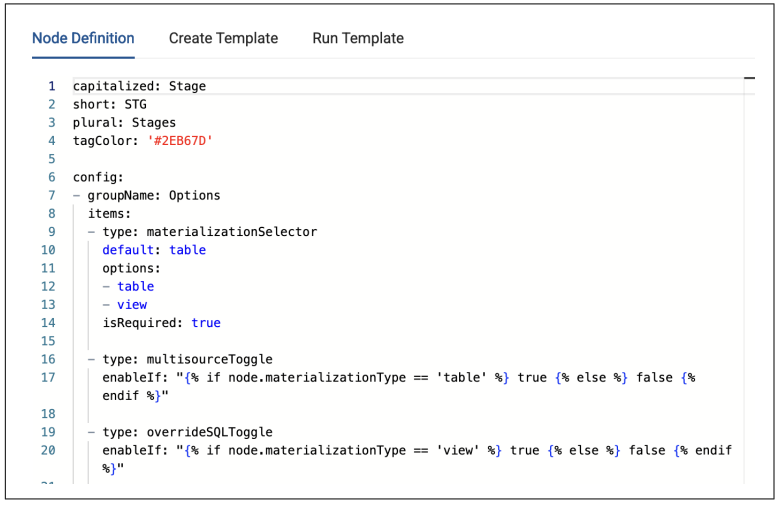


Figure 2-3. YAML used in the node definition to define the attributes of the node type.

The Coalesce documentation contains information on the configuration options available to include in your node definition YAML file as seen in Figure 2-4.

Figure 2-4. Coalesce documentation representing the various configuration options for the node definition of a node type.

Once you have your node definition defined, you can configure the Create and Run templates.

Create Template

The create template for a node type defines the logic used to automate the creation of the object. Typically, this involves Data Definition Language (DDL) that leverages attributes from the node definition—such as the selected materialization type—to execute a CREATE statement in SQL.

You can define create templates using a combination of SQL and Jinja, as shown in Figure 2-5.

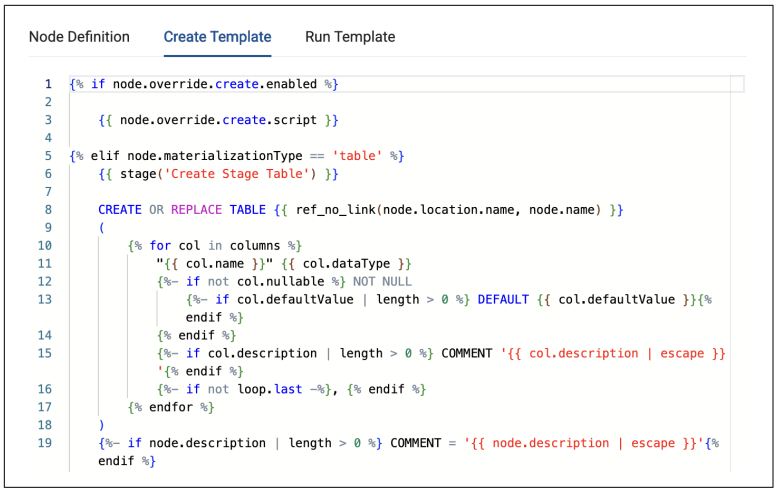


Figure 2-5. The create template representing the SQL and Jinja used to automate your DDL.

After you define the logic, Coalesce runs the DDL for each object instance in your pipeline. Executing the create template creates the object in your cloud provider automatically.. It is important to note that the create template is *only* creating the object, it is not loading data into the object. In order to insert data into the object, you need the run template!

Run Template

The final component of a node type is the run template. This is where you define the logic to automate the execution of your transformations. Typically, it involves Data Manipulation Language (DML) that leverages Coalesce's column-aware architecture to execute SQL and populate objects with data.

Like create templates, run templates are defined using a combination of SQL and Jinja as shown in Figure 2-6.

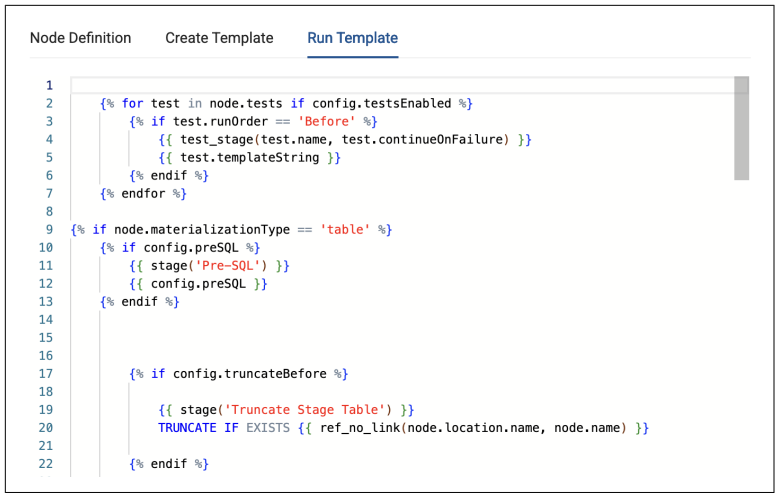


Figure 2-6. The run template representing the SQL and Jinja used to automate your DML.

With the logic defined, Coalesce will automatically run the corresponding DML for each node type. This will populate the objects in your data platform with data based on the logic and upstream data coming from your data pipeline. It's important to note that some node types may not need a run template, such as a view node type, which is just storing a query to be run in the future. A view is never actually populated with data.

Now that you understand the components of a node type and how you can create reusable patterns, let's explore the advantages of developing with nodes.

Importance of nodes

Up to this point, we've seen how column-aware architecture drives pattern-based development, accelerating data transformations within Coalesce. Now, it's important to bring everything together and understand how these concepts translate into more efficient pipeline development.

Standardization

We've already discussed how node types establish a framework for consistent data development across your team. Now, let's delve into

why this standardization is essential for long-term, scalable pipeline development.

Standardization ensures that everyone operates from the same foundation, automatically and consistently. There's no need for new custom code, special permissions, or isolated environments. A node type is defined once, and it's ready to use repeatedly without additional setup.

This standardization enables your team to embed best practices directly into the nodes. Each time a node runs, you can trust it to perform exactly as intended. It also allows you to optimize your code, ensuring that each instance of a node type runs efficiently and minimizes compute resource usage within your data platform.

Standardization streamlines development by reducing repetitive object creation and manual coding. This allows your team to focus on data transformation and modeling rather than boilerplate setup. By minimizing development overhead, you can build and deploy pipelines more efficiently.

As discussed earlier, you can define a dimension node type to support type 2 slowly changing dimensions (SCDs). Whenever you need to implement a type 2 SCD, you can add the dimension node to your pipeline and configure it with just a few clicks as seen in Figure 2-7. In seconds, you can automate the execution of a type 2 SCD, eliminating the need to write code from scratch or manually adapt copied code to your specific tables.

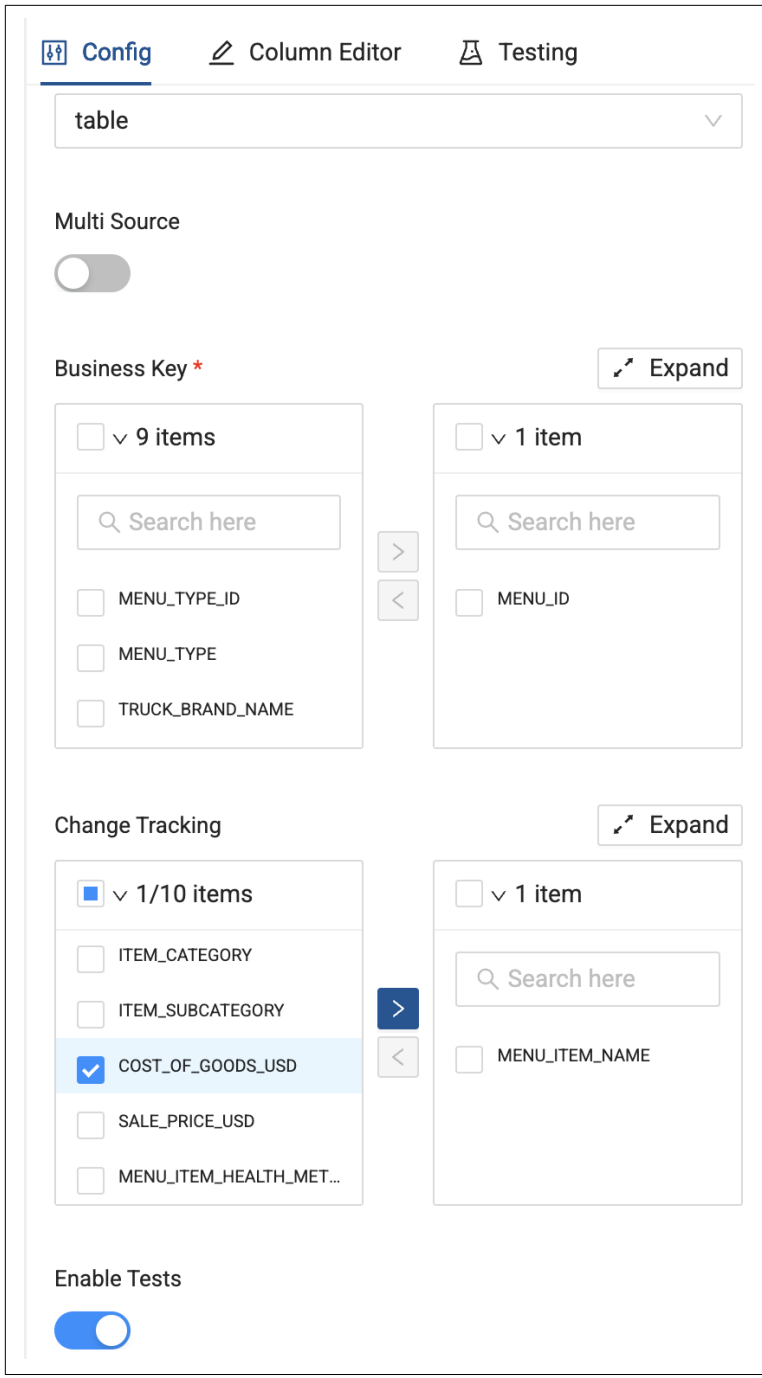


Figure 2-7. The configuration options for a dimension node, allowing you to set up type 2 SCD capabilities in seconds.

By using standardization in your pipeline development, you gain all of the advantages discussed here. This does not mean that you are giving up flexibility, as you'll see in the next section.

Flexibility

Coalesce provides multiple built-in node types, including a dimension node with preconfigured Type 2 SCD options, to streamline development. While these predefined options simplify automation, data projects often require customization.

Coalesce supports both custom code and a GUI-driven interface (Figure 2-8), allowing you to define node behavior precisely while maintaining an efficient and user-friendly development experience.

Figure 2-8. Coalesce provides you the ability to define your own data patterns and write custom SQL all through a power user interface.

Coalesce enables you to create User Defined Nodes (UDNs), allowing you to configure nodes tailored to the specific needs of your data development. Additionally, Coalesce provides a variety of prebuilt node types available on the Coalesce Marketplace, designed to address a wide range of use cases. We'll learn more about Coalesce Marketplace in the next chapter.

Independent blocks of logic

You've likely encountered SQL transformations that span hundreds of lines, filled with multiple CTEs and repetitive code that violates the DRY (Don't Repeat Yourself) principle. Coalesce node types are designed to address this by breaking transformations into logical, reusable blocks.

Instead of bundling multiple CTEs and subqueries into one sprawling query, Coalesce allows you to handle each logical task in its own

node. This means each CTE in a large query can become a separate node in your pipeline.

You might be thinking that building pipelines this way will result in more objects compared to consolidating everything into a single query—and you're right! But typically, this is only a problem because more objects typically means more development time and management. But, because Coalesce relies on standardized, reusable nodes, the building process is incredibly fast. And with column-aware architecture in place, managing a pipeline at any scale becomes simple. As we'll explore in the next section, there are significant advantages to designing pipelines using this pipeline approach rather than relying on complex CTEs.

The Pipeline Development Approach

Complex data processing often involves breaking tasks into sequential steps, where the output of one step feeds into the input of the next. While Common Table Expressions (CTEs) can achieve this, Coalesce takes a modular, pipeline-based approach that offers significant advantages over traditional CTE development.

Transparency
: One of the primary benefits of a pipeline approach is improved transparency. Instead of consolidating all logic into a single query, Coalesce uses nodes to represent individual logical blocks. This modularity makes it easy to navigate your pipeline and understand the function of each step. Additionally, you can run individual nodes independently to view their results in context, simplifying analysis and debugging.

Troubleshooting
: The transparency of pipeline logic also simplifies troubleshooting. Debugging a monolithic SQL query with hundreds of lines and multiple CTEs can be tedious and error-prone. In Coalesce, each logical step is a standalone node, allowing for a more straightforward process to identify and resolve errors or data issues. This modular design reduces the overhead required to troubleshoot and maintain data pipelines.

Testing
: Coalesce facilitates out of the box and SQL-based testing in each node in the pipeline as shown in Figure 2-9. For example,

you can validate uniqueness, check for null values, or run other data quality checks at any step. By catching unsupported data or logical errors early in the pipeline, you can prevent issues from propagating to downstream processes, saving time and effort.

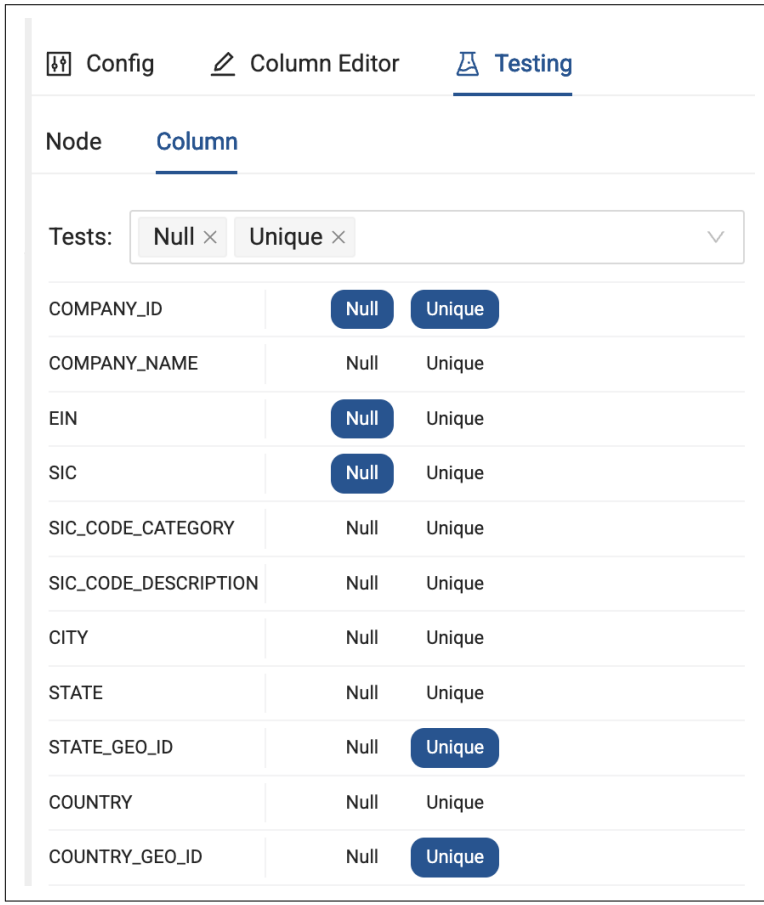


Figure 2-9. Out of the box tests that can be applied easily to any column in the node.

Developing pipelines in this way allows complete control over each element, while providing a streamlined approach to development.

Reusability

Breaking a data pipeline into nodes promotes reusability of logic across the pipeline. For instance, if you create a node to deduplicate

records in an orders table, downstream nodes can reference and reuse this logic without reprocessing. This eliminates redundant computation and ensures consistency across the pipeline.

In contrast, using CTEs often requires duplicating logic across multiple queries, which is harder to manage and less efficient.

Developing with a pipeline approach offers substantial advantages in transparency, troubleshooting, testing, and reusability. While this method may result in more objects in your data platform, Coalesce's column-aware architecture and node standardization make managing pipelines of any size seamless and efficient.

The Development Workflow

So far, we've explored the fundamentals of nodes and the advantages of building pipelines with them in Coalesce. But what does the actual development workflow look like? While we'll dive deeper into pipeline construction in the next chapter, there are two essential concepts to understand for developing your data in Coalesce: *workspace development* and *environments*.

Workspace Development
: Each workspace in Coalesce comes with a build interface designed for data development. When working within a workspace, each user will have their own set of credentials. Typically, development occurs in a sandbox or personal space within your data platform. This means that any objects you create are isolated to your storage location, allowing you to experiment and iterate without impacting shared environments.

  Think of a workspace as your personal sandbox for developing and testing data pipelines. Once you've finalized your pipeline, you can deploy your work to an environment for broader use.

Environments
: Coalesce provides the use of environments for deploying data pipelines, allowing for a structured workflow from development to production. After finalizing changes in the workspace, commit them to a Git branch and deploy to the target environment.

  Environments in Coalesce are tied to designated locations in your data platform, such as specific databases or schemas through storage locations and storage mappings. For best prac-

tices, Coalesce supports multiple environments, such as *QA* and *PROD*, allowing you to test and validate your work before deploying it to production. Figure 2-10 illustrates an example of a typical environment setup.

Figure 2-10. The deploy interface in Coalesce, where you can deploy your pipelines to higher environments in your data platform.

Knowledge Sync Complete

You are now equipped with the foundational knowledge needed to start developing your data in Coalesce. In this chapter, we covered column-aware architecture, how it supports pattern-based development through nodes, and why nodes serve as the optimal building blocks for building pipelines.

If you encounter challenges as you progress through this guide, feel free to revisit Chapters 1 and 2 for a refresher.

Up next, we'll walk you through the process of building a data pipeline—from data source to insight-ready tables. See you in the next chapter!

CHAPTER 3

Building Data Pipelines in Coalesce

A Note for Early Release Readers

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 3rd chapter of the final book.

If you'd like to be actively involved in reviewing and commenting on this draft, please reach out to the editor at *mpotter@oreilly.com*.

So far, you've spent time learning to set up Coalesce and explored core components and concepts of the platform, such as projects, workspaces, node types, storage locations and mappings, and column-aware architecture. In this chapter, you'll put this knowledge to work by learning how to build data pipelines in Coalesce.

In this chapter you'll learn how to add data sources, create nodes, and write SQL to transform data directly within any node. You'll also master functionality like creating joins, bulk-editing columns, and applying tests to ensure data quality. Plus, you'll discover how Coalesce Marketplace enhances your pipeline with powerful extensions.

Let's get started!

The Build Interface

In Chapter 1, you briefly learned about the Build Interface, which is where you will be spending your time in this chapter. The Build Interface is where you will develop your data products and build node graphs and pipelines. You can assess it by launching any workspace from the projects page as seen in Figure 3-1.

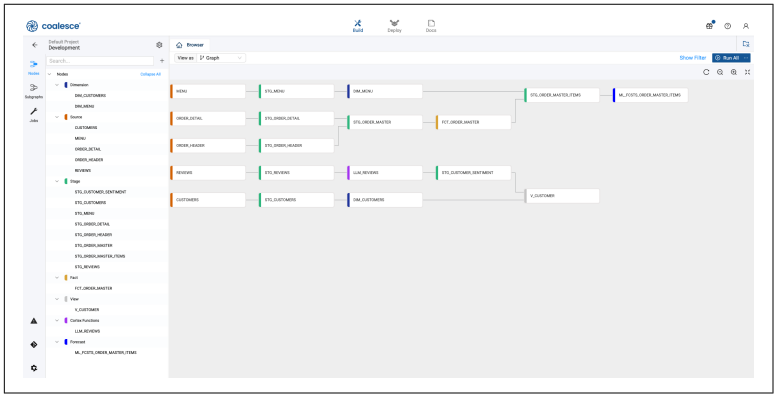


Figure 3-1. The Build Interface in Coalesce, displaying nodes organized in the form of a pipeline.

The Build Interface contains all of the necessary components to build data pipelines. In the sidebar in the upper left side of the interface, you'll see navigation options for Nodes, Subgraphs, and Jobs. You'll learn about subgraphs and jobs more in chapter 4, as the focus of this chapter is on building pipelines with nodes.

In the lower left corner of the sidebar, you'll have navigation for your Problem Scanner, which will alert you on any issues or notifications you should be aware of within your data pipeline. You'll also be able to access your Git Integration to easily version control your work. Finally, you'll see the Build Settings cog, which contains all of the settings for the workspace you are building in, as seen in Figure 3-2, such as the storage locations you learned about in Chapter 1. Throughout the rest of this guide, we'll explore each of the items located in the Build Settings so don't worry if you don't know what each of these items mean.

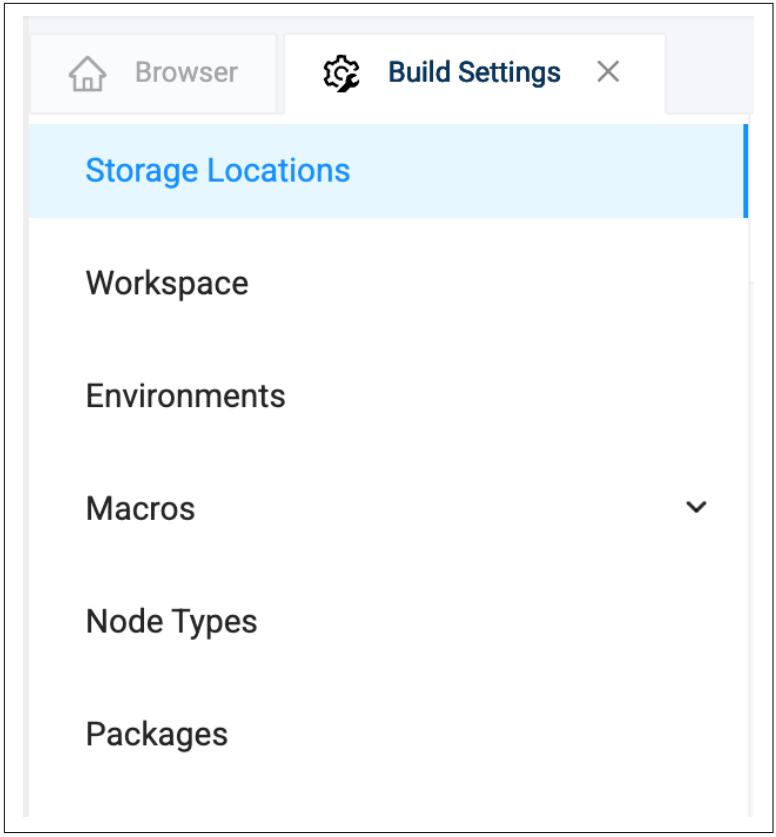


Figure 3-2. The Build Settings available in each Workspace in Coalesce.

The Build Interface also contains the Workspace Settings for the workspace you are working in. You learned all about the Workspace Settings in Chapter 1, when we discussed connecting to your data platform, but you can easily access those settings from the Workspace Settings Cog in the upper left corner of the screen, as seen in Figure 3-3. You can also find them in the Workspace line item in the Build Settings.

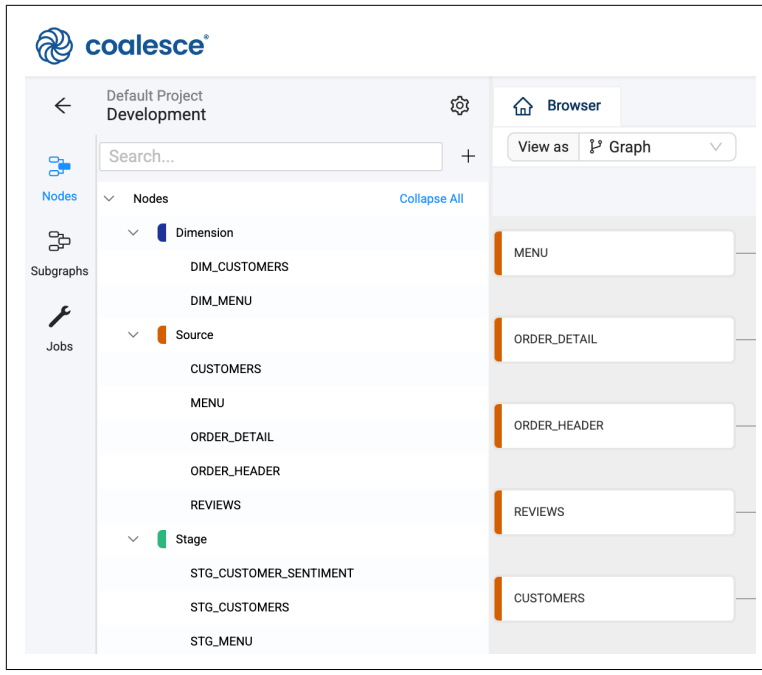


Figure 3-3. The Workspace Settings cog, which allows you to configure your workspace settings.

When it comes to building data pipelines, you'll do this work within the Browser, which is where your Directed Acyclic Graph (DAG) and nodes in your pipeline are displayed, as shown earlier in Figure 3-1.

Now that you're familiar with the essential elements of the Build Interface, it's time to start building data pipelines. First up: data sources.

Adding Data Sources

Data sources are a core component to any data pipeline—after all, you can't build a pipeline without a source. Adding data sources in Coalesce is simple and straight forward. In the upper left corner of the Build Interface, select the plus "+" button, and choose Add Sources. This will open the Data Sources modal, which will display all of the storage locations configured in the Workspace.

Each line item is a Storage Location that is pointing to a database and schema. You can see in Figures 3-4 and 3-5 how the Storage Mappings for each Storage Location correspond to the data sources available in the modal.

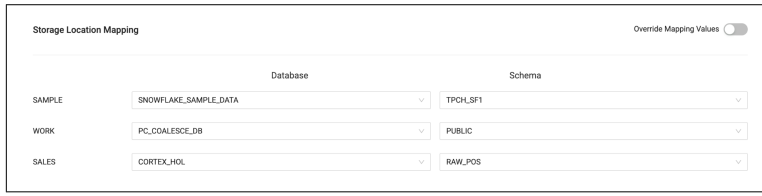


Figure 3-4. storage mappings in the workspace settings, pointing your storage locations to the database and schema in your data platform where your data exists.

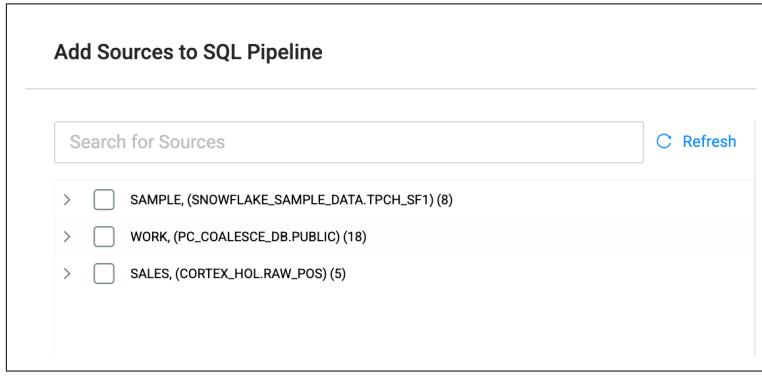


Figure 3-5. The same storage locations and mappings showing up in the data sources selector.

By selecting any of the Storage Locations available, you can view all of the objects available to use within Coalesce. For example, in Figure 3-6, you can see there are eight objects available that we could addinto our pipeline.

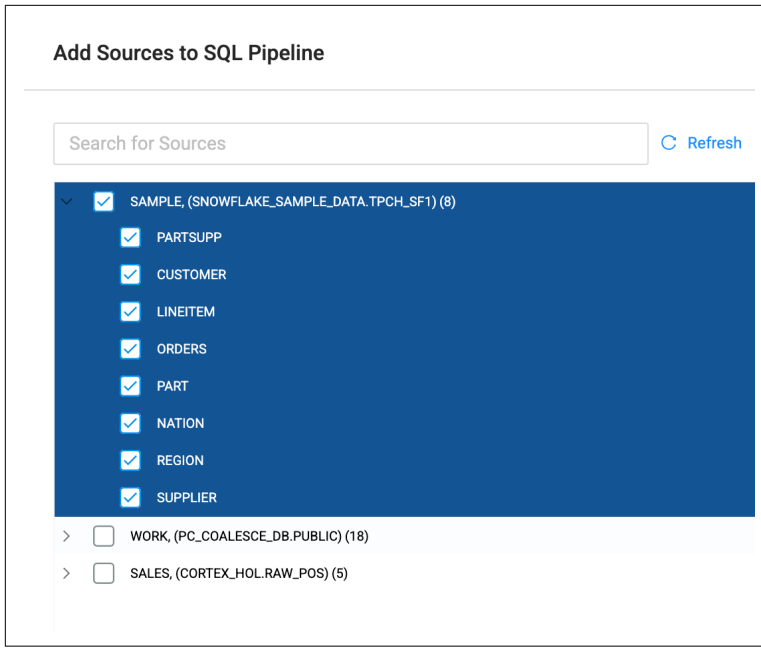


Figure 3-6. Selecting the data sources from the data sources modal.

You can either select the specific objects you need, or select all of them by selecting the checkbox next to the Storage Location name. You can select as many objects between any of the available storage locations. Once you have selected all of the objects you want to add to your pipeline as data sources, select the Add X Sources button in the lower right corner of the modal, where the X represents the number of objects selected.

Coalesce will automatically add all of the objects you selected into the browser of the Build Interface as seen in Figure 3-7. Source Nodes are represented by a dark orange color – as every node has a unique color. This is consistent for any source node added into your Workspace.

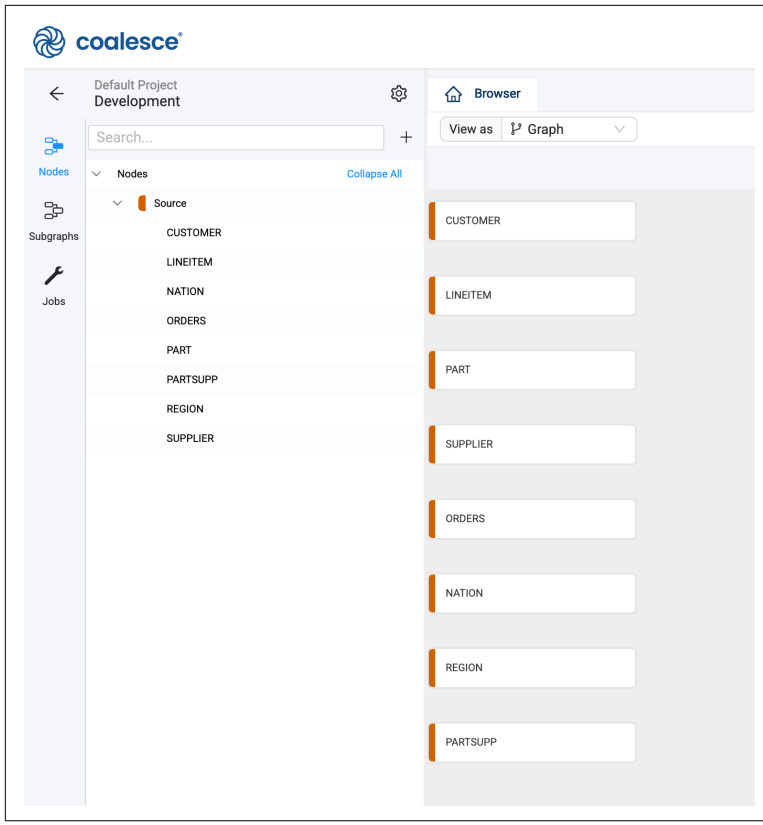


Figure 3-7. Data sources being displayed in the Browser.

With your data sources added into your Workspace, you can now begin transforming this data by using Node Types – whether out of the box, custom node types, or, as we'll see later in this chapter, from Coalesce Marketplace.

Adding Nodes to Your Pipeline

At the core of the developer experience in Coalesce is the ability to quickly add nodes to your data pipeline and start transforming data with ease. Nodes can be added directly from the platform or through Coalesce Marketplace. Because each node follows a standardized structure, the anatomy remains consistent across your pipeline, creating a predictable and unified experience for every developer. Coalesce includes several built-in Node Types to help you move faster right out of the gate:

- Stage
- Persistent Stage
- Dimension
- Fact
- View

You can right click on any of the source nodes and hover over Add Node to view these nodes as shown in Figure 3-8. Let's explore each of the node types below.

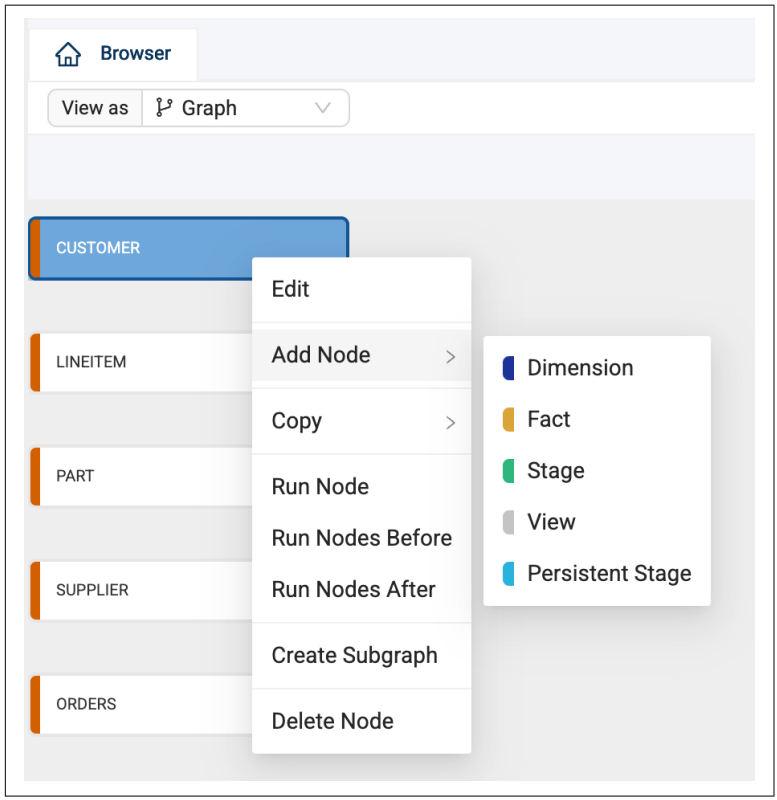


Figure 3-8. Node Types available to add to a data pipeline.

Stage

A Stage Node Type in Coalesce allows you to develop and deploy work in a table or view. It provides an intermediary working or staging layer to store, prepare, and transform raw data before down-

stream tables in your data pipeline use these data. This staging layer is a common data engineering practice for preparing your data. Just as a cook needs to prepare raw ingredients such as carrots and onions by slicing and dicing them before cooking, you can think of Stage Node Types as the preparation layer.

Stage Node Types are by default set to truncate the data in your table during each execution. This means that all data is deleted from the table and the fresh data coming from the upstream node will be processed into the Stage node. If the truncation is deactivated, the execution nature of the node would be to append data to the underlying table.

Persistent Stage

Similar to a Stage, a Persistent Stage is an intermediary Node Type that provides data persistence across execution cycles. Unlike a Stage Node Type, a Persistent Stage contains a business key allowing you to determine the unique identifier for your data source, which, in turn, allows you to persist or store historical data and load net new records into the object. This functionality is particularly beneficial when the objective is to retain historical data for prolonged durations.

Dimension

Coalesce supports Type 1 and Type 2 Slowly Changing Dimensions (SCD) out of the box. Each Dimension requires a business key, or unique identifier, to be defined. You then have the option of defining changing tracking columns, which will automatically configure the object as a Type 2 SCD when columns are selected. This functionality is particularly beneficial in tracking changes to the dimensions (data that describe your facts) in your data.

For example, you may want to know any time your customer's address changes when data is loaded from your sales data source. You can simply select the address column in your data source as a change tracking column, and Coalesce will automatically generate best practice Type 2 SCD Structured Query Language (SQL) within your data platform, saving you hours of developing time.

Fact

Fact Node Types provide you the ability to develop and deploy tables containing the measures or facts in your data platform. Each Fact Node Type contains a business key as well as functionality around varying operations for working with measures i.e. revenue, cost of goods sold, profit, etc. By using Fact Node Types, you can easily distinguish your Fact and Dimension nodes in the Browser when looking at your DAG.

View

By default, the View Node Type is turned off when using Coalesce for the first time. You can enable it by going to the Build Settings and selecting Node Types and turning on the Enable toggle for the View Node Type. This Node Type allows you to create objects as a view in your data platform, which means it is not storing any data, while also providing the flexibility to write custom SQL directly into the SQL editor.

While these five Node Types listed come out-of-the box, Coalesce is not limited to just these Node Types. As we learned in Chapter 2, you have the ability to create your own Node Types, and as we'll see next, use any of the Nodes from Coalesce Marketplace.

Coalesce Marketplace

Coalesce Marketplace provides a wide array of packages that help bring extensibility to your data pipeline projects. These packages are composed of one or more Node Types that help you solve specific problems or provide functionality to accelerate data development as seen in Figure 3-9.

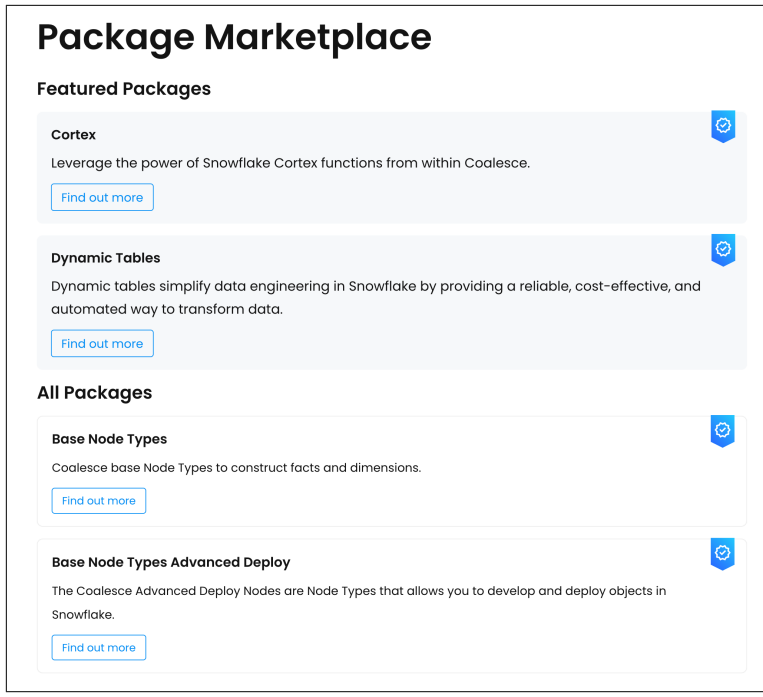


Figure 3-9. Packages available on Coalesce Marketplace that can be added to your Workspace.

Any package with the blue certified checkmark means that Coalesce has certified the package for production environments. Packages that don't include the checkmark are often developed by other engineers or organizations and are not guaranteed to work in all situations. You can see an example of this in Figure 3-10.

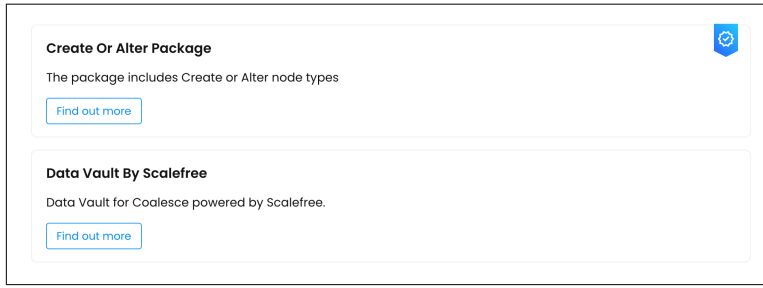


Figure 3-10. A Package that has been certified by Coalesce with the blue checkmark and a package that has not been certified.

You can easily install any of the Node Types from a package by selecting the "Find out more" button on any package. Within the package is a Package ID, and this package ID is how the package is installed within Coalesce.

When inside the Build Settings, you will see the Packages settings. Within the packages settings, you will see the option to either Browse or Install packages, as seen in Figure 3-11. The Browse button will open a new tab and take you to Coalesce Marketplace where you can view all of the packages available to install. The Install button allows you to install a package by providing the Package ID of the package from Coalesce Marketplace.

Figure 3-11. The Browse and Install buttons in the upper right corner of the Packages settings.

When installing a package, you will provide a package ID, version of the package, and an alias. By default, the package will install as the latest version, but you can manually change the version to any previously supported version. The alias of the package is the name or alias under which the Node Types will be displayed in the Browser, as seen in Figure 3-12 and 3-13. Each Node Type that is installed from the package will be displayed in the Node Type settings within the Build Settings of your Workspace. You can then add these Node Types to your pipeline just the same as any other Node Type.

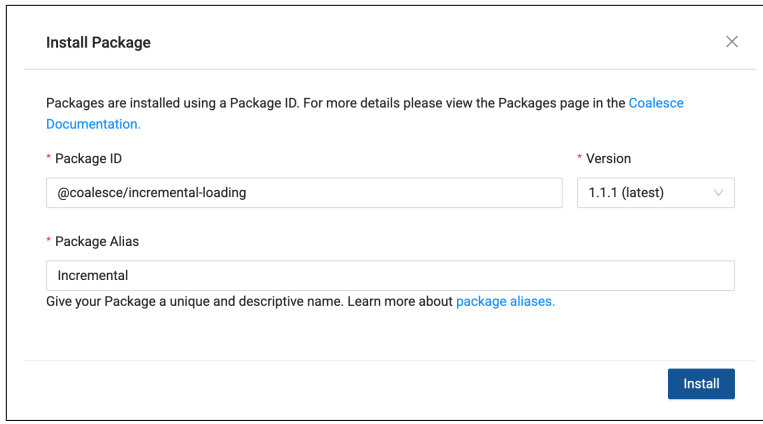


Figure 3-12. Providing an alias when installing a package.

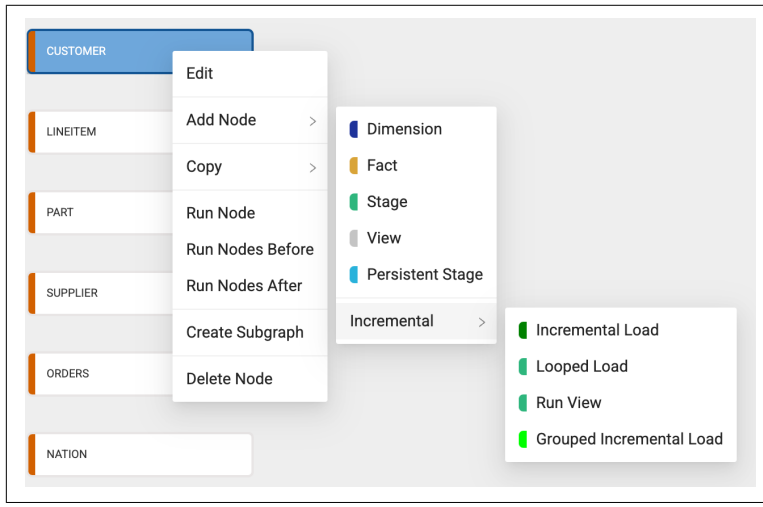


Figure 3-13. How the alias shows up when you add nodes in the Browser.

Whether they are out-of-the box, from Coalesce Marketplace, or custom, you'll use the same method to add all Node Types to your data pipeline. Let's learn how to begin building on our data sources using different Node Types.

Putting Node Types to Work

As we discussed earlier in this chapter, a common pattern of data engineering is to create a staging layer to prepare your data for the

needs of your business users. To do this in coalesce, we can use the Stage Node Type. You can select any node, including source nodes, in Coalesce and right click on the node and hover over Add Node to view all of the Node Types available to add to your data pipeline. In this case, select Stage.

Coalesce will add a green Stage Node Type to the Browser and will automatically add a prefix to the node: STG_, as seen in Figure-Image 3-14. This naming convention provides an additional way to easily see the Stage nodes in your Browser.

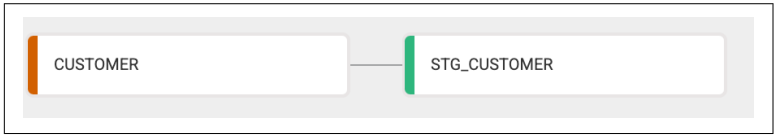


Figure 3-14. The Stage Node Type with the STG_ prefix applied to each node

As you learned about in Chapter 2, Coalesce uses data patterns to provide the templates making up each Node Type. Because each Node Type is a standardized object, you can add them to your pipeline the same way. This also means that you can add them in bulk to multiple objects, since the standard never changes.

In the Browser, you can see multiple nodes selected at once. By right clicking on any of the nodes selected, you can hover over Add Node and, to complete our staging layer, select Stage. Coalesce will automatically add a Stage Node Type to each data source as seen in Figure 3-15, allowing your team to immediately begin transforming your data, without having to configure the code of each object individually.

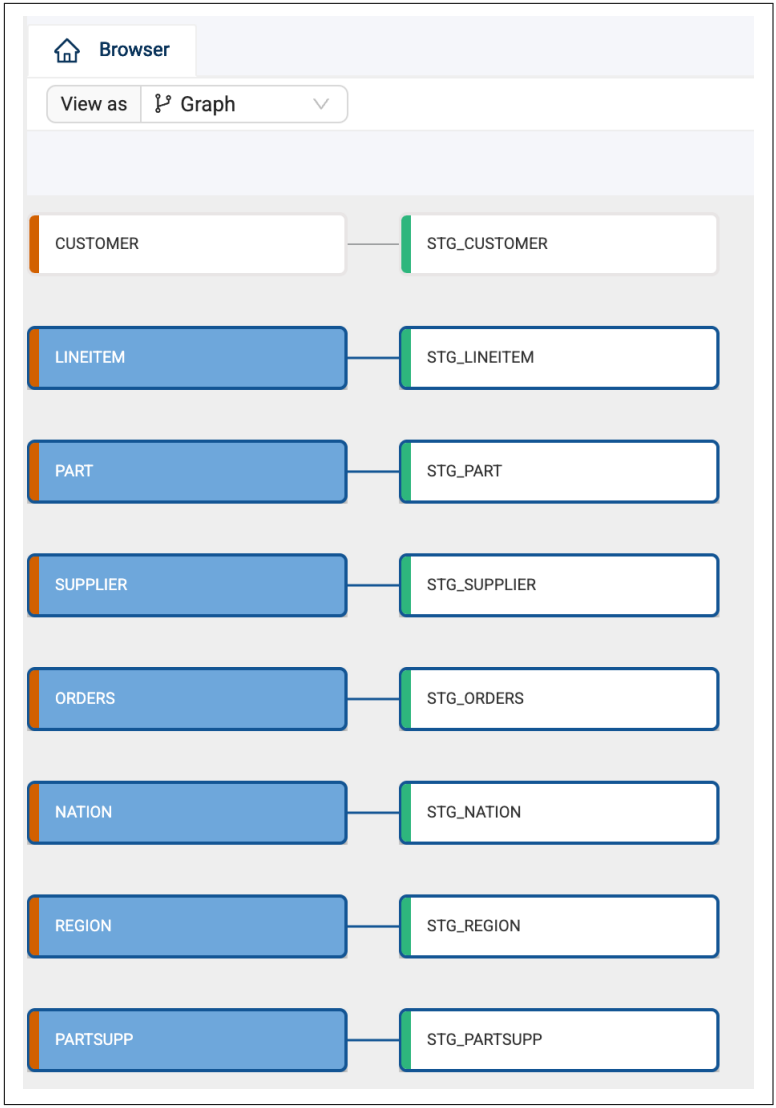


Figure 3-15. Bulk adding Stage nodes to the data pipeline

You can add any other Node Type available in your Workspace the same way that you added Stage Node Types. For instance, you can add a dimension node by right clicking on the STG_CUSTOMER node and selecting Dimension, as shown in Figure 3-16.

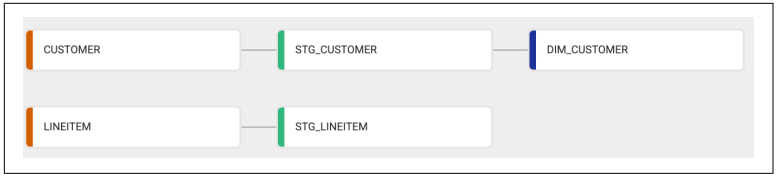


Figure 3-16. STG_CUSTOMER node with a dimension node as a dependency.

You can continue this process for every object needed to develop your data in your pipeline. As you add nodes to your pipeline, you will need to configure them to help you provide solutions to the problems you are solving. In order to do this, you will need to understand the anatomy of a node.

The Anatomy of a Node

In Chapter 2, we discussed how a Node Type is created (Node Definition, Create Template, Run Template). In this chapter, you will learn about the basic anatomy of a node, so you can effectively work with any Node Type. Whenever you double click on a Node in the Browser, it will open the Node Editor. While the configuration options (which you'll learn about shortly) may be different, the basic anatomy of each node is the same. Let's dive into this a bit deeper.

The Mapping Grid

When opening a node for the first time, you'll immediately see the mapping grid, as shown in Figure 3-17. The mapping grid is the display of all of the columns, including the name, data type, transformations, and even comments that are inherited from the parent node(s) in the DAG. You can easily add new columns, apply transformations, change data types, and apply a multitude of other operations in the mapping grid.

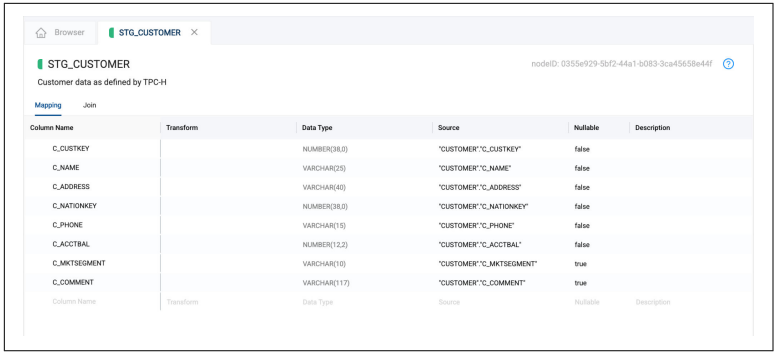


Figure 3-17. The mapping grid of the STG_CUSTOMER node.

Every node in your DAG will contain a mapping grid, even if it only includes a single column or line item, and is your way of knowing which columns you are working with.

The Configuration Options

Each Node Type contains configuration options that are unique to that Node Type. For example, the Dimension Node Type contains configuration options such as a Business Key and Change Tracking columns, which is different from a Stage Node Type which contains other configuration options. The configuration options of each Node Type allow users to accelerate their data development by reusing what has already been created as a standard. This is why configuration of a Type 2 SCD saves hours of time for data developers in Coalesce, because you can configure with just a few clicks, while automatically generating 100s of lines of SQL for you.

You can view the configuration options of any node in the Config tab in the upper right corner of the Node Editor as seen in Figure 3-18.

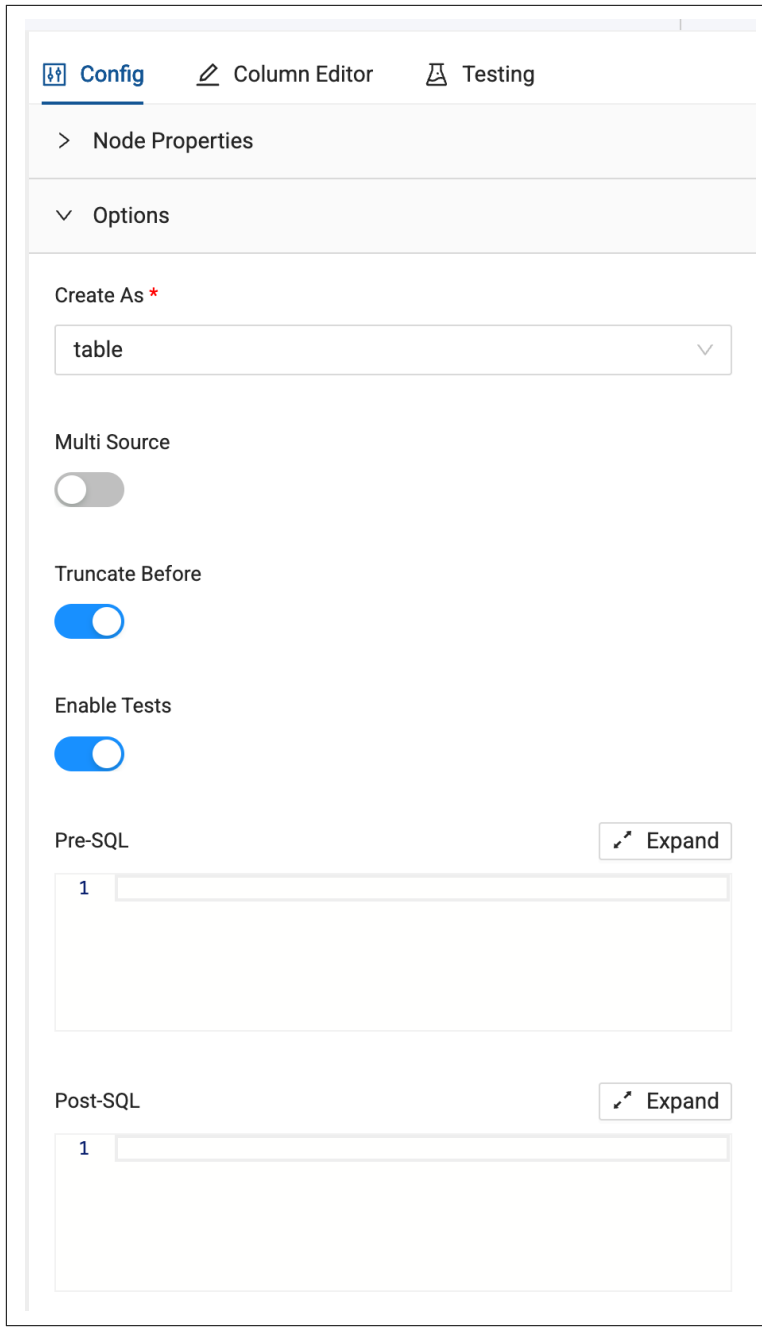


Figure 3-18. The configuration options of a node, providing the ability to configure a node without having to write code.

Create and Run

A critical part of developing data pipelines in Coalesce is the ability to create and run nodes. But what does that mean? When adding a node to your pipeline in Coalesce, the node is not immediately created in your data platform i.e. Snowflake. You as the data developer need to create the object in your data platform. The Create and Run buttons allow you to create objects and then execute any action on the object. Let's break this down in an example.

When working with a Stage node, you have the ability to materialize the object as either a table or a view. Let's assume you choose to build your object as a table. In order for the object to be created in your data platform, you need to select the Create button. In doing so, Coalesce will automatically take all of the metadata about the node, and automatically generate the Data Definition Language (DDL) for the object and create a blank object in your data platform. Coalesce will use the name of the node as the name of the object in your data platform as shown in Figures 3-19 and 3-20.

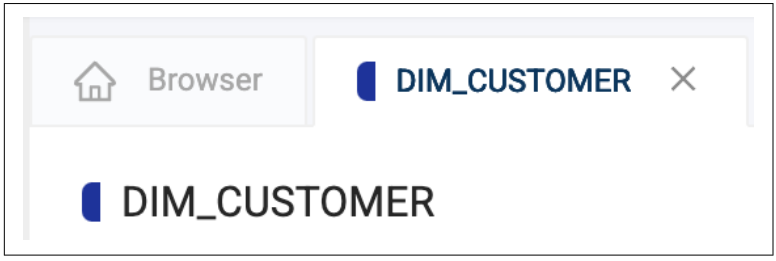


Figure 3-19. Name of the object in Coalesce.

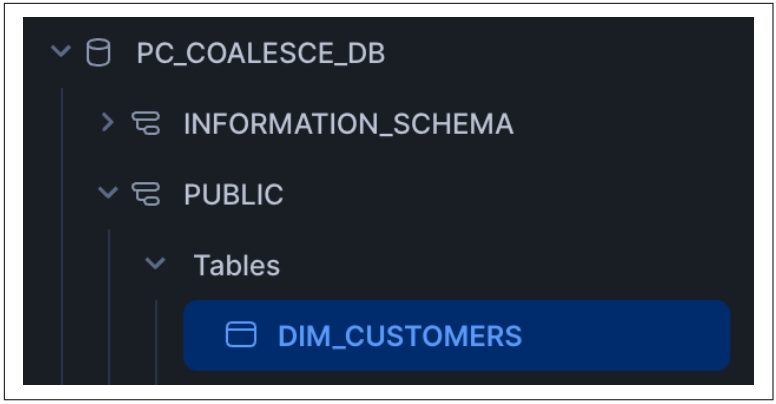


Figure 3-20. The same object created with the same name in your data platform.

Once the Stage is created as a table in your data platform, it is still empty. This is where the Run button comes in. By selecting Run within the node, Coalesce will again take all of the metadata provided from the node (column names, transformations, configurations, etc.), and automatically generate the Data Manipulation Language (DML) for the object, to insert data into the table. You'll be able to see the output of the operation in the Data Preview pane, which will display any data available after a successful run.

This is how Coalesce acts as the interface between your data platform and your raw data. Now you may be thinking that if you need to manually create and run each node, that is not a particularly sustainable approach to pipeline development, and you would be right! In Chapter 4, you will learn about deployments, and how you can automatically schedule your data pipelines to refresh, which effectively runs any of the Create and Run operations as needed.

The Join Tab

Each node type contains a Join Tab. This is located next to the mapping grid, as shown in Figure 3-21. Within the Join Tab, you will notice that there is a line of SQL already in the editor. The SQL shows a FROM statement with a REF function nestled within some curly braces. This line of SQL is creating the dependency to the upstream or parent node i.e. the REFerence to the parent.

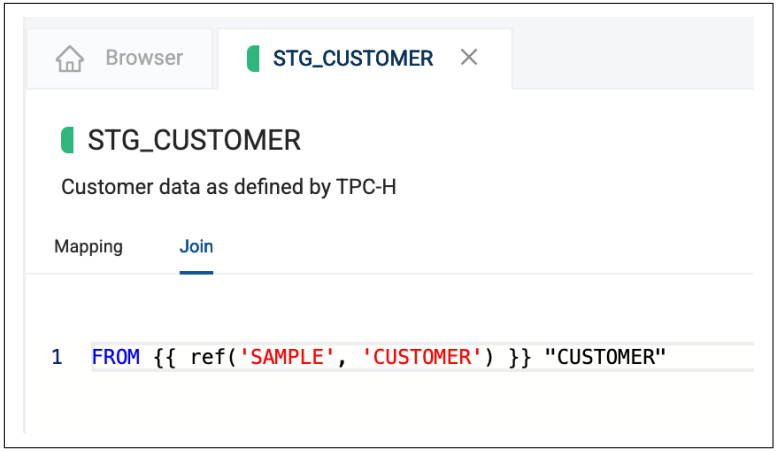


Figure 3-21. The Join Tab next to the mapping grid.

Within the Join Tab, you can perform multiple operations such as joins, filters, and even window functions. The Join Tab is often used for applying transformations to the object as a whole, rather than a singular column. We'll dig into the Join Tab more shortly, but for now, you only need to know where it is and what it can do.

You may notice a few other options within each node, and we'll get to those later in the chapter, but now it's time to get to the core of what nodes are doing, transforming data!

Data Transformations in Coalesce

In this section, I will provide a general overview of core data transformation functionality in Coalesce, but know that there are virtually an unlimited number of ways you can transform your data in Coalesce using various Node Types and settings. To cover the basics, I'll focus on column level and node level transformations.

Column Level Transformations

To kick things off, let's start with column level transformations. These are applied to individual columns within the mapping grid and are written using any valid SQL expression supported by your cloud platform. Let's look at an example.

In the node in Figure 3-22, we are applying an UPPER() function to a variable character column in order to apply consistent text casing across the field.

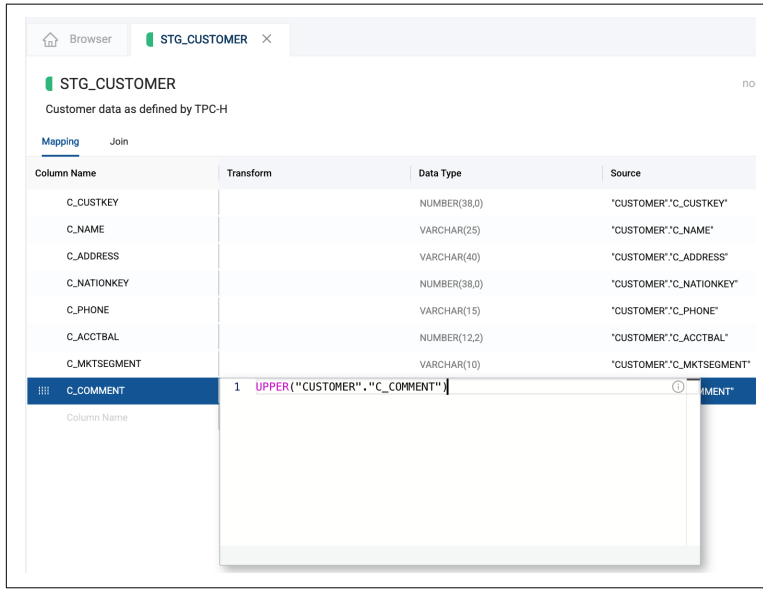


Figure 3-22. Applying a column level transformation to a column in the mapping grid.

While this is a relatively simple example of a column level transformation, you can probably begin to see the ease of management for writing transformations in this way, such as the column level transformations shown within Figure 3-23.

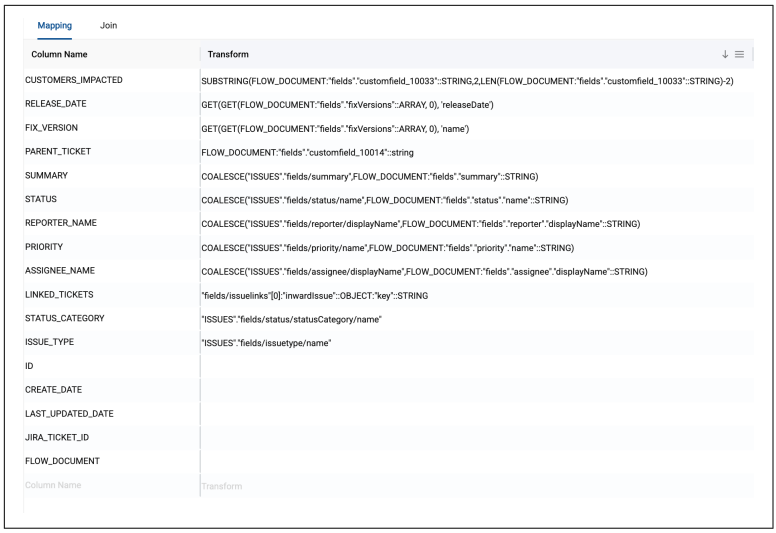


Figure 3-23. Multiple column transformations easily managed in the mapping grid.

Column level transformations can be quite complex, such as nested CASE WHEN statements that contain aggregate functions. Effectively, if you could write the column level transformation in your data platform, you'll be able to write it within Coalesce.

Each column level transformation will contain the name of the column being transformed, as well as the upstream dependency of that column. In this way, you are listing the full table and column reference to provide the metadata to generate the transformation code automatically for you.

It's important to note that column level transformations are only transforming singular columns. You will not be able to filter an entire table in a column level transformation, that is where you would use the Join Tab.

Node Level Transformations

Using the Join Tab, you can apply table or node level transformations to your data. Let's assume you've transformed all of your columns and are ready to apply any SQL necessary to support the logic of the table. For example, maybe you used an aggregate function and now need to supply a GROUP BY—you can supply this in the Join Tab. Maybe you're working with sales data for an

Ecommerce organization and only want to view sales where the order is delivered—you could apply the filter in the Join Tab. You can see examples of these in Figure 3-24. Effectively, if you need to apply any SQL at the table level, you should apply it in the Join Tab.

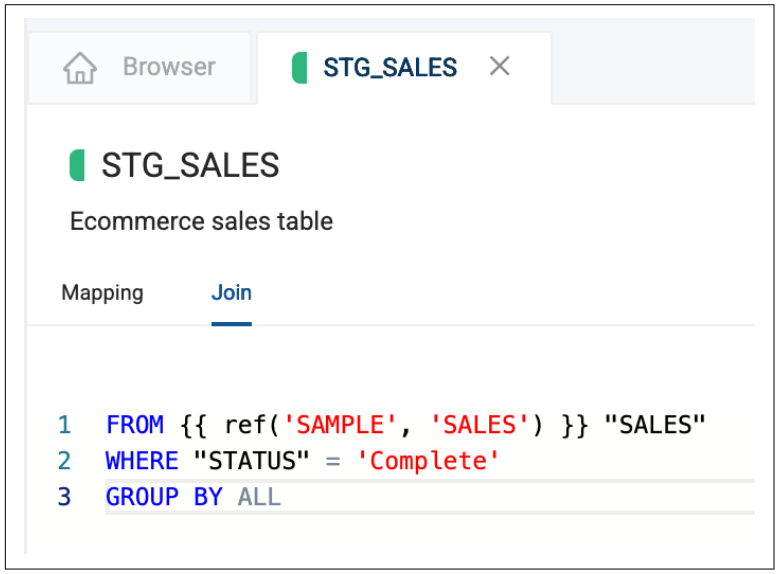


Figure 3-24. Writing SQL to configure the node at the node level.

Whether you are using column level transformations or transforming data at the node level, you can use any SQL supported by your data platform to do so. There is no proprietary syntax limiting your ability to write SQL or use a different language, you can just plug and play. Now that you're familiar with the Join tab, let's explore its namesake–the join itself!

Joins

Up until this point, we have been working with and transforming data in single nodes. But what happens when you want to join multiple nodes together? Coalesce makes this just as easy as adding a node to your data pipeline.

In the Browser, you can select two or more nodes you want to join together. Once selected, you can right click on either node and hover over Join Nodes, and you will see all of the options available for adding a node, but now available as a joined node. In

Figure 3-25, the STG_ORDERS and STG_LINEITEM nodes can be joined together as a new Stage Node Type. Figure 3-26 shows the result of a join in the Browser

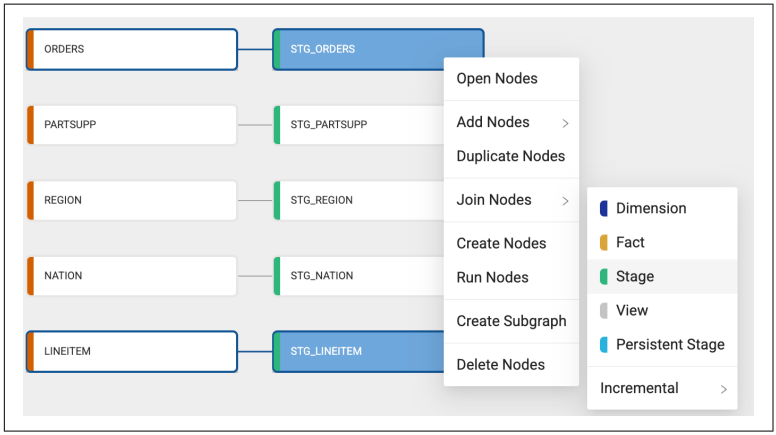


Figure 3-25. Applying a join using the Join Nodes option in the Browser of Coalesce.

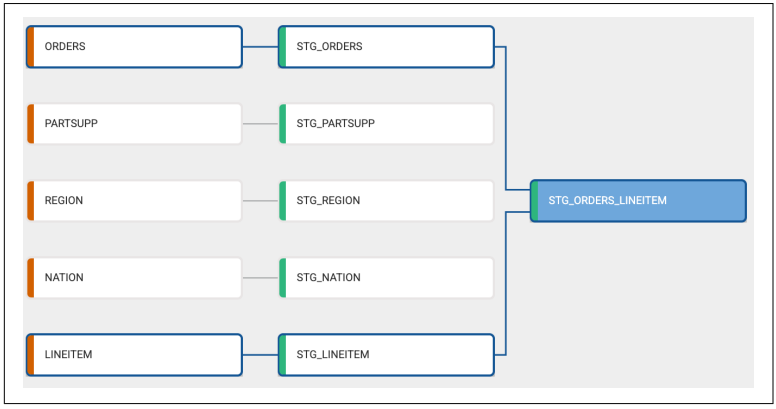


Figure 3-26. The result of the Join Nodes option in the Browser.

Once the joined Stage node is selected, Coalesce will automatically drop you into the Node Editor. To configure the join, navigate to the Join Tab. You will see a join statement that Coalesce has automatically generated for you. In most cases, all you need to do is provide the join condition i.e. the column names used to join the tables together. This can be done by removing the placeholders shown in Figure 3-27 and replacing them with the column names needed to configure the Join as shown in Figure 3-28.

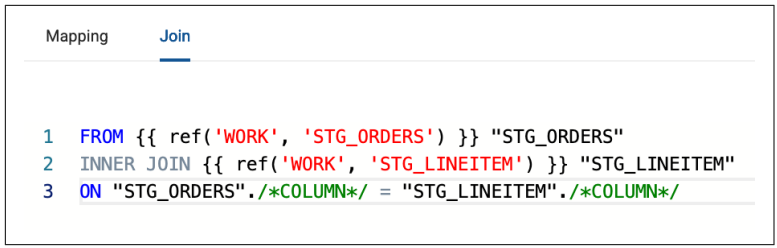


Figure 3-27. Column placeholders in the Join Tab for join conditions.

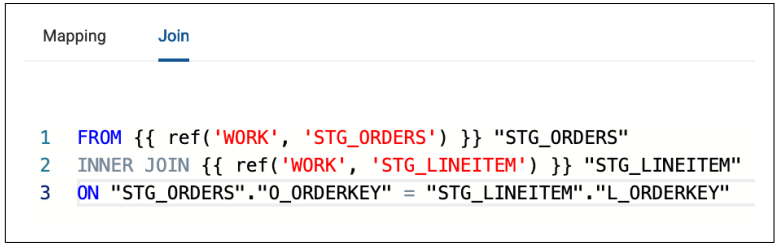


Figure 3-28. Configuration of the join in the table with the proper column condition

As you learned earlier, when it comes to table level transformations, we could always add another join condition if necessary as shown in Figure 3-29.

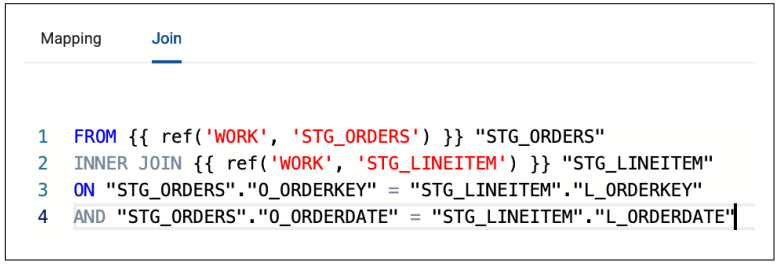


Figure 3-29. Adding another join condition showing flexibility of SQL editor.

Coalesce automatically infers when a join is occurring. If you happen to delete or break your code, you can always have Coalesce regenerate the join. In the upper right corner of the Join Tab SQL editor, you'll see the Generate Join dropdown, as shown in Figure 3-30, which allows you to regenerate the code back in the SQL editor, or copy directly to your clipboard.

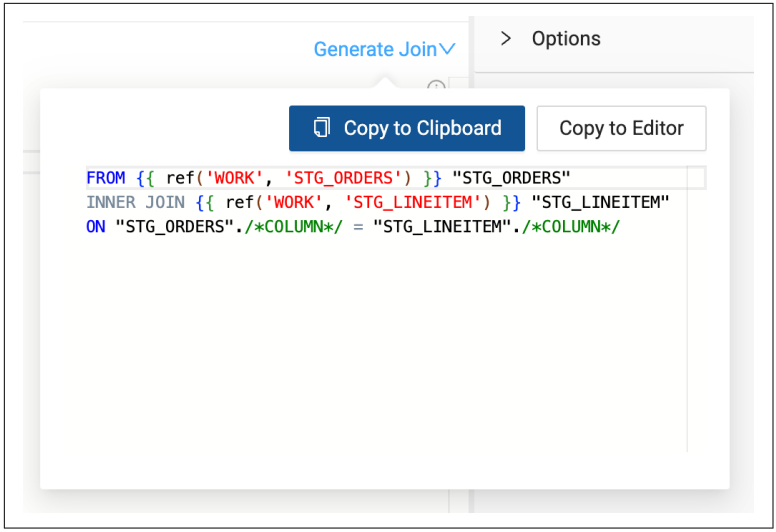


Figure 3-30. The automatic join generation that Coalesce provides in each node.

This same process can be used to join any number of nodes together, making it simple and easy to manage even the most complex joins.

Once tables have been joined, there are often many columns that need additional work—whether it is renaming, changing data types, or applying transformations. In Coalesce, you can handle these tasks efficiently using the bulk editor, which you will learn about next.

Bulk Editing

Joins can cause the need for you to deal with multiple columns at once. In traditional data development scenarios, you may be required to manually delete columns one by one, or apply transformations to one column at a time. But because Coalesce uses data patterns and metadata to accelerate your data development, you can bulk edit columns straight from the mapping grid.

Let's assume you've joined the two nodes together from our previous example. You now have multiple columns containing variable character data types. In the case where you want to apply consistent text casing like our example from earlier, you can bulk add this transformation to each column you select, and only write the transformation one time.

If you were to select all of the VARCHAR columns in the node and right click on any of the columns, you could select Bulk Edit, as shown in Figure 3-31. This would open the Column Editor next to the configuration options you learned about at the beginning of the chapter.

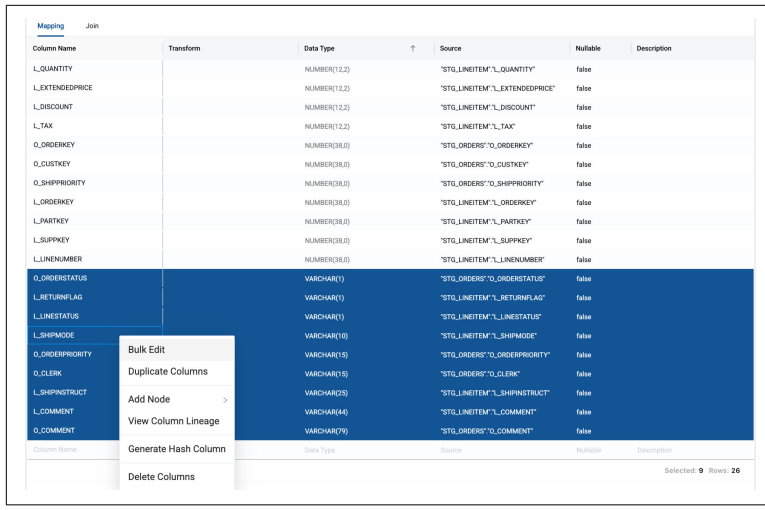


Figure 3-31. FigureImage 3-31. Selecting columns in bulk in order to bulk edit them.

You could then select the attribute you wanted to transform, such as the column names, data types, or transformations. For our example, we'll choose transformation and write a simple UPPER({{SRC}}) function that can be applied to each column as shown in Figure 3-32. You'll learn about the {{SRC}} token in Chapter 4, but for now know that it automatically resolves the column in any transformation!

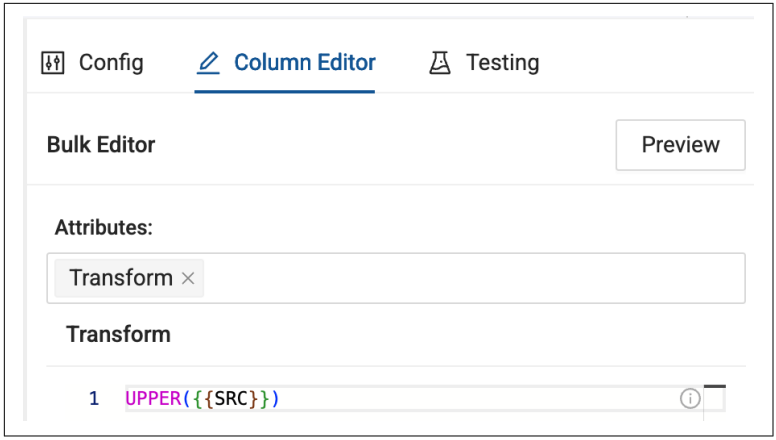


Figure 3-32. The bulk column editor, applying the UPPER function to multiple columns as a transformation.

An attribute of a column can be bulk transformed in this way. Additionally, if we were to want to remove multiple columns, we could easily select all of the columns we wanted to delete, right click on any of the selected columns, and select Delete Columns, as shown in Figure 3-33.

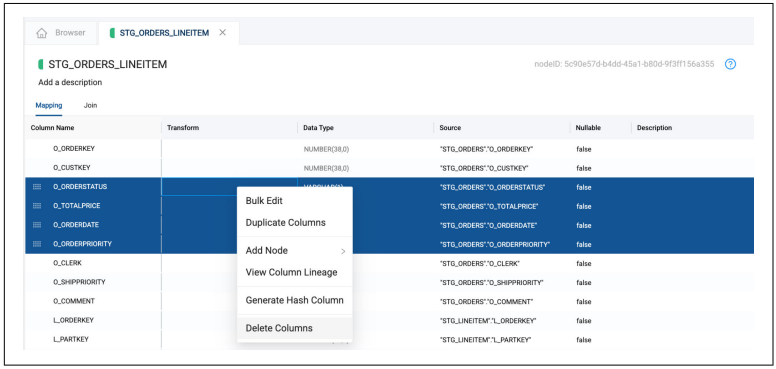


Figure 3-33. Bulk deleting columns from the mapping grid.

By allowing users to work with columns in bulk, Coalesce makes it easy to update multiple fields at once—saving time and letting developers focus on higher-value work.

Data Transformation in Process

There was quite a lot of information to digest in this chapter. You learned about the Build Interface and all of the components it contains. You learned all about adding data sources and how to add Nodes to your pipeline, including the many different kinds of Node Types available to you. You also learned about the anatomy of a node and how to use SQL to transform data inside of any node, while also joining nodes together and applying bulk updates. Whew!

Coalesce is no less powerful because it leads with a user interface. In fact, it enhances efficiency by standardizing the way work is done, ensuring everyone builds from the same foundation. And when you need to write SQL, it is always available. A visual interface does not limit capability—it amplifies it.

In the next chapter, you'll take your new data transformation skills and sharpen them by learning how to manage the data pipelines you build in Coalesce. You're quickly becoming a data developing master.

CHAPTER 4

Managing Data Pipelines in Coalesce

A Note for Early Release Readers

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 4th chapter of the final book.

If you'd like to be actively involved in reviewing and commenting on this draft, please reach out to the editor at *mpotter@oreilly.com*.

In the previous chapter, you used Coalesce to build data transformations and pipelines. But managing existing pipelines is just as crucial as creating new ones. This chapter covers both: building any data product in Coalesce and how to manage pipelines of any size effectively.

You'll learn how to inspect objects to understand their contents and where they are built. You'll use column-level lineage to perform impact analysis and leverage the problem scanner to identify and resolve issues efficiently. This chapter also explores key management features, including version control, macros and parameters, testing, subgraphs, jobs, and environments.

First, you'll navigate the nodes in your pipeline to establish a solid foundation for managing your data workflows.

Managing Nodes Using Views

As you build nodes in Coalesce, obtaining full context about them can be challenging but essential for development decisions. Coalesce makes this process straightforward. By default, it displays pipeline development as a graph, which is shown in Figure 4-1.

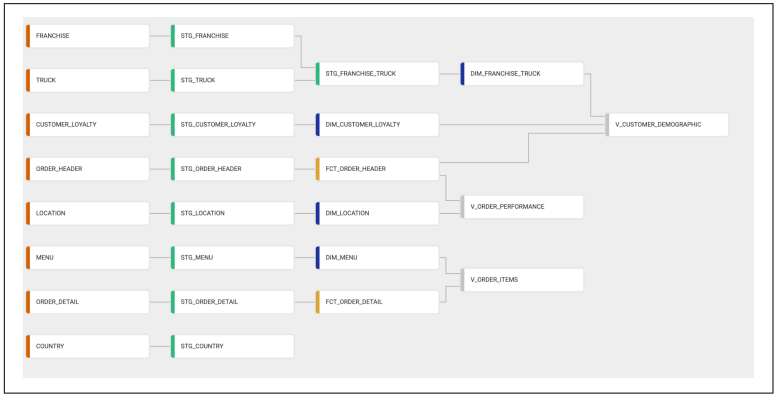


Figure 4-1. Image of multiple node types creating a graph in Coalesce.

Additionally, Coalesce provides two more views: the node grid and the column grid. These views offer deeper insights into your pipeline. Let's explore all three views.

The Graph

The graph view provides the best way to see your entire pipeline and its dependencies. It constructs a Directed Acyclic Graph (DAG) that displays the connections and dependencies of each node. This view gives a comprehensive understanding of how data moves through the pipeline. When troubleshooting, the graph view helps quickly identify dependencies and determine how far upstream or downstream a specific object is. It also makes it easy to distinguish nodes by their unique colors, as shown in Figure 4-2.

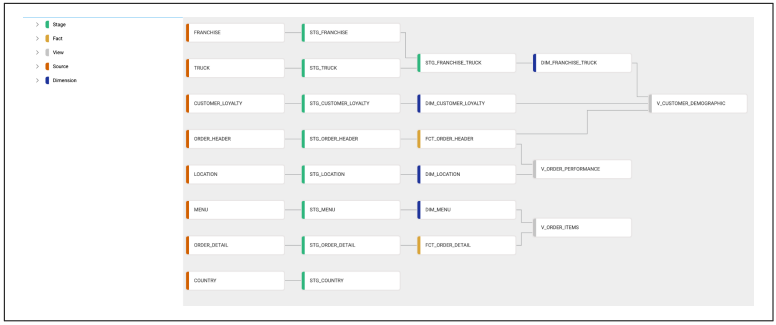


Figure 4-2. The node menu rolls up all of the nodes to the respective color of the node type.

The Node Grid

When developing data products in Coalesce, finding a specific node or object in the graph can be challenging, especially in projects with hundreds or thousands of nodes. For large projects, you can use subgraphs to help with this, but you'll learn more about that later in this chapter.

The node grid provides several ways to filter and locate nodes efficiently. It allows filtering by individual columns, so you can display only nodes of a specific type, such as Fact nodes. Additionally, the filter bar supports selector queries, which you'll learn about in the next section, enabling you to include or exclude values from your search to refine the displayed nodes. For example, if you are processing orders data with some stage nodes, you can use the filter bar and selector syntax to search for nodes that follow a STG_ORDER naming convention, as shown in Figure 4-3.

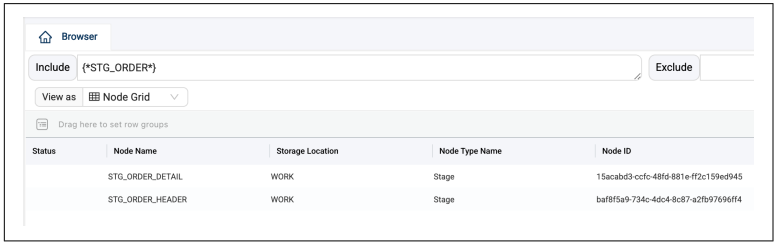


Figure 4-3. Using an Include filter to filter the Node Grid for only tables with STG_ORDER in the name.

You can also group nodes in the mapping grid by their associated storage location. To do this, drag and drop the Storage Location

column into the row groups area. This immediately organizes nodes into groups based on their storage location. You can further refine these groups using column filters and selector queries.

The Column Grid

Column-level architecture, which you have already explored in previous chapters, makes it easy to manage columns in Coalesce. A primary way to do this is through the column grid, which provides a detailed view of every column in your workspace in one place. This view offers several advantages.

First, the column grid displays all columns in a single location. Instead of being scattered across isolated documentation or tied to specific objects, this view provides complete transparency. It allows data practitioners to see all columns in a workspace without searching through individual objects or fragmented documentation.

Second, the column grid supports the same filtering and selector queries as the node grid. This makes it easy to refine searches and locate specific columns. For example, you can apply a filter to find any column containing a specific name, as shown in Figure 4-4.

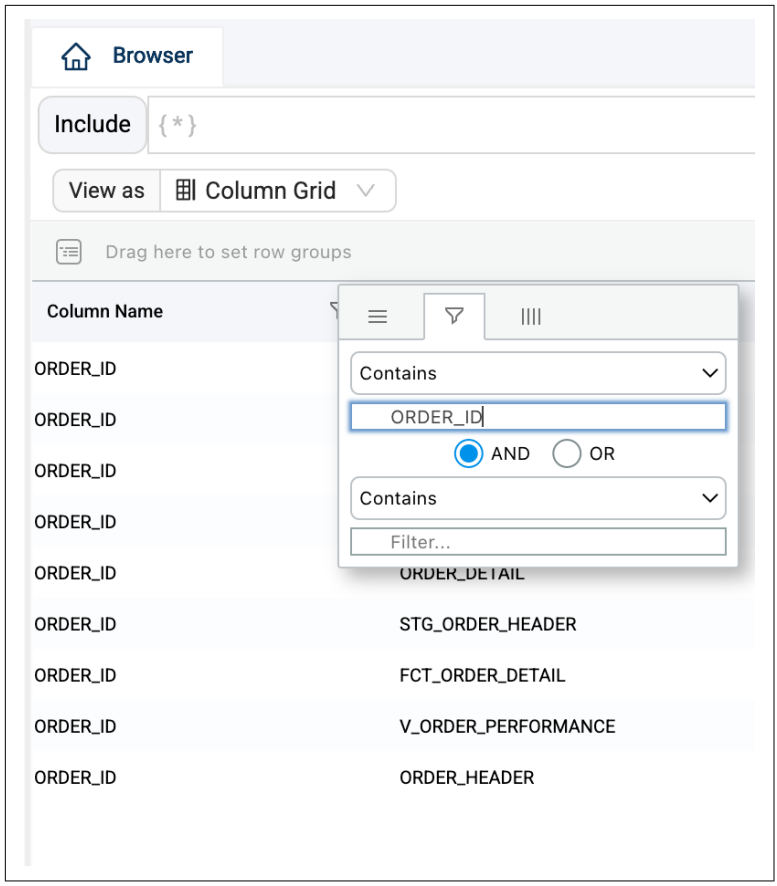


Figure 4-4. Using a column header filter in the Column Grid to filter for ORDER_ID.

Like the node grid, the column grid allows you to create groupings using column headers. For example, you can group by Storage Location and Node Name to refine searches for columns within a specific location or object in your data platform.

You can also use selector queries or column filtering to view and manage transformations applied to any column. This provides a powerful interface for handling column-level transformations. For example, you can search for a specific column name and review all associated transformations. This is especially useful when updating a transformation across multiple objects.

For instance, let's assume you use a transformation to extract the area code from phone number columns in various datasets. This transformation relies on a LEFT function to capture the first three digits of the phone number. However, after an application update, phone numbers now include a country code (such as +1 for the United States). As a result, the existing transformation no longer works correctly because it captures the country code along with the first two digits of the area code.

To update this transformation, you can filter the mapping grid to locate the specific transformation. Then, using the Bulk Editor, you can update every affected column at once by modifying the function and applying the changes with a single click, as shown in Figure 4-5.

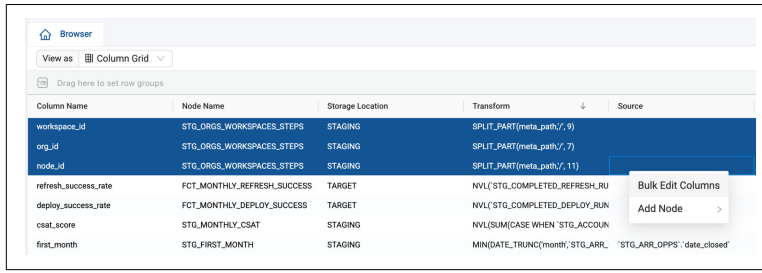


Figure 4-5. Bulk editing columns after identifying the transformations in the Column Grid.

Each view in Coalesce offers distinct ways to manage objects and columns in your data pipeline. These views equip you with the tools needed to efficiently navigate and control your pipelines. However, Coalesce provides additional functionality that extends beyond node visualization, allowing for even greater project management control.

Subgraphs

So far, you have explored how to use Coalesce pipeline views to manage objects in your pipeline. As your workspace grows to hundreds or thousands of nodes, viewing everything in the graph can become challenging. Subgraphs help manage this complexity by allowing you to isolate parts of the node graph.

Subgraphs break down defined sections of your pipeline into logically separate sections. They can be defined manually or by using a Selector query. For example, suppose you are working on an enterprise data warehouse in Coalesce with dozens of data sources.

Your graph has grown to over 500 nodes, making it difficult to locate nodes associated with a specific data source, such as customer relationship management (CRM) data. To address this, you can organize each data source and its processing nodes into separate subgraphs.

To create a subgraph, select all nodes related to your CRM data that you want to associate with the subgraph. Then, in the Subgraphs tab, click the + button and select Create Subgraph. Coalesce will generate a subgraph that displays only the nodes associated with your CRM data, as shown in Figure 4-6.

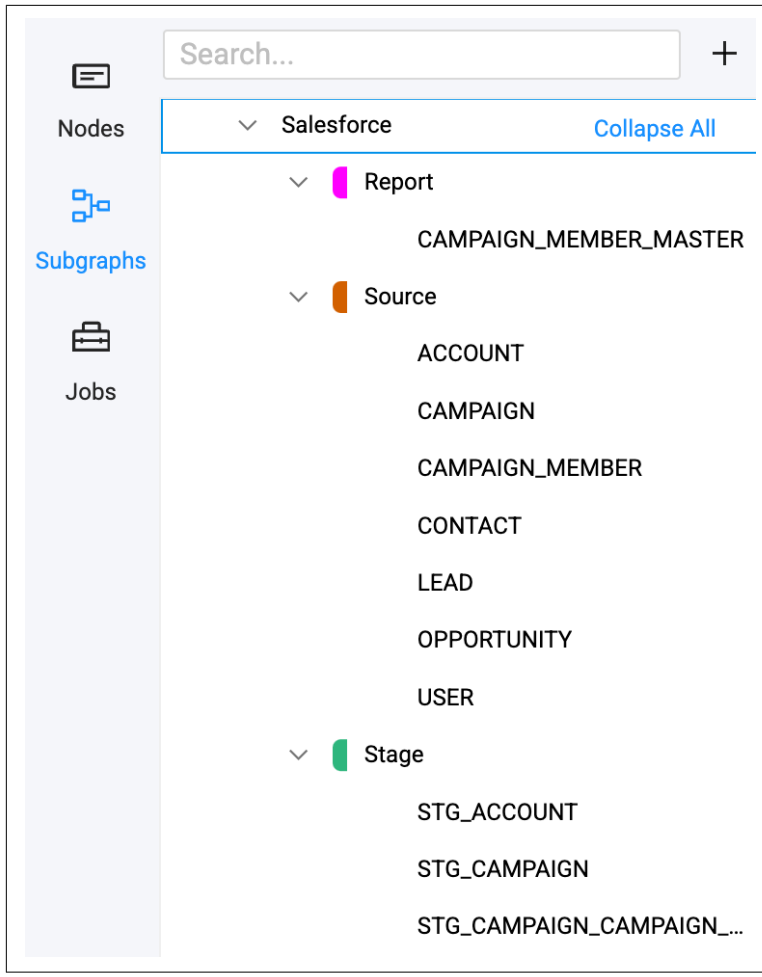


Figure 4-6. Subgraph created to contain only data from a CRM (Salesforce).

Additionally, just as you can use selector syntax for viewing nodes, you can use it to select nodes to add to a subgraph as shown in Figure 4-7.

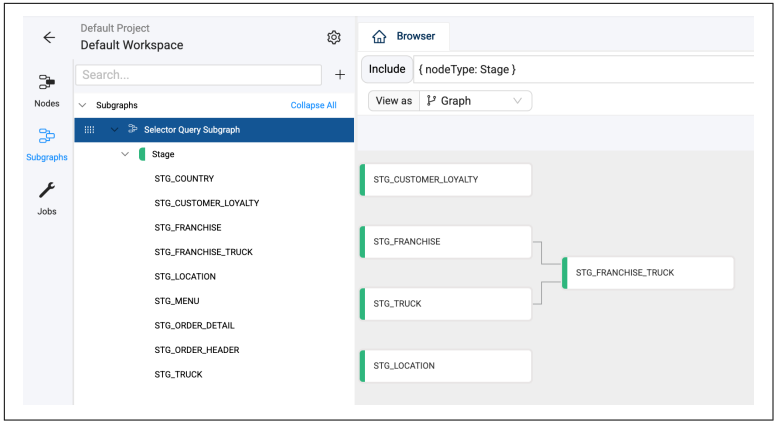


Figure 4-7. Using Selector Syntax to select the nodes to add to a subgraph.

Up to this point, you have seen multiple references to selector syntax, but you have not yet explored what it is. Now that you know where you can use selector syntax, let's take a closer look at what it is and how to apply it.

Selector Queries

Selector queries, refine and enhance search criteria throughout various functions in Coalesce. Each selector query follows a structured search syntax with attributes that allow you to retrieve specific information.

For example, to filter your pipeline for only Stage nodes, you can use the nodeType attribute in the following query:

```
{ nodeType: Stage }
```

This query filters the workspace to display only Stage nodes. You can apply the same approach to any other node type.

Coalesce supports multiple attributes, which are documented in the Coalesce documentation, to help refine and enhance queries. You can combine multiple attributes within a single query. For example:

```
{name: SUPPLIER* location: BI nodeType: Dimension }
```

This query searches for:

- Any node name that contains SUPPLIER, using the asterisk (*) as a wildcard.
- Nodes built within the BI Storage Location.
- Only Dimension nodes that meet these criteria.

Operators further enhance selector queries by providing more control over searches. For example, the plus (+) operator selects predecessors or successors based on placement:

```
+{name:STG_SUPPLIER}
{name:STG_SUPPLIER}+
```

You can also use the AND and OR operators to add conditional logic:

```
**{name:STG_} OR {name:REGION}
```

Selector queries allow precise filtering of pipelines, giving you more control over how your data is managed. You will revisit selector queries when learning about jobs later in this chapter. But for now, you should feel comfortable using them to filter views and create subgraphs. Selector queries aren't the only way to be able to refine your search within your data pipeline. Coalesce provides column level lineage to allow you to understand how each column flows through your pipeline, which you'll learn about next.

Column Level Lineage

In chapter 2 of this guide, you explored column-level architecture in Coalesce. Since Coalesce is built with column-level architecture from the ground up, you can take advantage of powerful features that streamline data pipeline management and accelerate development. One of the most powerful components that results from this is column-level lineage.

Column-level lineage in Coalesce allows you to track each column as it moves through the pipeline. This includes transformed columns, showing how multiple columns combine into one or how a single column splits into multiple columns. Leveraging its column-level architecture, Coalesce also enables you to propagate column additions and deletions to downstream nodes with a single click.

To view column-level lineage, double-click into any node, select one or more columns, right-click the selection, and choose View

Column Lineage. Coalesce will open the column lineage view, highlighting the selected column(s) and displaying their flow throughout the entire pipeline, as shown in Figure 4-8.

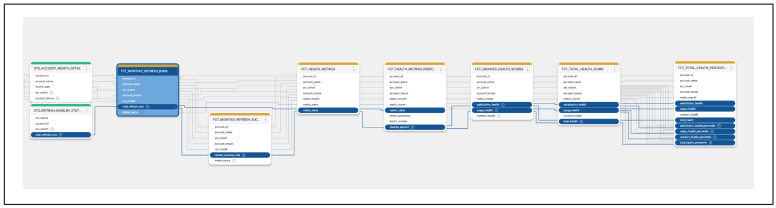


Figure 4-8. Column level lineage of how columns flow through a pipeline and are transformed.

This context enables you to perform quick and precise impact analysis, allowing you to see exactly how changes will affect your pipeline. Additionally, any column with a transformation can display its transformation within column-level lineage, providing further insight into how modifications will impact your pipeline, as shown in Figure 4-9.

Figure 4-9. Column level lineage displaying the transformation involved in the column.

For example, suppose your ETL solution adds a new column that needs to be exposed in a dashboard for stakeholders. In traditional pipeline development, this process can be tedious, requiring you to open every object the column passes through, run new DDL scripts, and potentially perform complex backfilling operations. In Coalesce, you can propagate the new column to all necessary nodes in just a few clicks.

In this scenario, the pipeline is undergoing a schema change. Let's assume the pipeline processes data through a staging layer before exposing it to downstream nodes. You can rename or transform the new column in the staging layer and then view its column lineage. Within the column lineage view, you can navigate to the selector dots next to the column name and select Propagate Addition, as shown in Figure 4-10.

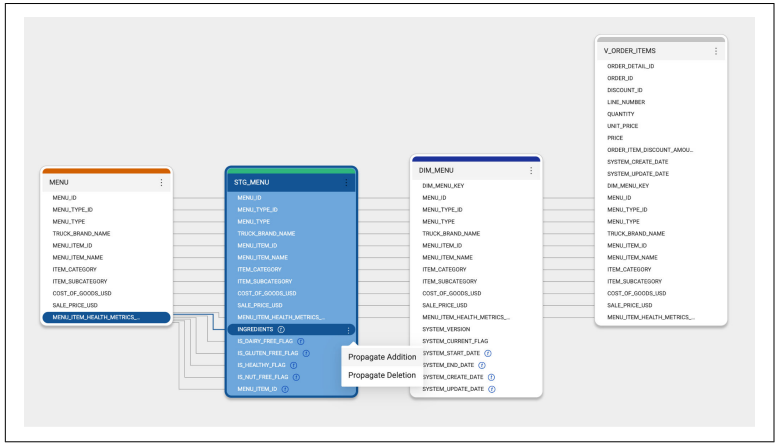


Figure 4-10. Preparing to propagate an addition of a column to downstream nodes.

After selecting Propagate Addition, you can choose which downstream nodes should receive the new column. Check the box next to each node name, then click Preview and Apply to finalize the change, as shown in Figure 4-11.

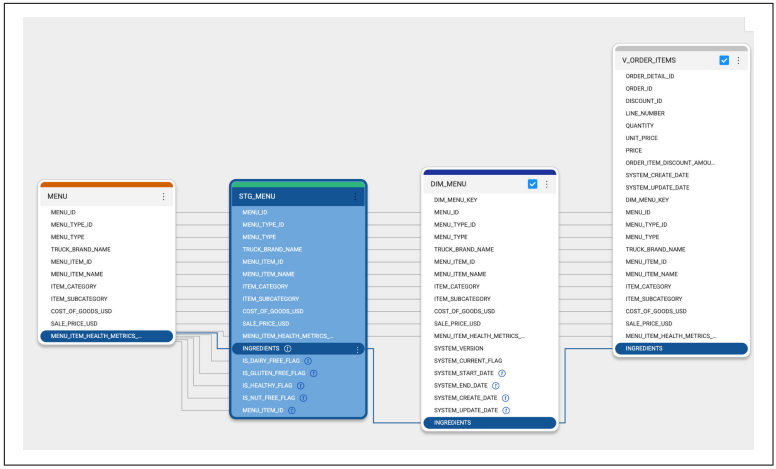


Figure 4-11. Persisting the column to the downstream nodes.

It's really that simple. Coalesce handles all necessary DDL changes, updates each selected node with the new column, and allows you to process data as needed within your nodes.

Column-level lineage provides a clear view of how each column moves through your data pipelines. Additionally, it also enables you to propagate changes with just a few clicks, reducing errors and accelerating development.

The Problem Scanner

So far, you have seen how to manage pipelines when you know what needs to change, such as updating a transformation for a column change. But what about unexpected changes? A column update might have unintended consequences. Removing a column from a node could break dependencies without you realizing it. How do you catch these issues before they cause problems?

The Problem Scanner is a critical feature in Coalesce that alerts you to potential issues in your workspace. It detects broken dependencies, column breakages due to renaming or dependency changes, and misconfigured systems such as Git or your data platform.

Using the Problem Scanner to identify issues before they impact a production environment is essential for maintaining stable pipelines. This ensures business users always receive accurate and reliable data. The problem scanner automatically scans your entire

pipeline in real-time and you can always access it in the lower left corner of the build interface.

A common alert from the Problem Scanner occurs when column dependencies are broken. As shown in Figure 4-12, Coalesce highlights these breakages within the Problem Scanner. You can select any listed issue to navigate directly to the affected node and resolve the problem.

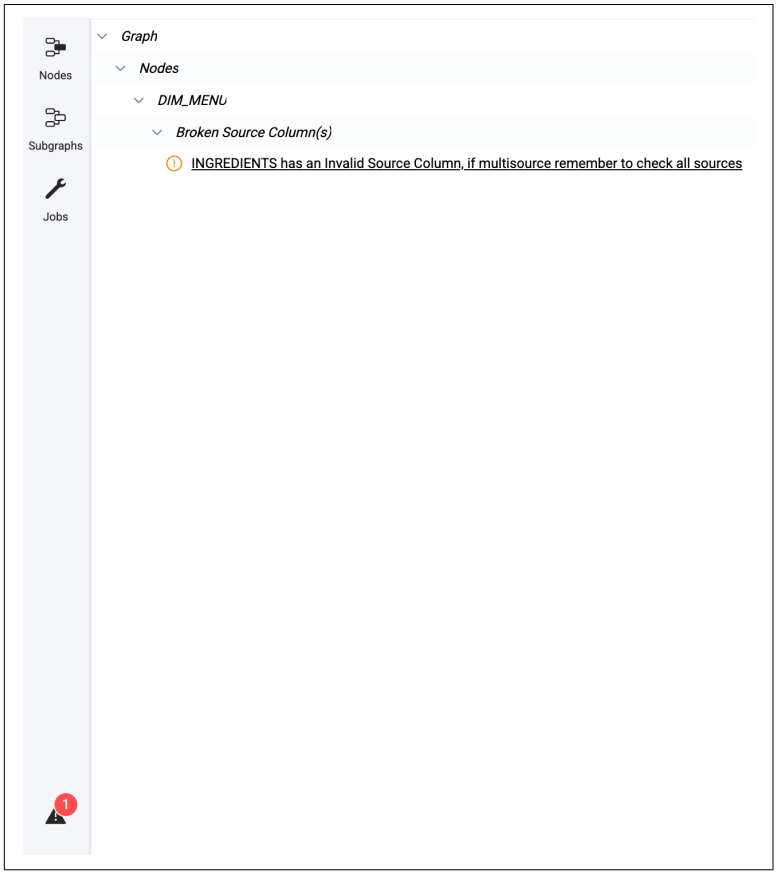


Figure 4-12. The Problem Scanner identifying a broken dependency.

You can then apply the skills learned in the previous chapter to transform or bulk edit the issues identified by the Problem Scanner within the node. While appearing as a subtle feature, the Problem Scanner helps prevent confusion by identifying breaking changes

before they become critical issues, helping you spend more time on building data products.

Version Control

Version control is a critical part of any data development project. Coalesce provides robust version control, enabling a continuous integration and continuous development (CI/CD) process through your preferred Git provider. In this guide, we assume you are using GitHub.

In Chapter 1, you learned that Coalesce natively integrates with Git. Now, it is time to understand how version control works within Coalesce. Once Git is integrated, you can create a project and a workspace. The project links to the Git repository of your choice, while the workspace is associated with a branch from that repository, as shown in Figure 4-13.

Figure 4-13. Workspace with a Git Branch tied to it.

With your repository and branch configured, you can launch your workspace and begin building data products as you've learned to do in the past two chapters. As you build, you will notice the version control icon in the navigation bar on the left side of the screen towards the lower left corner as shown in Figure 4-14.

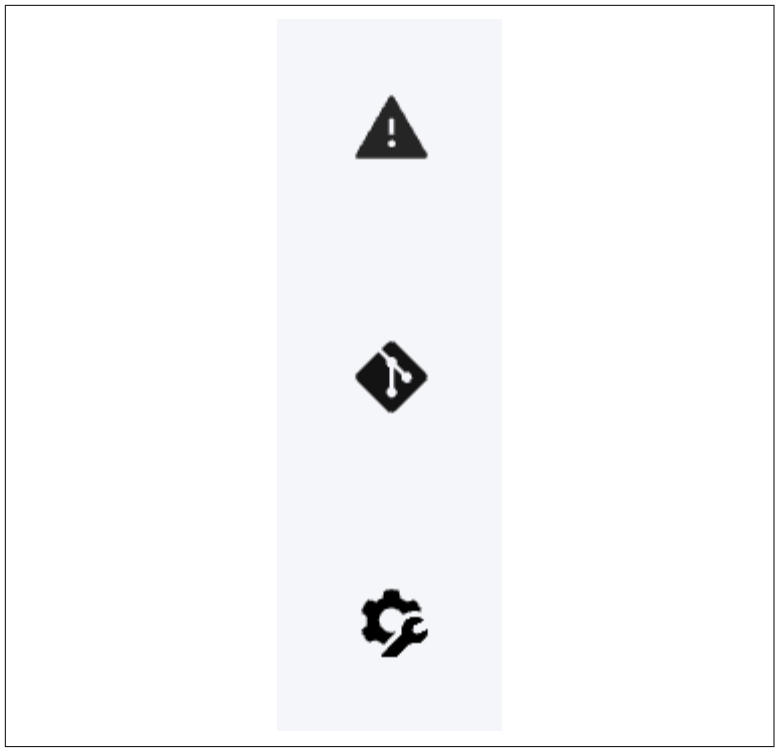


Figure 4-14. The Git Icon in the middle of the navigation bar.

By clicking on the version control icon, the Coalesce git modal will open and you will be able to view all of the work you've performed in your workspace since your last commit. If you've never performed a commit, the modal will display all of your work, broken down into the categories they are associated with, such as Nodes, Subgraphs, and Packages, as shown in Figure 4-15.

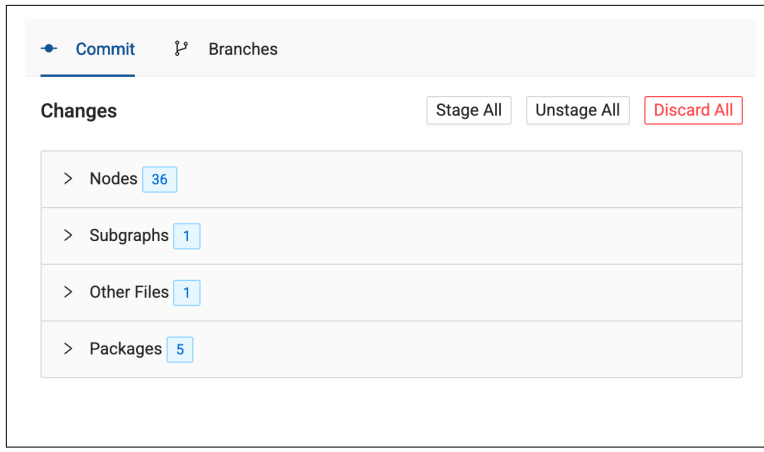


Figure 4-15. Coalesce categories of files ready for version control.

When you open any version-controlled file, you can view a file differential (diff). For new files, the diff will be blank. For recently changed files, Coalesce highlights modifications using red and green lines, as shown in Figure 4-16. In Coalesce, all work is saved as YAML files, so diffs are displayed in YAML format.

Figure 4-16. Coalesce highlighting the file diffs in the Git Modal.

Once you have reviewed your diffs and have approved your work, you can choose to commit partial amounts, or all of your work at once. Within the Git modal, by default, every file is selected to be included in a commit. You can deselect any of the files you wish to

exclude from a commit, by selecting the blue checkbox next to the file name as seen in Figure 4-17.

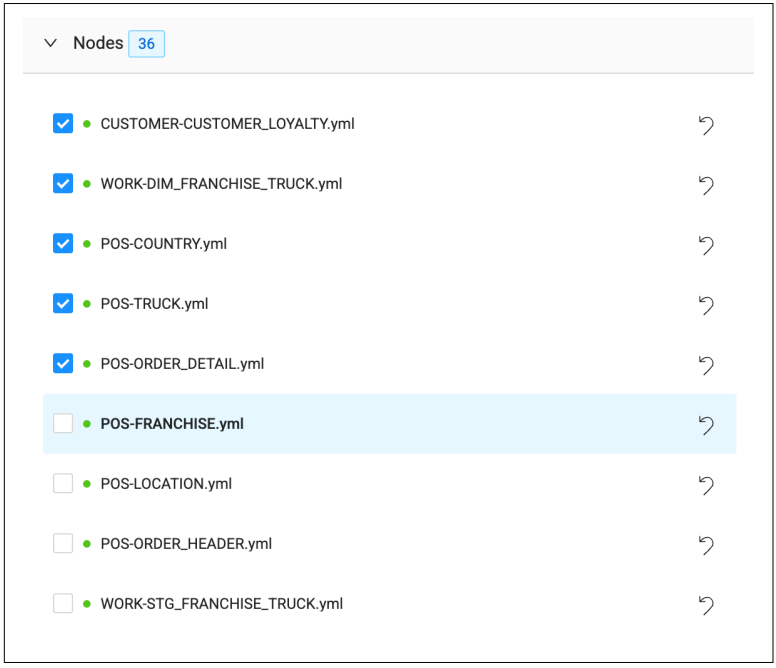


Figure 4-17. Selecting only certain files to be committed.

After selecting the files to commit, you can write a commit message or use the Coalesce AI commit message generator to generate one based on your changes. Once the commit message is in place and the files are selected, click Commit and Push to commit your changes and push them directly to your version control system from Coalesce.

So far, you have explored the Commit section within the Git modal. However, you can also checkout, merge, or create a new branch from any commit within the modal. To do this, select the Branches tab next to the Commit tab, as shown in Figure 4-18. The Branches tab displays all commits associated with the branch you are working in. Coalesce also highlights the commit currently associated with your workspace.

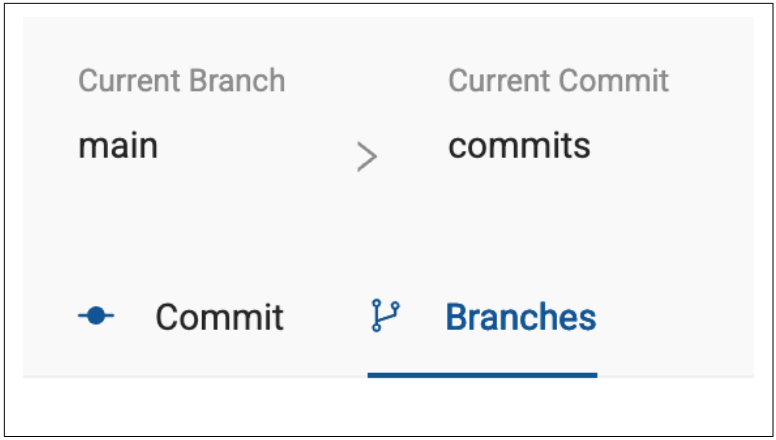


Figure 4-18. The Branches tab, allowing you to work with any branch in your Git Repository in Coalesce.

To change branches, open the Selected Branch dropdown and choose any branch associated with the Git repository connected to your project. This allows you to pull in branches from other workspaces within the same project.

To check out a branch, select it from the dropdown, as shown in Figure 4-19, and click Check Out Latest. This checks out the branch, provided there are no unsaved commits in your current branch. If you need to check out a branch without committing or saving your current work, select the new branch and click Force Checkout. Coalesce will prompt you to confirm before proceeding.

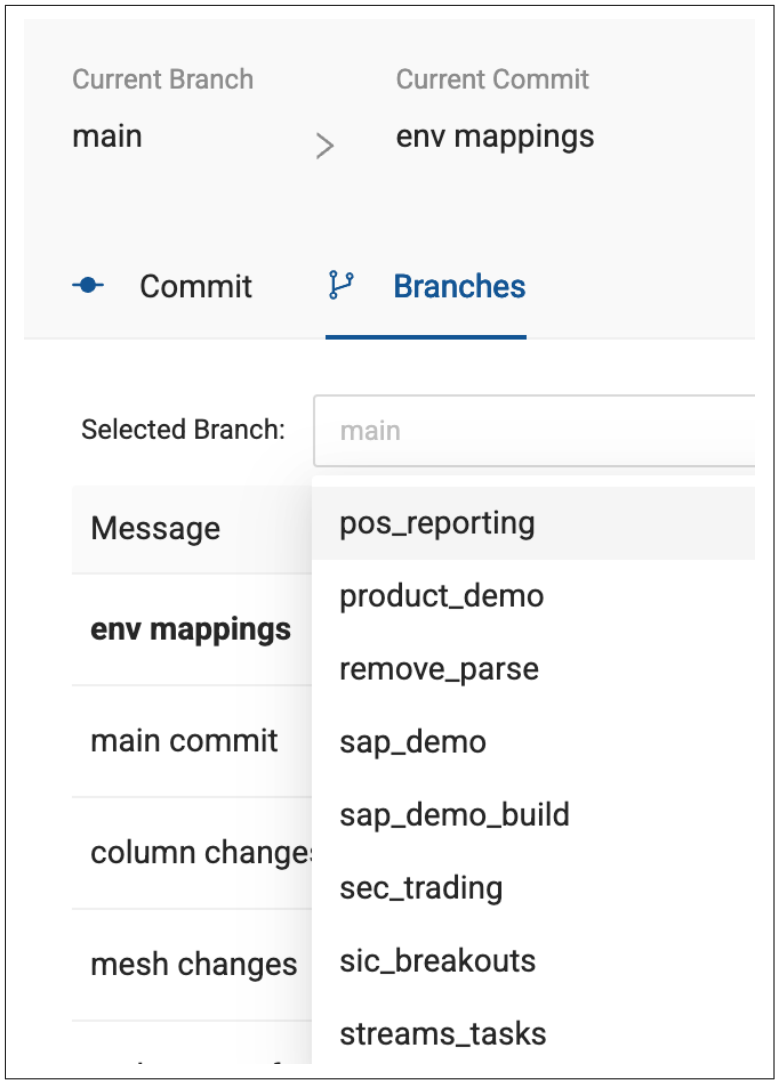


Figure 4-19. The branch selector within Coalesce.

You can merge commits from one branch into another directly within Coalesce. To do this, open the Git modal, select Branches, and choose the branch you want to merge your work into. Coalesce will display all commits in the selected branch. Click Merge, next to the commit where you want to merge your work. Coalesce will preview the merge before applying it. If everything looks correct,

confirm by selecting Merge, and Coalesce will handle the process automatically.

Alternatively, you can use the Merge Latest button to merge only the most recent commit from your current branch.

You can also create a new branch from any commit in any branch within the Git modal. To do this, click New Branch next to the desired commit, as shown in Figure 4-20.

Figure 4-20. You can create a new branch from any commit by selecting the New Branch button.

All of the changes within the git modal in Coalesce will be reflected in your version control system that you have integrated. This gives you complete control over the entire development and versioning of all of your work in the same system.

Macros and Parameters

This section could have been included in the previous chapter, but it fits better as a management topic since it streamlines data development and provides greater control over transformations. Macros and parameters are related but serve different purposes, so we will cover them separately. Let's start with Macros.

Macros

Macros are functions that take arguments and return a string. You can use these strings in node templates, transformations, and joins to ensure consistency across your project. The returned strings ultimately represent SQL commands, allowing your team to define reusable code in a single location and apply it throughout the project.

For example, to determine if a numerical column contains an odd or even value, you could write a macro as follows:

```
{%- macro even_odd(column) -%}
    CASE WHEN MOD({{ column }}, 2) = 0 THEN 'EVEN' ELSE 'ODD' END
{%- endmacro %}
```

In this macro, the function receives a column as an argument and returns either 'EVEN' or 'ODD'. To use this macro in a transformation, call it using the syntax `{{ function('"schema"."table"') }}`. In this case, the function is named even_odd, so calling the macro would look like:

```
{{ even_odd('"CUSTOMER"."C_NATIONKEY"') }}
```

or

```
{{ even_odd('{{SRC}}') }}
```

Macros support everything from simple SQL statements to complex transformations. You can apply them directly to your workspace or include them in any package installed from the Coalesce Marketplace.

To add a macro, navigate to Build Settings and select Macros. You will see sections for Workspace Macros and, if installed, Package Macros, as shown in Figure 4-21.

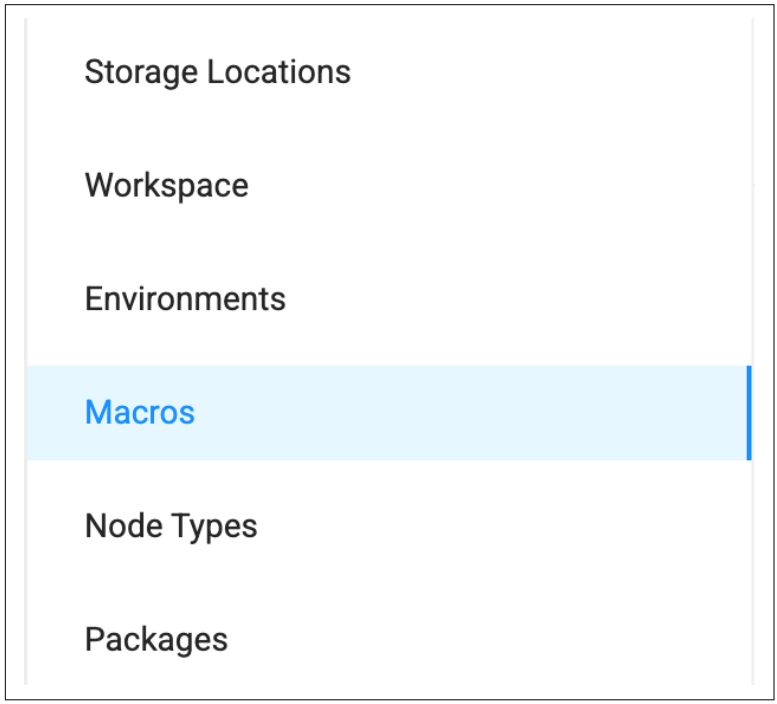


Figure 4-21. Macros area of the Build Settings.

Sometimes when working with Macros, they need to receive input from a parameter, so let's talk about this next!

Parameters

Parameters act as read-only environment variables. Parameters are defined as a JSON blob set by the user that can be accessed in the metadata during template rendering. These values can be set for a Workspace/Environment at various points to provide flexibility of template behavior during Deploying or Refreshing.

There are several reasons to use parameters as outlined below:

Environment Configuration
: Parameters allow you to configure your data pipeline for different environments (for example, development, staging, production) without modifying the underlying code. For example, you can use parameters to specify different database credentials, file paths, or other environment-specific settings.

Dynamic Data Filtering
: You can use parameters to dynamically filter or partition data based on certain criteria. For example, you can use a date parameter to only process data for a specific date range or a customer ID parameter to process data for a particular customer.

Reusability and Modularity
: By changing certain values or logic into parameters, you can make your data pipeline more modular and reusable. This promotes code reuse and maintainability, as you can easily swap out parameter values without modifying the core transformation logic.

Testing and Debugging
: Parameters can be helpful for testing and debugging purposes. You can use different parameter values to simulate various scenarios or edge cases, making it easier to identify and fix issues in your data pipeline.

Scheduling and Automation
: When scheduling or automating data pipeline runs, you can use parameters to pass dynamic values or configurations. This allows you to adapt the pipeline behavior without changing the underlying code, enabling greater flexibility and control.

Separation of Concerns
: By externalizing certain values or configurations as parameters, you can separate the concerns of data transformation logic from the specific values or configurations used in different contexts. This promotes better code organization and maintainability.

Compliance and Auditing
: In regulated industries or environments with strict data governance requirements, you can use parameters to enforce data access controls, data masking, or other compliance-related configurations.

You can easily add parameters into your workspace or environment. To add a parameter to your workspace, navigate to the workspace settings and select the parameter item as shown in Figure 4-22. You will learn about Environments and where to add parameters to them at the end of this chapter.

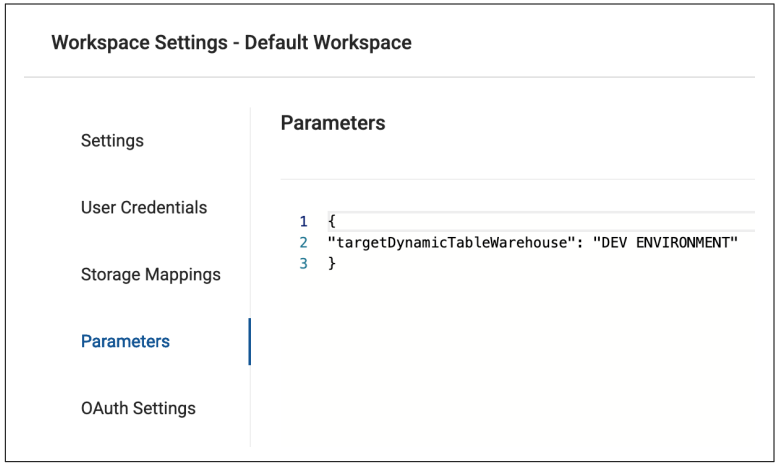


Figure 4-22. The Parameters item within the Workspace settings.

Macros and parameters provide a flexible and modular way to organize and manage data and operations throughout your pipelines. Now that you understand how to manage and work with pipelines, the next step is testing and running them as jobs in an environment. The rest of this chapter will focus on these final steps.

Testing

Coalesce provides built-in testing capabilities to assess data quality. Data quality tests help identify issues such as missing values, incorrect data types, outliers, and violations of business rules. Testing within Coalesce ensures that transformed data meets required standards before it is loaded into the target node. Integrating data testing into the pipeline workflow enables early issue detection and ensures consistency across data sources and transformations.

Coalesce handles testing through built-in Node Testing capabilities, which support tests at both the Column level and the Node level, as shown in Figure 4-23.

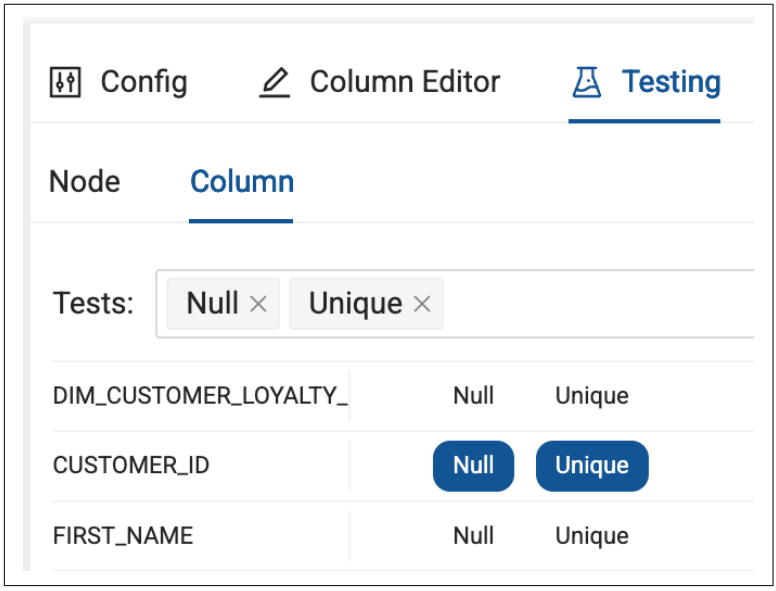


Figure 4-23. Column level tests available out of the box.

You can use these to test for:

Column-Level Tests
: Test individual columns for uniqueness and null values.

Node-Level Tests
: Validate data quality across an entire node.

To apply tests, select a node and click the Testing icon. Choose the tests to run and specify which columns to include, as shown above in Figure 4-23.

You can also write custom node-level tests using SQL queries. If the query returns a result, the test fails. Additionally, you can configure options to determine whether a test failure should halt pipeline execution and whether tests should run before or after the node refresh.

For more comprehensive and extensible testing, you can use the Test Utility Package from the Coalesce Marketplace. This package provides granular control over every component of each node and column within your data pipeline.

Jobs

Now that you understand how to manage pipelines, use version control, and test data, the next step is organizing pipelines into Jobs. In Coalesce, you'll use Jobs to refresh data pipelines. You can execute these through the CLI or the native Coalesce Job scheduler.

Each job is identified by a Job ID, which includes all nodes associated with that job. You can use a selector query to assign specific nodes from your pipeline to a job. To manage jobs, navigate to the Jobs menu in the left-side navigation bar. Here, you can view, modify, and add new jobs. To create a new job, click the + button and select Create Job. You will then define the job's associated nodes using a selector query, as shown in Figure 4-24.

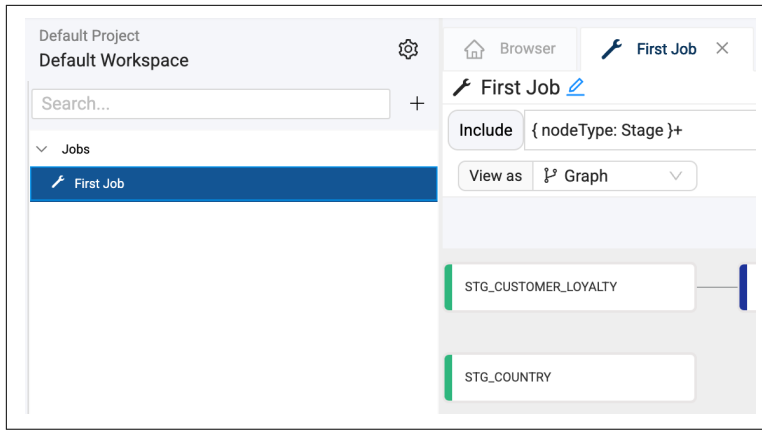


Figure 4-24. Selector syntax for defining a job.

Each Job has a unique Job ID associated with the job. This allows you to reference the Job ID from the CLI to refresh all of the nodes that are associated with the job. Alternatively, if you use the Coalesce Job scheduler, you can reference the job name and have the job refresh on a schedule. You'll learn more about the Coalesce Job scheduler in the next section on environments.

Environments

So far, this chapter has covered how to navigate, version, and manage data pipelines. The final and most critical component of pipeline management is validating pipelines and making them available for consumption. This is where Environments come into play.

Environments handle deploying data pipelines to your data platform. They enable efficient deployment strategies, allowing you to develop in a Workspace and then deploy to any Environment, such as production or QA. Each environment can have different Storage Locations, Storage Mappings, Parameters, and other configuration settings.

An Environment is configured similarly to a workspace, with its own Storage Locations, Storage Mappings, and a connection to your data platform. You can view existing environments from the Deploy Interface in Coalesce. If no environments exist, you can create one from the Build Settings of your workspace.

To create a new environment, open Build Settings in any workspace and navigate to Environments. View the list of available environments for management, then click Create Environment to open a configuration modal. Provide the connection details for your data platform and define Storage Mappings based on the Storage Locations available in your workspace.

By configuring environments this way, data and objects persist within specific locations in your data platform, allowing your team to explore and validate pipelines.

For example, let's assume you have completed development on a new pipeline and need to perform QA testing before deploying to a production Environment. You can create a QA Environment in Coalesce that deploys pipeline objects to a dedicated database and schema in your data platform for testing.

This approach ensures that your data platform is segmented for different stages of pipeline development, from initial testing to final production release, allowing business users to access validated data with confidence.

Deploying

Deployment is the process of pushing your work to your data platform at the specified location. Once you have created your environments, you can deploy your work at any time to the environment of your choice, as long as the required data sources are included to match those used in the workspace.

To deploy a pipeline to an environment, navigate to the Deploy Interface and select the blue Deploy button, as shown in Figure 4-25. This opens the deployment modal, where you'll choose a branch from the project you are working in and select a commit to deploy. For example, if you have just added new nodes to your pipeline and committed the work to your branch, you would select that commit for deployment.

Figure 4-25. The blue deploy button to deploy your commits to an environment.

Once you select a commit, the next step is to provide any parameters, just as you would when setting parameters in a workspace. Finally, Coalesce will generate a deployment plan, displaying all changes and updates. If there are no errors and the deployment looks correct, confirm the deployment, and Coalesce will push the changes to your data platform.

Refreshing Your Pipeline

Once you have deployed your pipeline, you need a way to refresh the data in those objects. Coalesce allows you to refresh pipelines using either the Coalesce CLI or the native Coalesce scheduler. This chapter focuses on the Coalesce scheduler, and we'll cover the Coalesce CLI in a later chapter.

Earlier in this chapter, you learned about Jobs. Once you've created a job, you can use the Coalesce Scheduler to refresh it on a regular cadence, ensuring your pipelines always contain fresh data.

To create a job schedule, navigate to the Deploy Interface and select the ellipsis next to the blue Deploy button, as shown in Figure 4-26. From here, choose View Scheduled Jobs to see all available jobs. You can also create a new job schedule by selecting Create Job Schedule in the upper-right corner.

NOTE

You must be an Environment Admin to perform this action. We'll cover Role-Based Access Control (RBAC) later in this guide.

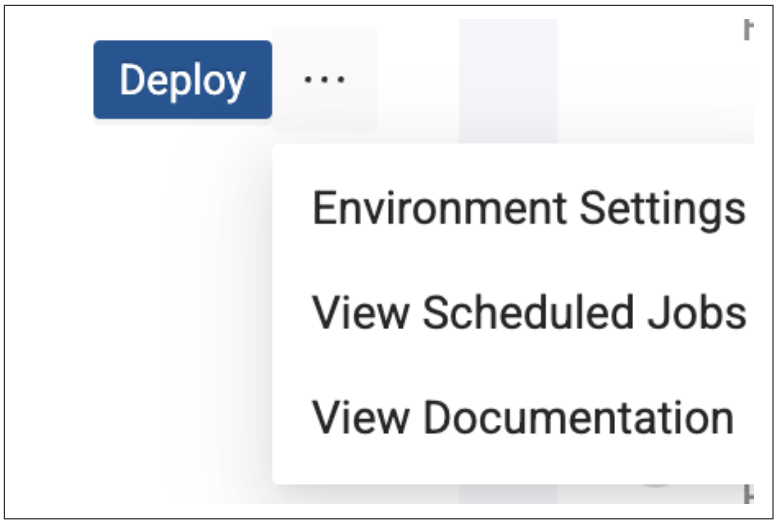


Figure 4-26. The ellipses next to the deploy button to schedule a job.

To complete the job schedule setup, fill in the required parameters, including Project, Environment, and the Job to run. Finally, specify the schedule for the job execution. Once configured, the Job Schedule will automatically refresh all nodes associated with the job at the specified intervals.

The Makings Of A Pro

In the past two chapters, you have learned how to build, manage, and refresh pipelines in Coalesce. You are now well on your way to becoming a Coalesce pro and fully autonomous in end-to-end pipeline development.

Specifically, this chapter covered how to navigate data pipelines, create subgraphs, use version control, leverage column-level lineage, and configure environments. With these skills, you are fully equipped to build and manage any pipeline in Coalesce.

In the next chapter, you will explore security and governance in Coalesce. This will give you the ability to not only build and manage pipelines but also oversee and secure your entire Coalesce instance as an administrator. See you there!

CHAPTER 5

Coalesce Security and Data Governance

A Note for Early Release Readers

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 5th chapter of the final book.

If you'd like to be actively involved in reviewing and commenting on this draft, please reach out to the editor at *mpotter@oreilly.com*.

Throughout this guide, you've laid the foundation for your data projects. You've used Coalesce's building blocks to construct your house, and in the last chapter, you organized everything inside it. Now it's time to protect what you've built. In this chapter, you'll focus on security and data governance in Coalesce—locking the doors, securing the windows, and keeping everything safe.

By the end of this chapter, you'll know how Coalesce supports Multi-Factor Authentication (MFA) and Single Sign-On (SSO). You'll learn how to configure OAuth for secure authentication. You'll also understand how Role-Based Access Control (RBAC) works and how to set it up for your organization. Finally, you'll

learn how to get a clear view of everything you've built and see how it connects across your entire data stack. Let's dig in!

Account Access

Let's begin the conversation about security by talking about accessing your Coalesce instance. Coalesce provides multiple different methods to authenticate, or login, to your Coalesce account. As of the time of this writing those methods are username and password and a variety of supported single sign-on providers, such as Okta. Let's go over a bit more about single sign-on.

Single Sign-On

SSO allows users to authenticate into Coalesce using credentials managed by an external identity provider, streamlining access and improving security. Coalesce supports multi-organization environments, so SSO users can log into multiple organizations with the same credentials. SSO users are provisioned "just-in-time," meaning their accounts are created automatically when they first log in through your identity provider and no manual setup is required beforehand.

To configure SSO in Coalesce, go to the User Menu in the upper-right corner and select Organization Settings—just like you did in Chapter 1. From there, choose Single Sign-On, then select your SSO Authority, as shown in Figure 5-1. Follow the instructions in the Coalesce documentation specific to your SSO provider to complete the setup.

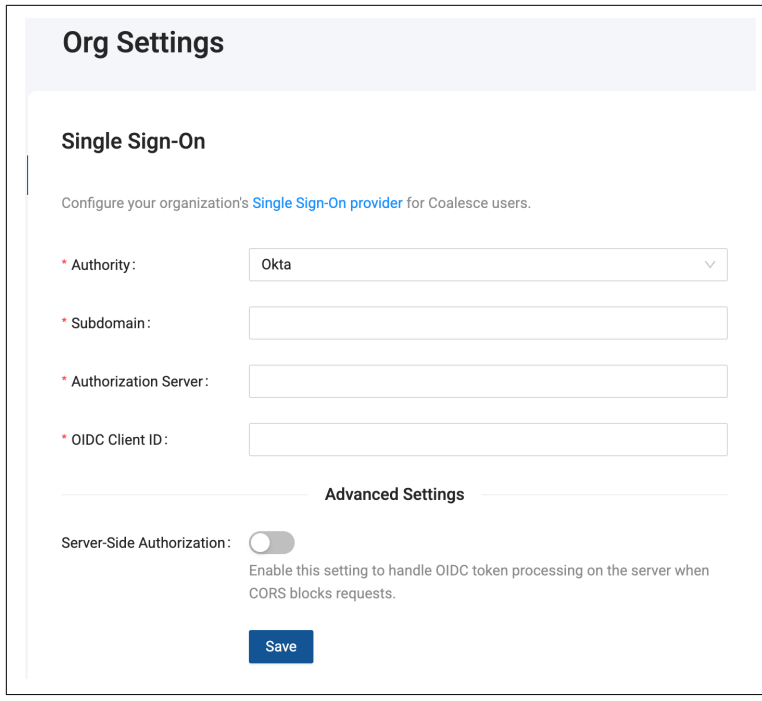


Figure 5-1. Single Sign-On settings within Coalesce.

While SSO is a convenient way to manage access, it's not the only option. If you prefer to use a username and password to log into Coalesce, you can add another layer of security by enabling Multi-Factor Authentication.

Multi-Factor Authentication

MFA adds an extra layer of security to your Coalesce account by requiring multiple factors: something you know (your password) and something you have (a time-based code from an authenticator app). Even if your password is compromised, MFA helps prevent unauthorized access by ensuring only someone with access to your second factor can log in.

To enable MFA in Coalesce, first navigate to User Settings by clicking the User Menu in the upper right corner. Then select Account and Security. Before turning on MFA, make sure your email address is verified. If it's not, resend the verification email and follow the prompts. Coalesce suggests using an authenticator app as your MFA

method. Once selected, you can scan the QR code from your mobile device and you will have a six-digit code generated by your Coalesce MFA now in your authenticator app.

Keep in mind, once MFA is enabled, it can't be turned off by the user. Only an administrator can disable it. MFA is managed on a per-user basis and can't be enforced or disabled across the entire Coalesce organization. Each user must enable MFA individually.

While the focus of this guide is centered on pipeline development in the Coalesce user interface, Coalesce also provides access to a command line interface (CLI) as well as exposing application programming interfaces (APIs). These extensions allow users to access and execute commands in Coalesce without logging into the Coalesce platform. This is done through the use of access tokens.

An access token allows you to make authenticated API requests, ideal for deploying and refreshing jobs using third-party tools instead of the Deploy Interface and job scheduler in Coalesce. To generate an access token, go to the Deploy Interface. At the top of the environments list, you'll find the Access Token Generator as shown in Figure 5-2. Copy the generated key, and you're ready to make API calls.

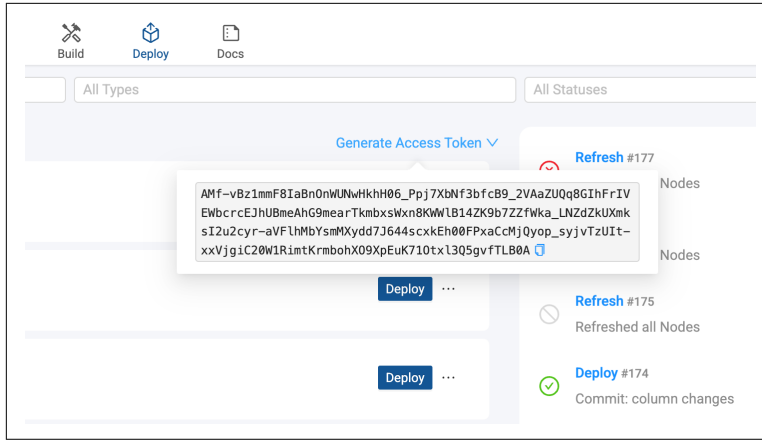


Figure 5-2. The Access Token generator within Coalesce.

So far, we've covered how users authenticate and access Coalesce. But how do you control *what* they can access once they're inside? That's where Role-Based Access Control comes in and that's what we'll cover next.

Role Based Access Control

Role Based Access Control (RBAC) in Coalesce gives organizations fine-grained control over who can access data assets and what actions they can perform within the platform. RBAC works by assigning users to roles. Each role carries a defined set of permissions that determine access to environments, objects, and specific capabilities, such as deploying jobs or editing nodes. Instead of assigning permissions to individual users, RBAC simplifies user management by allowing administrators to create and manage roles, making it easier to scale access control as teams grow.

At its core, RBAC helps organizations enforce data governance and security policies. It ensures that users only have access to the areas of Coalesce they need to perform their tasks. For example, a data engineer working in development might need full access to create and edit nodes, while an analyst working in production may only need read access to published tables. RBAC helps you enforce these boundaries in a clear and auditable way.

Coalesce RBAC is applied at the organization level and then again at the Project and Environment levels. This allows for granular control over exactly who should be accessing what projects and data. This can be seen in Figure 5-3.

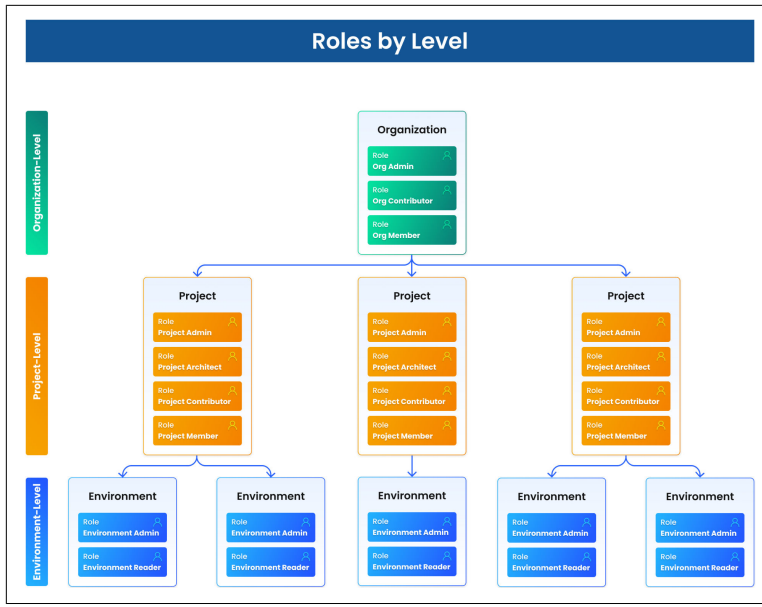


Figure 5-3. The hierarchy of role levels within Coalesce.

At the organization level, assign users an org level role: Admin, Contributor, or Member. After that, you can assign them to specific projects and environments with roles like Admin, Contributor, Architect, or Member. Finally, you have the option to add users to environments as either Environment Admin or Reader. You can see a detailed breakdown of each of these roles in tables 5-1 through 5-3.

Table 5-1. Organization Level Roles.

Role	Permissions Summary	Recommended For
Organization Administrator	The creator of the Coalesce App is automatically assigned as organization administrator. Only organization administrators can add other users, including other organization administrators. They have full access to all functionality in Coalesce.	Full administrative control
Organization Contributor	They can't add new users to the organization. They have access to read documentation, create API tokens, user settings, and Git account information. They will be able to set up a project, configure Git, add members to projects,and oversee work. Only have access to the projects created by them. If there are multiple organization contributors, they will need to share access with the organization contributor.	Managers who decide how each person will contribute to a project.

Role	Permissions Summary	Recommended For
Organization Member	This is the default role. They can edit Git account information, create API tokens, and read documentation.	Default Role

Table 5-2. Project Level Roles.

Role	Permissions Summary	Recommended For
Project Administrator	This role can manage projects, but not create them. An organization administrator or contributor can create projects. The role has access to projects, deployments, and environments.	Team manager or senior team member to manage projects.
Project Architect	This role can manage certain project information, build nodes, and generate API tokens. Assign this role to a data architect so they can build the needed node types, set storage locations, and create macros.	Senior data architects.
Project Contributor	A project contributor can't edit or create custom nodes or macros. They have read-only access to certain project settings. They have read access to projects, deployments, and environments.	Team members who need access to project information without making changes.
Project Member	Assign this role if you want to add them to the environment.	This role could be either a data engineer or a data platform engineer. The project member would not be actively involved in creating or maintaining data pipelines, but would need access to the environment level.

Table 5-3. Environment Level Roles.

Role	Permissions Summary	Recommended For
Environment Admin	This role manages environment settings, reads project documentation, and deploys either through the API, CLI, or Coalesce App.	Data platform engineer or operations who would approve deployments and schedule jobs.
Environment Reader	This role only has access to the documentation for the environment they are added to. They have access to certain API functions to get deployment information.	Business analyst or data analyst.

To manage roles, start by opening the User Menu in the upper right corner of the Coalesce interface. From there, select Organization Settings, then click Users–you'll see a list of existing users. You can either create a new user and assign the organization role at the time

of creation, or select the Actions button to edit the user where you can update their organization role as shown in Figure 5-4.

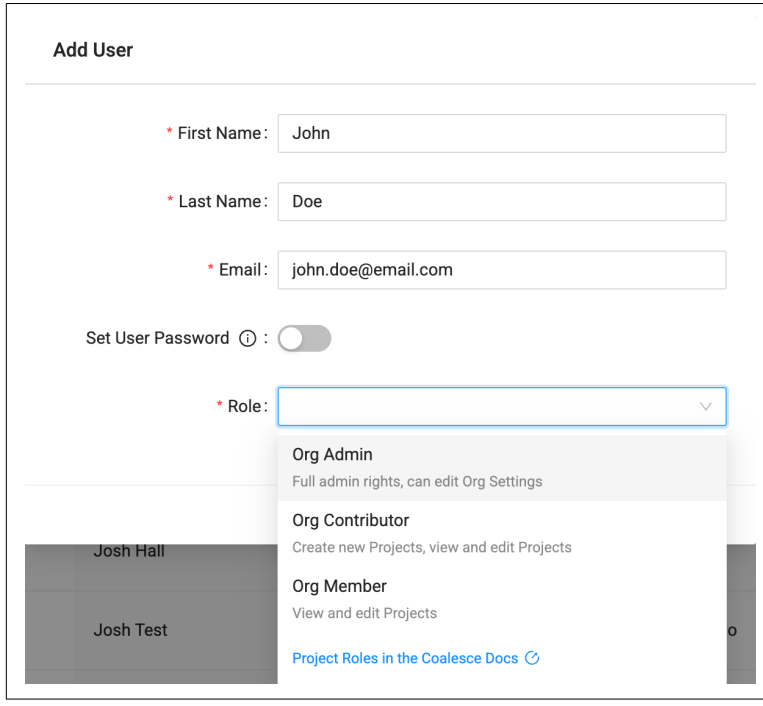


Figure 5-4. Assigning an organization role to a user in Coalesce.

After a user has an organization level role assigned, you can assign users to projects and environments. On the projects page, select any project within your Coalesce organization and select the Projects Settings button. You'll see the Members item within the modal which you can select to view the users in your Coalesce organization who have been added to the project.

To add a user to the project, select Add Member, choose their name, and then provision them with the project level access you want them to have as shown in Figure 5-5. Once you've added a member, you can easily remove them by selecting the Remove button next to their name. Managing access to projects is that easy.

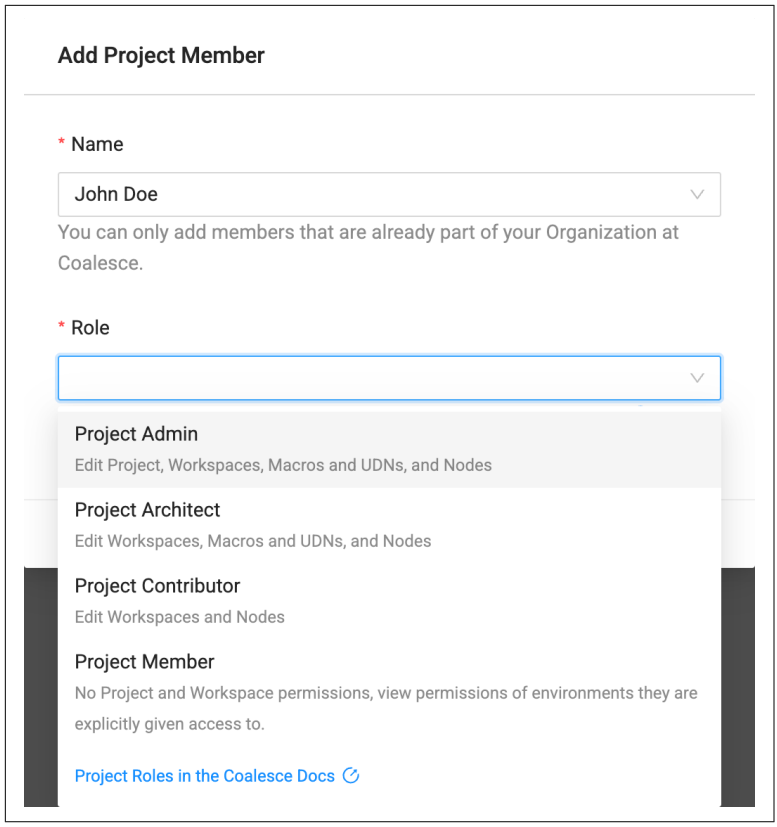


Figure 5-5. Assigning a user a project level role.

You can follow a similar process to provision users to Environments. Select the Deploy Interface and then the ellipsis next to the environment you want to provision access to. Select Environment Settings and similar to Projects, you will see the Members item within the modal. Within Members, select Add Member to add a user who has been provisioned with an organization level role as shown in Figure 5-6.

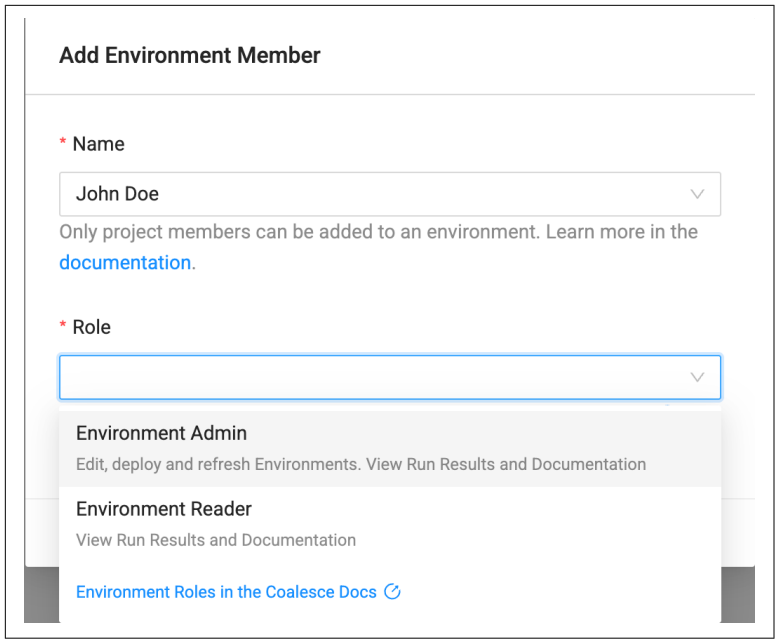


Figure 5-6. Adding a user to an environment in Coalesce.

RBAC is a key part of maintaining a secure and well-governed Coalesce environment. It ensures that users can only interact with the data and environments that are relevant to their role in the data pipeline lifecycle. But in some cases, it's not just your users that you want to provision to ensure your data is secure, it's the technology itself. In the next section, we'll talk about how Coalesce executes your pipelines on your data platform in a secure way.

Coalesce SQL Execution

Ensuring the technologies that you are using within your organization are compliant with the security requirements you demand is critical. As a solution that is transforming data, Coalesce has access to metadata within your data platform. However, Coalesce does not store data from your data platform. Ever. Coalesce is strictly executing the SQL that is generated based on the metadata from your objects and the SQL transformations you have supplied.

In order to support SQL execution from within a browser environment, Coalesce uses a direct proxy that forwards SQL text on behalf of user actions submitted through the Coalesce UI to the customer's

data platform instance. All data is encrypted via HTTPS/TLS and data results from SQL executions are never stored by Coalesce.

When you submit a SQL execution, the browser sends the SQL text to the Coalesce backend. After authenticating the user, Coalesce retrieves credentials from the industry standard credential manager and uses them to connect to your data platform. It then sends the results back to the browser over HTTPS and TLS.

When you request to fetch data, Coalesce follows the same SQL execution path and returns the first 100 rows to the browser for display. Coalesce never stores the data.

As you have seen, Coalesce never stores the data it processes. It only passes encrypted SQL and, when needed, returns a limited result set, 100 rows, to the browser.

With your users now provisioned correctly and your security in place, it is time to focus on sharing the results of your data projects. How do you expose that value to your stakeholders? That is exactly what the rest of this chapter will explore.

Documentation

Documentation is an essential component of building a sustainable data architecture, but is sometimes put aside to focus on development work to meet deadlines. Coalesce automatically produces and updates documentation as developers work.

At the core of Coalesce's documentation, it is capturing everything about each node you have created within any project, workspace, and environment. Information such as the database and schema of the object, each column, data types, lineage, and even the DDL and DML of each node is captured automatically.

However, Coalesce users can elevate their documentation further through the use of column and node level documentation.

Column level documentation lets users add descriptive comments and context for every column within a node. You can enter this information in the Description column within the mapping grid of a node, as shown in Figure 5-7. This field accepts plain text, which you can either enter manually or generate automatically using the Coalesce AI description tool.

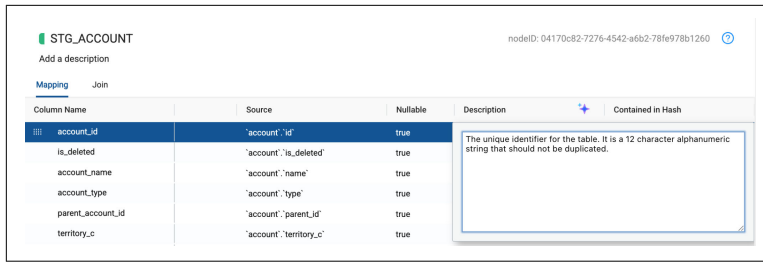


Figure 5-7. Providing a column level description within Coalesce.

The same applies to node level documentation. Business users often find it helpful when objects are clearly described, including what they contain and how they are intended to be used. In Coalesce, users can document a node by selecting the "Add a description" text box located beneath the node name, as shown in Figure 5-8. Just like with column level descriptions, users can enter this information manually or use the Coalesce AI tool to generate a description automatically.

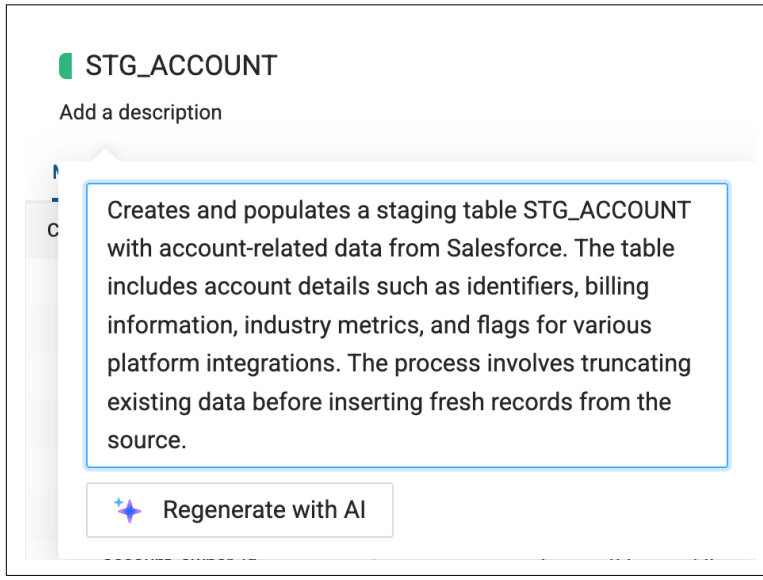


Figure 5-8. Node level description within Coalesce.

Coalesce captures all of this documentation in real time and displays it in the Docs Interface. When you open the Docs Interface, you can view all projects, workspaces, and environments across your Coalesce organization. Selecting any workspace or environment will

show you all of the nodes within that location, as shown in Figure 5-9, along with the metadata associated with each node.

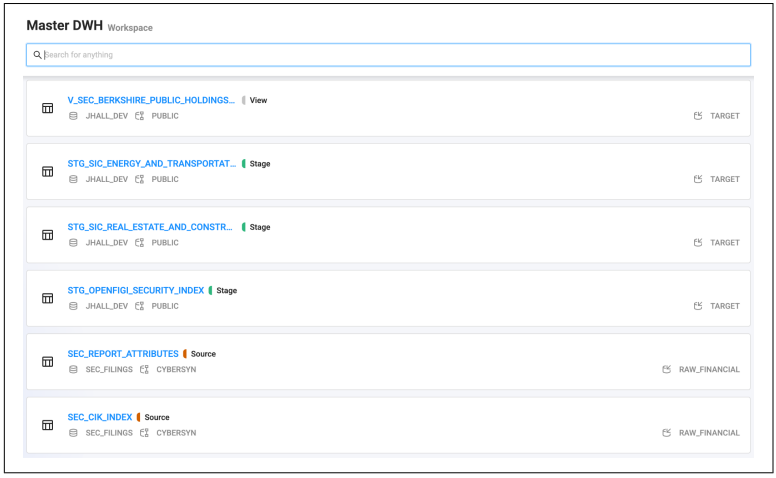


Figure 5-9. Each line item represents a node that has been documented automatically in Coalesce.

By selecting a node, you can drill into a more detailed view of its contents as shown in Figure 5-10. The Docs Interface also includes filtering options that make it easy to find specific nodes or pieces of documentation. This gives users the exact context they need to answer questions, troubleshoot issues, and understand how data flows through the project.

Figure 5-10. Documentation within a node inside of the Docs Interface in Coalesce.

Coalesce lets users manage their organization, control access, and view activity across all projects.

Peace of Mind

Coalesce includes intentional security measures at every step to protect your organization, your data, and your users. In this chapter, you learned the different ways users can authenticate into Coalesce. You saw how to control what users can access once inside, and how Coalesce ensures that your data is never stored, all while making your pipelines easy to understand through clear, structured documentation.

At this point, you have mastered the core components of Coalesce. From setting up your workspace to building pipelines, managing projects, and enforcing security and governance, you now have a complete view of what Coalesce can do. In the next chapter, you will put that knowledge to work as you explore real-world use cases Coalesce helps organizations solve.

CHAPTER 6

Modeling Patterns and Use Cases in Coalesce

A Note for Early Release Readers

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 6th chapter of the final book.

If you'd like to be actively involved in reviewing and commenting on this draft, please reach out to the editor at *mpotter@oreilly.com*.

By now, you have set up your Coalesce account and built scalable, production-ready pipelines. You have handled the core tasks: transforming, versioning, testing, and deploying data transformations. The next question is, what are you building for?

The answer depends on your organization's goals, but most teams follow a few common architectural patterns. There is no universal blueprint, but there are proven approaches that guide how teams model and operate at scale.

Throughout this book, we have used a house-building analogy to ground each phase of development. You laid the foundation, framed the structure, and secured your belongings. Now it is time to decide how to live in the space you created. How should the rooms be

arranged? What temperature should the thermostat be? What rooms should the kids sleep in? All things anyone living in the house has to decide, but may do a little differently than the next. That's exactly what we'll be focusing on in this chapter.

This chapter shifts the focus from how you build to why, and what comes next. It explores common patterns in modern data environments such as migrations, dimensional modeling, and data vault architecture. Each one addresses a different type of problem and requires a specific approach to modeling, testing, and maintaining pipelines over time. By the end of the chapter, you'll have a better understanding of the types of problems you can solve in Coalesce, and how to take advantage of powerful off the shelf functionality to solve your toughest problems.

Migration: From Legacy to Modern

Migrations usually start with a mix of dread and necessity. Moving SQL logic from one platform to another often means rewriting, refactoring, and revalidating everything. It's slow. It's tedious. And it's easy to miss things. But there are ways to break it down. Think of migration like moving to a new house. You have multiple options:

- You pack your own boxes, use your own truck, and move everything yourself.
- You hire movers to pack, drive, and unpack for you.
- You throw out or sell everything and start fresh.

All of these options are valid. Your approach depends on what you're moving, how urgent the move is, and how much help you have. In data terms, that means deciding whether to rebuild, lift-and-shift, or rearchitect.

Building New

There is a unique kind of momentum that comes with building from scratch. No legacy models to untangle. No outdated logic to reverse engineer. Just a clean starting point and the freedom to design something that actually fits your data and your team.

Coalesce makes it easy to start fresh. With nodes, teams can apply consistent modeling patterns across your architecture without

rewriting the same logic over and over. Because nodes are customizable, they reflect your team's standards, not a fixed set of rules. Combined with Coalesce marketplace, it becomes straightforward to define repeatable structures that still allow for edge cases.

Because Coalesce separates node configuration from the SQL logic you write to transform your data, the process of building out data models stays organized. You are not jumping between files or scanning through long SQL scripts to find what changed. Instead, changes are visual, versioned, and easy to understand.

One of the clearest advantages of starting from scratch in Coalesce is that you avoid inheriting tech debt. Every new node benefits from built-in lineage, clear naming conventions, and automatic documentation. The structure you define on day one scales with the project, rather than breaking down as complexity increases.

This is not about moving fast at the expense of long-term stability. Coalesce makes it possible to move quickly because the system is designed to prevent chaos later. When you build new, you are creating pipelines that are easier to maintain, easier to test, and easier for others to understand.

If you have the opportunity to start clean, Coalesce gives you the structure and flexibility to get it right the first time without slowing down.

Migrating As-Is (Lift-and-Shift)

A lift and shift is often the first step in a larger transition. Instead of redesigning everything from the start, you focus on getting your existing pipelines running in the new environment. It is a pragmatic move, especially when time, complexity, or risk makes a full rebuild unrealistic.

In Coalesce, a lift-and-shift means recreating existing logic in a cleaner, more structured way. You bring in the SQL that already works, but you do it with better guardrails. Clearly defined nodes structure your SQL, streamlining business logic. And your pipelines reflect how the data flows, not how it evolved over time.

You are not inventing something new here. You are giving your current data process a stronger foundation. Many teams use this phase to wrap legacy transformations in Coalesce nodes, apply consistent

patterns, and start building toward more modular design, even if the core logic remains the same.

This is also the time to clean up the small things that cause friction. Confusing joins, unexplained filters, cryptic column names, when these show up, Coalesce gives you the structure and visibility to fix them without losing track of the original intent.

Migrating as is does not mean standing still. By moving existing pipelines into Coalesce with clear structure and built-in governance, you create a stable baseline for future changes and avoid carrying over the same problems into the next phase.

Rearchitecting for the Future

Rearchitecting often follows a lift and shift. Once everything is running in the new environment, teams start to notice what no longer works. The logic that made sense five years ago may not hold up today. Business requirements change. Data volumes increase. What used to be manageable becomes brittle, opaque, or too difficult to extend.

In Coalesce, rearchitecting does not have to mean starting over. You can keep what works and replace what doesn't. Most teams begin by creating clearer boundaries, separating raw, staging, and refined layers, for example, or breaking monolithic SQL into smaller, purpose-built nodes. Business logic is fully tested by the team. Naming conventions get a second look. So does the overall shape of the pipeline.

The visual interface in Coalesce makes this easier to manage. You can trace dependencies across your pipeline, spot unnecessary complexity, and understand where changes will have downstream impact. It gives you the context you need to plan a rearchitecture without guesswork.

This kind of work is usually incremental. Few teams rewrite their pipelines all at once. Instead, they modernize one piece at a time, refactoring logic, reorganizing layers, cleaning up joins and filters as they go. Coalesce supports this approach by making changes transparent, standardized, and reversible. You can experiment with confidence, knowing every change is tracked and easy to roll back if needed.

Rearchitecting is about getting to a place where your pipeline is easier to understand, easier to maintain, and better aligned with how your organization actually uses data. Coalesce helps you do that without throwing everything away. As you build in Coalesce, modeling your data for sustainable and maintainable pipelines suddenly becomes important.

Modeling

There is no single way to model data. The right approach depends on your goals, your constraints, and how your team plans to use the data. In the sections that follow, we will walk through several important modeling paradigms: dimensional modeling, data vault, and operationalizing AI and ML pipelines. Each one reflects a different way of thinking about structure, whether you are optimizing for performance, traceability, or predictive insight. Coalesce supports all of these approaches by giving you the tools to apply consistent logic, enforce standards, and adapt your pipeline architecture as your needs evolve.

Dimensional Modeling: The Foundation of Analytics

Dimensional modeling has endured because it provides clarity. It structures data in a way that analysts, engineers, and business users can all understand. At its core, dimensional modeling separates data into two types of tables: facts and dimensions.

Fact tables record measurable events such as orders, page views, revenue, or inventory levels. They are typically long, narrow tables with foreign keys that point to dimension tables. Dimension tables provide context. They describe the who, what, where, and when behind each fact, such as customers, products, locations, dates, and so on.

The result is a model that is predictable and easy to navigate. There are two major variations of dimensional modeling: star and snowflake. A star schema keeps dimensions directly connected to facts. A snowflake schema organizes dimensions further into related lookup tables, often to reflect hierarchies or reduce redundancy. Either way, this approach gives downstream consumers a consistent structure, and it allows business intelligence tools and SQL engines to optimize queries more effectively.

Coalesce supports dimensional modeling not just as a concept, but as a core design pattern. You can explicitly define fact and dimension nodes and apply consistent standards across both. This is not just a label, fact and dimension nodes behave in structured ways. They follow naming conventions, align to logical layers, contain optimized, best practice SQL, and reflect real relationships in your data. The Coalesce interface gives you clear visibility into how these pieces connect.

Surrogate keys, slowly changing dimensions, and timestamped inserts, all of the essential components of dimensional modeling, are straightforward to take advantage of using Coalesce nodes. This becomes even more valuable as your model grows. Dimensional models often start simple but can quickly grow in complexity. A new metric in a fact table may rely on upstream changes across multiple dimensions. An update to customer classification may impact reports across several business units. With Coalesce, you can trace any column from the final report back to the raw source. Lineage is not just a static diagram, it is an interactive map that shows exactly how data is transformed along the way.

Coalesce also helps enforce modeling best practices without relying on tribal knowledge. The templates for fact and dimension nodes reflect your team's standards, whether that means generating business keys, applying audit columns, or implementing slowly changing dimension logic. Once the template is defined, it becomes part of how your team works, not a checklist you have to remember.

As your data model evolves, this structure is what keeps it from becoming fragile. The strength of a dimensional model comes from its consistency. Coalesce gives you the tools to maintain that consistency while still allowing flexibility when needed. You can make changes with confidence, understand the impact of those changes, and keep everything documented automatically.

So while dimensional modeling is a well-established practice, Coalesce brings it into a modern workflow. With repeatable patterns, clear lineage, automated documentation, and strong version control, you can build and maintain dimensional models that scale with your organization and remain easy to work with over time.

Data Vault: Designed for Flexibility and Traceability

Dimensional models work well when the scope is known and the data is clean. But not every environment fits that mold. Sometimes you are working with dozens of source systems, each evolving on its own schedule. Sometimes the requirements include strict auditability, full historical tracking, or regulatory compliance. In those cases, dimensional modeling alone may fall short. This is where the Data Vault paradigm becomes essential.

Data Vault modeling is designed for flexibility, scale, and traceability. It separates structural elements from descriptive ones, making it easier to adapt as systems and business logic evolve. There are three main components of a Data Vault system: Hubs, Links, and Satellites. Hubs store core business entities using unique business keys. Links define the relationships between those entities. Satellites capture descriptive attributes and track historical changes over time.

This separation allows you to ingest and structure raw data in a way that does not require upfront assumptions about how the data will be used. You are not forced to model for analytics on day one. Instead, you build a persistent layer of fact that reflects the raw truth of what happened, when it happened, and how it changed.

In Coalesce, working with the Data Vault pattern feels natural. You can use hubs, links, and satellites as dedicated node types, each with its own set of reusable templates and best practices. Using these nodes, you can apply consistent logic for key generation, audit tracking, and effective dating, without duplicating code or relying on manual processes.

This is where Data Vault shines. It is built for change. When you add a new source system or onboard a new satellite, you do not need to redesign your existing structure. You simply extend the model. The core business keys remain stable. The lineage remains intact. And every change is versioned, timestamped, and traceable.

Data Vault is not optimized for reporting speed out of the box. That is not its goal. Its strength lies in preserving the integrity and history of your data. It creates a single, trusted layer that downstream models, dimensional or otherwise, can pull from with confidence. In regulated industries, or any environment where data lineage and historical accuracy matter, this becomes critical.

Coalesce enhances that traceability by using lineage that is not a static diagram, but contains a fully interactive map of where each field came from and how it was transformed. Combined with automatic documentation, version control, and Git integration, it becomes easy to reconstruct the state of your data model at any point in time.

The modular structure of the vault also makes it easier to manage complexity as your architecture grows. You can incrementally build and refactor hubs, links, and satellites without disrupting downstream models. Coalesce supports this kind of modular thinking, giving you the tools to build a flexible, extensible data backbone without losing control.

So when the priority is not just reporting, but long-term data stewardship, Coalesce and Data Vault are a natural match. They give you the structure, visibility, and discipline to scale confidently, even when everything else is changing, especially in the world of mainstream machine learning and artificial intelligence.

Wrapping Up

Architectural patterns are not about following trends. They are about solving the real problems that arise when data volume grows, systems multiply, and business needs evolve. Whether you are starting fresh, reworking legacy logic, modeling for analytics, preparing AI features, or integrating complex external sources, the patterns in this chapter give you a framework for thinking clearly about structure.

Coalesce is built for exactly this kind of work. It gives you a way to standardize without getting stuck. Templates let you codify best practices without locking you into rigid abstractions. Visual lineage helps you understand what is happening and why. And the ability to wrap even advanced functionality, like AI functionality, into nodes means you are never stitching things together on the side.

The best pipelines are not just technically correct. They are understandable. They are maintainable. And most importantly, they are adaptable. That is what sets long-lived architectures apart, they support change without falling apart.

If you have made it this far, you are likely already solving meaningful problems with data. You learned in this chapter how to think

about migrations within Coalesce. You also learned about some of the common patterns that Coalesce seamlessly supports. These patterns are just one way to think about what comes next. Whether you are scaling up, branching out, or just trying to keep things clean as complexity grows, Coalesce gives you the structure to stay organized and the flexibility to move fast.

You've built your data products with confidence. In the next chapter, you'll see how your team uses them, learn how to govern them more effectively, and use that insight to iterate and improve your pipelines.

CHAPTER 7

Wrapping It All Up with the Coalesce Catalog

A Note for Early Release Readers

With Early Release ebooks, you get books in their earliest form—the author's raw and unedited content as they write—so you can take advantage of these technologies long before the official release of these titles.

This will be the 7th chapter of the final book.

If you'd like to be actively involved in reviewing and commenting on this draft, please reach out to the editor at *mpotter@oreilly.com*.

At this point, you have everything you need to own your data transformations—whether you're a team of one or managing a hundred engineers. In Chapter 5, you locked down governance and security. Now it's time to take the final step: making your data truly discoverable and usable across your business. That's where the Coalesce Catalog comes in.

This chapter is all about helping you take action, not just on the data you've transformed, but the rest of the systems in your stack. The Coalesce Catalog is your window into the entire data ecosystem. Think of it like the index of a book (hey, like this one!). It tells you what's inside, where to find it, and how everything fits together.

Or, to bring back our house analogy, this is where you stop building and start living. The Coalesce Catalog shows you what's in the fridge, where the remote's hiding, and how clean the kids *actually* left their rooms. You can even ask it questions and it won't talk back.

By the end of this chapter, you'll know how to use Coalesce Catalog to unlock the full value of your transformed data, support your stakeholders, and close the loop between creation and consumption. Let's get into it.

Getting Set Up with Coalesce Catalog

Before you can explore metadata, digest documentation, and start understanding lineage, you'll need to connect Coalesce Catalog to your data sources. Setup is fast and straightforward. In this section, we'll walk through how to access your workspace and connect to your data stack to start enriching your catalog from day one.

Once that foundation is in place, you'll be ready to navigate the catalog, support your team, and empower business users to answer their own questions, anytime they need.

Accessing Your Workspace

Coalesce Catalog is a cloud-based application, just like the Coalesce transformation product you've learned throughout this book. You'll either receive an invite link via an email or be prompted to sign in using your work email or Single Sign-On (SSO).

During onboarding, Coalesce will typically help configure key settings for your organization—like enabling approved email domains or setting up SSO. Once that's in place, users can self-serve and create accounts with minimal friction.

User roles are intentionally simple and most users can explore and document your data out of the box. Admins have extra permissions for managing integrations and settings, but there's no complicated access model to learn.

Connecting to Your Data Stack

Once you're in, the next step is connecting your primary data source. Coalesce Catalog supports a wide range of databases and

warehouses, but the setup process is consistent no matter which one you use.

Start by creating a dedicated read-only account in your database. This account only needs access to metadata, like schemas, tables, columns, and query history. It should be restricted from accessing any actual data. It's highly advised to follow along with the onboarding steps of the tooling you are integrating with Coalesce Catalog as each step will have dedicated support for the system you are integrating.

Once the account is ready, open the Integrations section in Coalesce Catalog, as shown in Figure 7-1, choose to connect a database, and enter your connection details. These typically include a host, credentials, and any necessary configuration values. Credentials are encrypted, and only metadata, not data, is ever ingested.

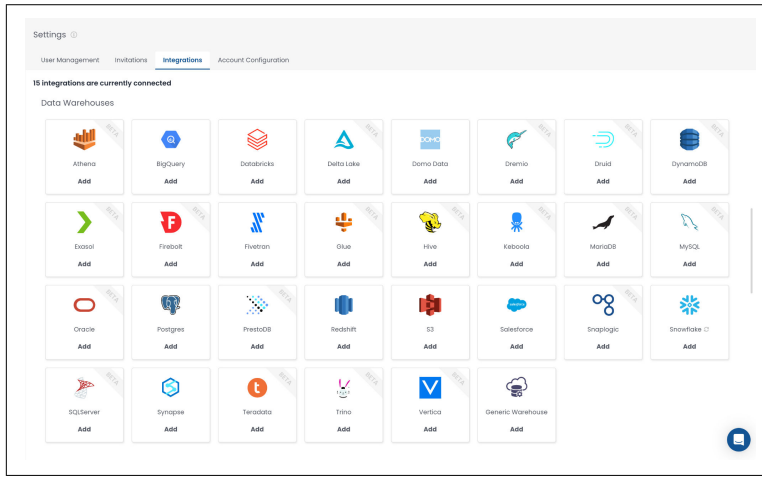


Figure 7-1. The integrations page in Coalesce Catalog. Integrate to over 60+ modern data stack systems.

After saving the connection, Catalog runs an initial sync. It scans your database for structure, databases, schemas, tables, columns, and collects query history to begin building usage insights and lineage. If your environment is large, the first sync may take a bit, but you'll be notified once it's complete.

From then on, the catalog stays up to date automatically. Changes like new tables, renamed columns, or updated usage patterns are picked up on the next scheduled sync. If needed, you can fine-tune

which parts of your environment are scanned by applying filters to include or exclude specific schemas or tables.

You can follow the same process to connect additional tools—whether that's another database, your BI platform, or something else in your data stack. Just choose the integration, provide the credentials, and let Coalesce Catalog handle the rest.

Metadata Management and Data Lineage

Coalesce Catalog is built to simplify metadata management and illuminate how data flows across your environment. In this section, you'll learn how metadata is ingested, how to document and enrich your assets, and how to use the lineage graph to bring clarity and confidence to your data.

Documenting Data Assets

Coalesce Catalog makes it easy to enhance your metadata so it's not just accurate, but genuinely useful. Each table, dashboard, and column can include rich-text documentation as shown in Figure 7-2.

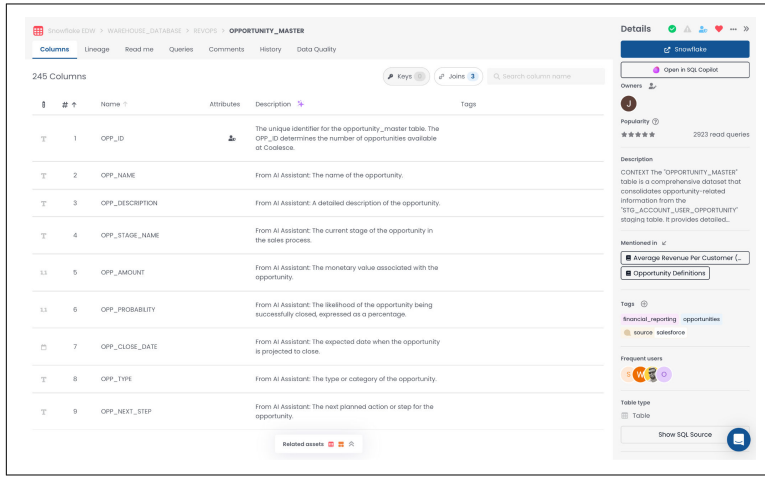


Figure 7-2. Using rich-text to document columns and providing context to the table.

When documenting common fields like "id" or "created_at," Coalesce Catalog suggests existing definitions from similar columns elsewhere in your catalog. You can choose to reuse or propagate

those descriptions to other same-named columns to maintain consistency. If you need to document many fields at once, the metadata editor provides a spreadsheet-style view to streamline updates.

AI support can help speed up your documentation process. The "Describe with AI" feature drafts suggestions based on context and naming conventions. You can review, edit, or regenerate as needed as it's designed to help, not replace, your judgment.

Within each asset, you can use tags to help categorize assets by things like business domain, sensitivity, or quality status. Then, you can assign owners to each asset to ensure accountability, or take advantage of frequent users surfacing automatically so you can identify subject-matter experts even if they aren't officially assigned.

To help teams track progress, Coalesce Catalog also calculates a documentation completeness score. This score is based on criteria like whether an asset has an owner, a table description, and column-level documentation. While all of this documentation is helpful for gaining contact, one of the best ways to understand how data assets came to exist is by understanding the lineage.

Understanding Data Lineage

Lineage is one of the most powerful features in Coalesce Catalog. It shows how data moves from source to transformation to consumption—helping you debug issues, validate assumptions, and avoid downstream breakage.

You can explore lineage in two main ways: the list view and visual graph. The list view gives you a summary of upstream and downstream relationships. The visual graph brings everything to life as shown in Figure 7-3. It starts focused, just the selected asset, and lets you incrementally expand upstream or downstream by clicking the + or - buttons. This makes it easy to trace dependencies or investigate impact without getting overwhelmed. Different asset types are visually distinguished, and interactive elements help you move through the lineage step by step.

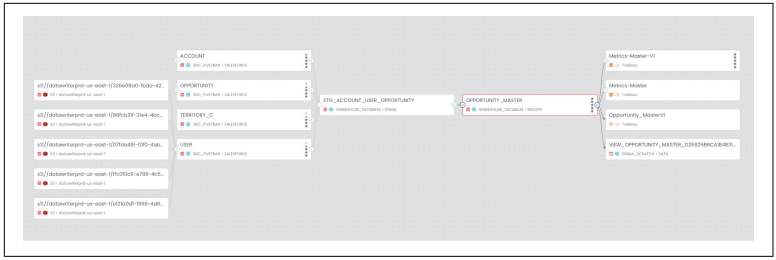


Figure 7-3. Using lineage to understand the flow of your data from source to consumption.

With this lineage, it's easy to perform impact analysis to help you avoid breaking dashboards or reports by showing what depends on a dataset before you change it. You can also perform root cause analysis allowing you to trace confusing or incorrect values back to their source. And for governance, lineage provides visibility into how sensitive data flows through the system, supporting compliance efforts and data quality reviews.

Quality and Governance Metadata

Coalesce Catalog brings in quality and governance signals so you can make informed decisions about what data to trust and how to use it. If your environment includes testing or monitoring tools, Coalesce Catalog can surface quality results directly alongside assets. This might include test outcomes, data freshness indicators, or known issues—giving users immediate confidence (or healthy skepticism) about what they're looking at.

Stewards or admins can certify trusted datasets by selecting the green checkmark in the detail pane of any table or view, which adds a visible badge for users to identify and gain confidence in the data they are using. Conversely, deprecated assets are clearly flagged allowing users to reach out for better guidance or find a certified table. These labels steer teams toward the right sources of truth and can be applied to any asset, as shown in Figure 7-4..

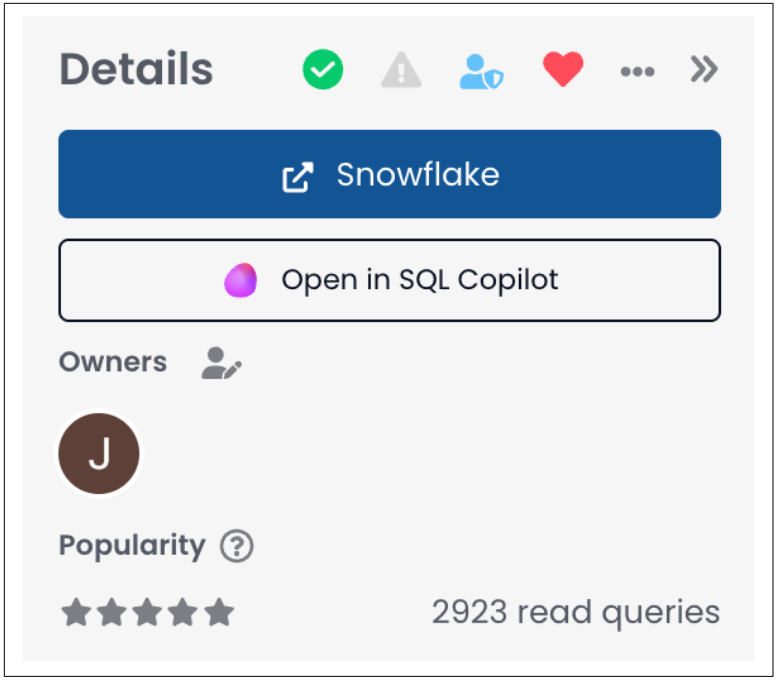


Figure 7-4. Labeling a table as certified (green), containing PII (blue), and a personal favorite (red).

You can flag sensitive data, like personally identifiable information (PII), either automatically or manually. Coalesce Catalog may detect fields like email or ID numbers based on pattern recognition, but confirmation by data stewards ensures accuracy. These PII flags appear in search, filters, and lineage, helping teams protect and handle data responsibly.

This is another case where lineage can support governance workflows. Teams can track where sensitive data originates and where it flows, making it easier to manage policies like data retention or privacy compliance. Seeing this full path of a dataset builds trust and enforces transparency.

By unifying documentation, lineage, quality signals, and governance context, Coalesce Catalog becomes a single source of truth for how your data works—and how it should be used. This in turn promotes self-service discovery and a core driver in collaboration for all teams working with your data.

Collaboration and Data Discovery

Fostering a data-driven culture takes more than cataloging assets, it requires making data easy to find, understand, and trust. Coalesce Catalog is designed to encourage collaboration and simplify discovery so everyone, from engineers to analysts to business users, can participate in growing the organization's knowledge base.

Collaborative Features

Documentation in Coalesce Catalog is treated as a team sport. Anyone can contribute, ask questions, or surface issues, which makes it easier to crowdsource knowledge and keep your documentation current.

Every data asset includes a comment section where users can ask questions or start discussions. These are threaded conversations that stay visible for others, reducing repeated questions and giving future users valuable context. For example, a business user might flag an unexpected number, and an analyst could reply with an explanation or link to related data.

When documentation is missing, users can request it directly from the asset. This sends a notification to the assigned owner or data team, creating a lightweight workflow to close documentation gaps. Once the request is fulfilled, the user is notified, keeping the loop tight and productive.

If there's a problem with a dataset, like unexpected nulls or incorrect values, users can report an issue directly from the catalog. This keeps issues tied to the source instead of scattered in Slack threads or buried in emails. When the issue is resolved by your data or engineering team, it can be closed with context for others to review later.

All of these activities generate notifications. Whether it's a comment, mention, request, or issue, Coalesce Catalog ensures the right users are alerted, either in-app, via email, or through connected tools like Slack. Notifications can also be team-based, so entire groups are notified for certain tags or domains.

The Slack integration plays a major role in bringing Coalesce Catalog into your team's daily workflow. Notifications can be routed to Slack channels, and users can search the catalog directly in Slack via

the native Coalesce Catalog Assistant. For example, asking "do we have any dashboards on regional sales" in a data channel may return a few results with direct links of high relevance. This dramatically reduces the number of one-off data questions.

These features work together to create a shared sense of responsibility and community around data. Producers document and maintain, consumers ask and learn, and the catalog becomes more useful and mature over time. But sometimes users need to dig deeper than just asking a question in Slack. In this case, they can use catalog's rich discovery features to help answer their questions.

Discovery Features

Beyond collaboration, Coalesce Catalog helps users find what they need, even when they don't know exactly what they're looking for. The search experience is tuned to prioritize the most useful results where popular assets, certified datasets, and frequently accessed resources appear higher in results, helping users zero in on trusted sources. If you're new to the catalog and search for "Sales," the most reliable table or dashboard will rise to the top.

On the homepage or in search results, Coalesce Catalog can highlight trending assets, datasets or dashboards with a spike in usage. This visibility helps teams stay aligned during reporting cycles or major projects. For each asset, you can see who uses it most frequently. This makes it easy to find someone to reach out to when you have questions. Whether you're new to the company or just working with a new domain, knowing who uses the data helps you find context fast.

The knowledge section serves as a data wiki for your entire data team. It's the place to define business terms like "active users" or "net revenue," and to link those definitions directly to the datasets or dashboards that implement them. If someone searches a business concept, they might land on a glossary page that explains the metric and links out to relevant data assets. This is incredibly beneficial in providing a singular definition for metrics used throughout the organization.

The Catalog browser extension lets you bring Coalesce Catalog into other tools. While working in a database UI or BI tool, the extension will show you all of the relevant information documented in the

catalog within the page you are viewing as shown in Figure 7-5. This provides in-context help without switching tabs.

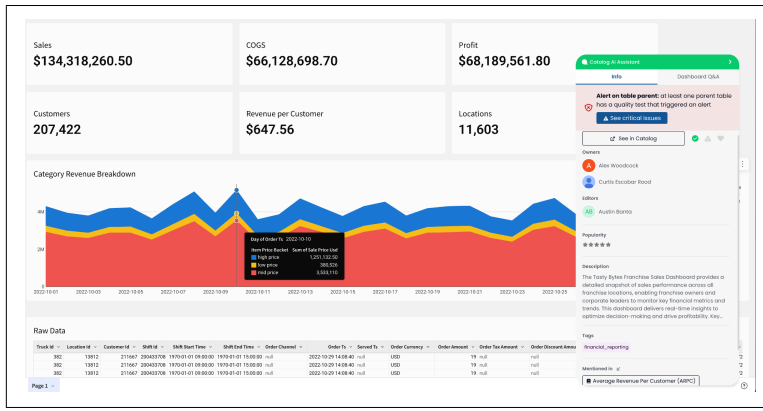


Figure 7-5. The Catalog assistant follows you into the tools you are already working in, providing context even when you're not directly in the catalog.

If enabled, the AI Assistant adds another layer to data discovery. Users can ask natural language questions like "how do we calculate churn rate?" and get context-rich answers drawn from your documentation. This is especially helpful for non-technical users who aren't sure what to search for. Over time, Coalesce Catalog learns from these usage patterns and it can recommend assets, highlight similar datasets, or identify potential overlaps. This means every user's experience is enhanced with the use of AI throughout each touchpoint of the Coalesce Catalog.

Using the AI Assistant for Data Discovery and Exploration

The AI Assistant in Coalesce Catalog acts like a knowledgeable colleague who is available anytime. Using natural language, anyone can explore the catalog, uncover definitions, trace data lineage, or find the right dashboard without knowing table names or writing SQL.

The AI Assistant lets you search the catalog in plain English as shown in Figure 7-6. Whether you're looking for "monthly revenue by region" or "customer churn rate," it interprets your intent and surfaces the most relevant datasets, dashboards, or glossary terms.

It's tuned to your company's language and structure, so it understands business context and not just schema names.

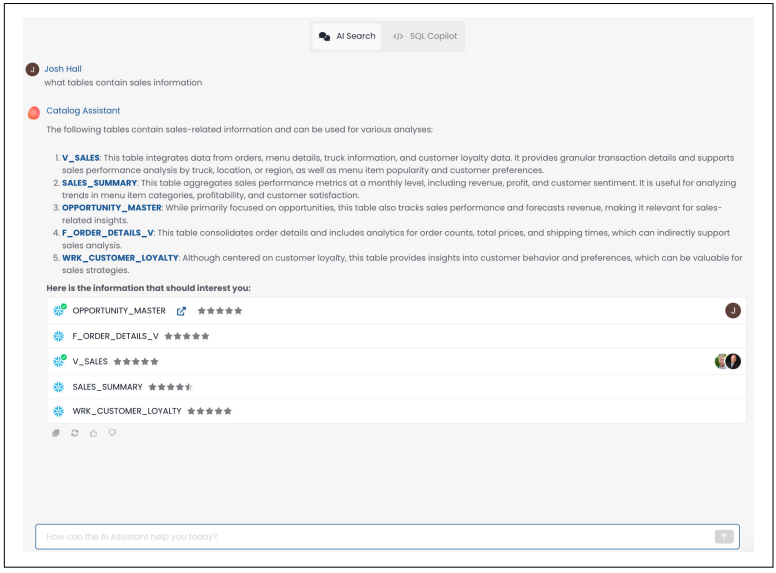


Figure 7-6. Using the AI Assistant to search the catalog in plain English.

It also acts as an instant Q&A tool. You can ask questions such as "How is churn rate calculated?" and it'll respond with the official definition from your knowledge base, along with links to relevant assets. If you're not sure where customer purchase history is stored, you can ask and the assistant will locate the right datasets for you.

Beyond answering questions, the AI Assistant also connects the dots. Ask about a dataset, and it might tell you what dashboards use it, who owns it, or what glossary terms are related. If you're already viewing a table, it adapts to you, letting you ask things like "What upstream source does this come from?" or "Are there issues reported on this asset?"

You can access the assistant from within the catalog interface, whether browsing a table, searching across assets, or chatting via integrated tools like Slack. This allows users to take advantage of all of the rich-context stored in Catalog regardless of the system they are working in, meeting them where they already work. In doing so, the entire data transformation cycle is brought full circle allowing,

users to have clarity into even the most complex data transformations, even if they don't know how to write SQL themselves.

From Chaos to Clarity

You've seen it all at this point. You know what it looks like to build a modern, high-functioning data practice using Coalesce. From hands-on SQL transformations to catalog-wide collaboration, this guide has walked you through the full lifecycle of managing, documenting, and delivering trusted data at scale.

At its core, Coalesce is designed to take the repetitive, brittle, and fragmented parts of data work and turn them into something structured, powerful, reusable, and automated. You started by learning how Coalesce transformation gives engineers a clean, visual interface for building logic while still generating code that's precise and production-ready. With every node you added, every column you documented, and every relationship you configured, you built something bigger than a data pipeline, you built a process others can follow.

Then with Coalesce Catalog, you learned how to capture the entire context around your data: where it comes from, how it's used, who owns it, and what it means. No more scattered documentation. No more Slack messages asking what "ARPU" stands for. No more hoping someone wrote it down in a Google Doc two quarters ago. Catalog ties everything together, your lineage, your logic, your glossary, and puts it at your fingertips. And with the AI Assistant layered on top, even non-technical users can find what they need with a simple question.

Together, these two parts, Transformation and Catalog, create a feedback loop. When your transformations are well-documented, they're easier to understand. When they're easier to understand, they're easier to trust. And when people trust the data, they use it. That's how you shift your organization from pipeline firefighting to true data enablement.

This isn't about moving data from point A to point B. It's about creating systems people can understand, build upon, and rely on. It's about letting engineers scale their output without adding complexity. It's about letting analysts explore without gatekeeping. It's about

empowering business users to make decisions without waiting in line.

As you move forward, here are a few things to keep in mind:

- Build with clarity. Your future teammates will thank you.
- Don't treat documentation as a separate task. When it's part of the workflow, it actually happens.
- Make the catalog the first stop for every data question.
- Use lineage not just to debug, but to build trust.
- Enable business users to answer their own questions with the AI assistant. This gives you more time to work on high-value tasks.

What you build using Coalesce isn't just a set of tables and models, it's a foundation. One that enables faster development, better governance, and a more confident data culture. So whether you're scaling a team of 100 or holding down the fort as a team of one, Coalesce gives you the structure, automation, and transparency to do more, with less friction.

Now it's your turn to carry that momentum forward. Keep iterating. Keep sharing knowledge. Keep raising the bar for what good data work looks like in your organization.

Now go build something remarkable.

About the Author

Josh Hall is a data engineer and product marketer that loves to help others learn, explore, and understand the power and impact of the world of data. He spends much of his time creating video content to help others understand technology, writing blogs to distill technical concepts, and traveling to speak at data conferences and user groups. With over half a decade of data consulting experience under his belt, and several more years with data products, Josh wants to take all of his hard earned knowledge and allow others to expedite and amplify their data journey.