

Chapter 7. Exploring Databricks SQL

Databricks SQL is an essential component within the Databricks ecosystem, simplifying the process of querying, visualizing, and alerting on data. It empowers data analysts and business intelligence professionals to easily uncover insights and extract value from their data. This chapter explores Databricks SQL, covering its various components, such as SQL endpoints, dashboards, and alerts.

What Is Databricks SQL?

Databricks SQL (DBSQL) is a data warehousing solution specifically designed for scalable business intelligence applications. It offers features for executing and managing SQL queries, creating interactive dashboards, and setting up alerts, all while maintaining unified governance. With Databricks SQL, teams can easily analyze and visualize large datasets, share insights, and make data-driven decisions.

In the Databricks workspace, you can find Databricks SQL tools in the left sidebar, under the SQL section, as illustrated in [Figure 7-1](#).

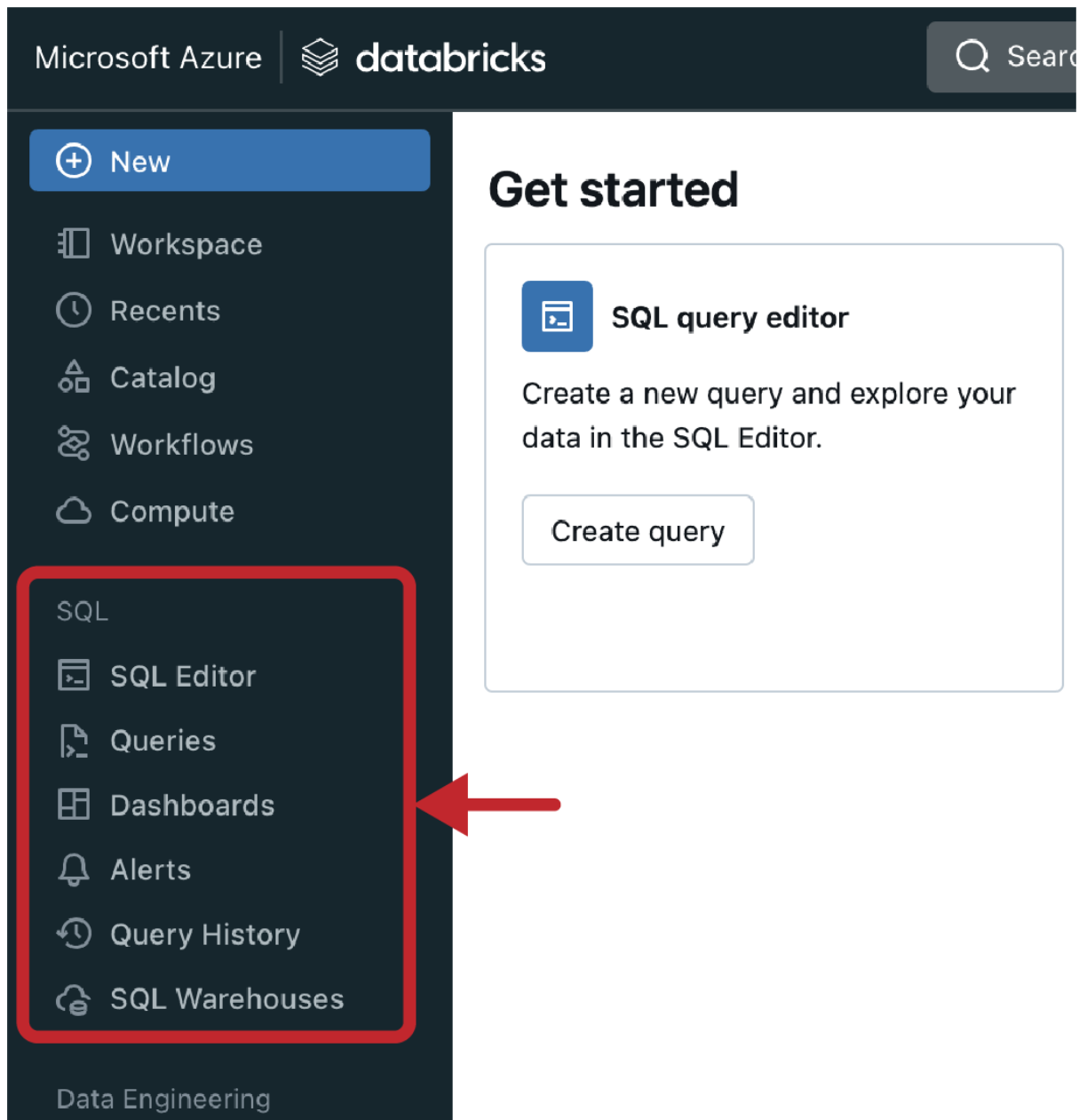


Figure 7-1. Databricks SQL menu

As shown, Databricks SQL offers a range of new options, including the following:

SQL editor

An intuitive interface for writing and executing SQL queries

Queries

A repository for storing and managing frequently used queries

Dashboards

A visualization tool for creating interactive dashboards

Alerts

A feature for setting up custom alerts and notifications based on data changes

Query history

A log of executed queries, facilitating tracking and auditing

SQL warehouses

Scalable compute resources optimized for executing SQL workloads

Each of these components plays a crucial role in leveraging the full potential of Databricks SQL. Let's take a closer look at each of them, starting with SQL warehouses.

Creating SQL Warehouses

SQL warehouses are the backbone of Databricks SQL, providing the computational power necessary to run SQL queries at scale. They are essentially compute clusters based on Apache Spark and the Photon engine. These clusters are highly optimized for SQL workloads and provide some additional benefits that enhance performance and increase concurrency compared to traditional clusters. This enables you to efficiently manage and execute your queries and dashboards.

To get started with SQL warehouses, let's navigate to the SQL Warehouses tab in the left sidebar of the Databricks workspace. This page is the control center for creating and managing your SQL computational resources, as shown in [Figure 7-2](#).

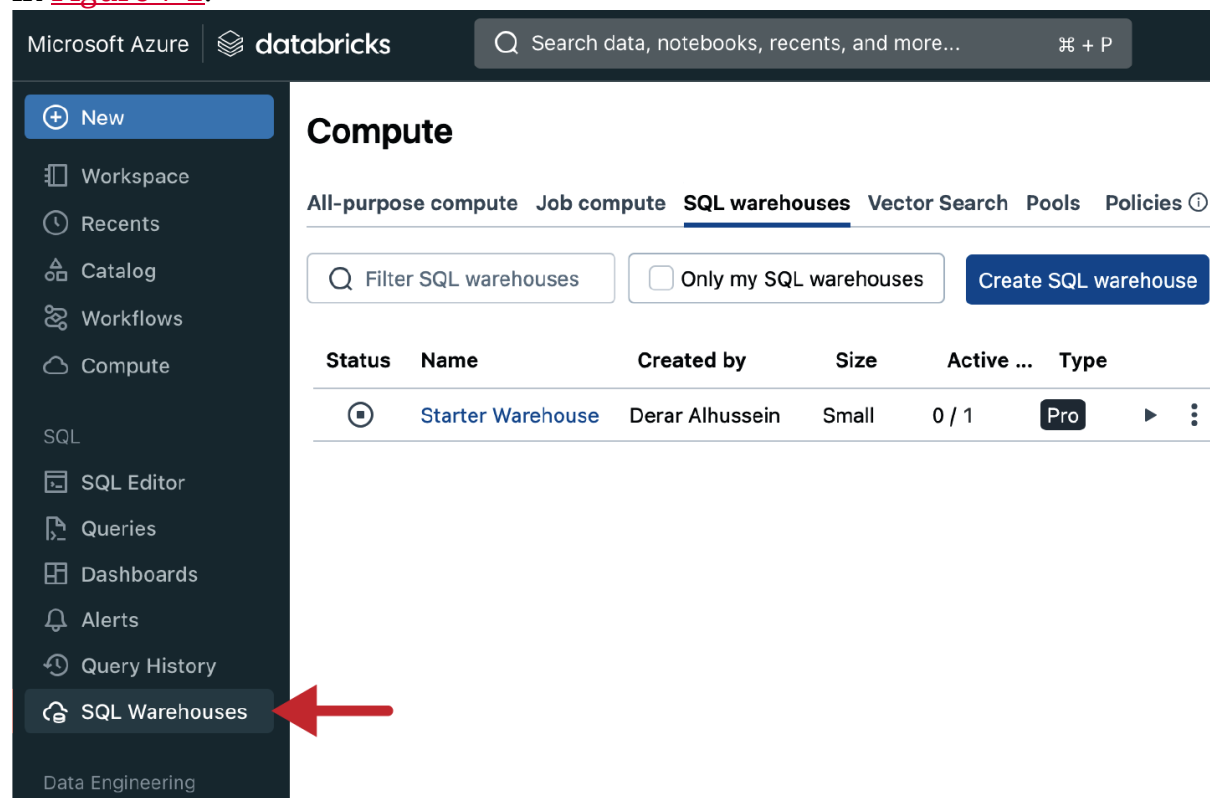


Figure 7-2. SQL warehouses page

Configuring a SQL Warehouse

To set up a new SQL warehouse, click the “Create SQL warehouse” blue button, which will prompt a configuration window, as illustrated in [Figure 7-3](#). Here, we can specify the details of our new warehouse.

New SQL warehouse ✕

Name

SQL warehouse name

Cluster size ⓘ

2X-Small 4 DBU / h ▾

Auto stop

☒ After minutes of inactivity.

Scaling ⓘ

Min. Max. clusters (4 DBU)

Type

☒ **Pro** ⓘ ☐ Classic

Advanced options ▾

Cancel

Create

Figure 7-3. SQL warehouse configuration window

Let’s create a SQL warehouse instance, which we’ll call “Demo Warehouse.” For this demonstration, we’ll set the cluster size to 2X-Small to allocate minimal resources suitable for light workloads. We will leave all other options at their default settings and proceed to click Create.

SQL warehouses typically take a few minutes to start. However, you may also have the option to choose a Serverless compute type, as illustrated in [Figure 7-4](#). This offers a fully managed service by Databricks that does not require managing infrastructure in your cloud account. It starts instantly, without the need to wait for provisioning any resources.

New SQL warehouse

Name

SQL warehouse name

Cluster size ⓘ

2X-Small

4 DBU / h

▼

Auto stop

After

10

minutes of inactivity.

Scaling ⓘ

Min.

1

Max.

1

clusters (4 DBU)

Type

→

Serverless ⓘ

Pro ⓘ

Classic

Advanced options

▼

Cancel

Create

Figure 7-4. SQL warehouse configuration window with Serverless compute type

Once the creation process is complete, our SQL warehouse is running and ready to be used, as shown in [Figure 7-5](#).

[SQL Warehouses](#) >

Demo Warehouse



Permissions



Edit



Stop

Overview

Connection details

Monitoring

Status



Running

Name

Demo Warehouse (ID: a620dd90d7ad18c2)

Type

Serverless

Cluster size

2X-Small

Figure 7-5. Fully operational SQL warehouse

SQL Endpoints

A key feature of SQL Warehouses is the provisioning of SQL endpoints, which allow external business intelligence (BI) or other SQL-based tools to connect and access data in the lakehouse. To connect to your SQL warehouse, you can refer to the connection information provided under the “Connection details” tab, as shown in [Figure 7-6](#).

Demo Warehouse



Permissions



Edit



Stop

Overview

Connection details

Monitoring

Server hostname

adb-1539897213482703.3.azuredatabricks.net



Port

443



Protocol

https



HTTP path

/sql/1.0/warehouses/a620dd90d7ad18c2



Figure 7-6. Connection details of the SQL warehouse

With our SQL warehouse up and running, we can now leverage its capabilities for an array of SQL workloads. In the next sections, we will explore how to use the SQL warehouse to create dashboards, run queries, and more.

Designing Dashboards

Dashboards in Databricks SQL offer a dynamic way to visualize data and insights derived from SQL queries. They enable you to create interactive and shareable visualizations of your data, making it easier to understand and communicate complex information.

To begin working with dashboards, navigate to the Dashboards tab in the left sidebar of your Databricks workspace. Here, you can manage your existing dashboards or create new ones, as shown in [Figure 7-7](#). You can also explore sample dashboards by clicking the “View samples gallery” button.

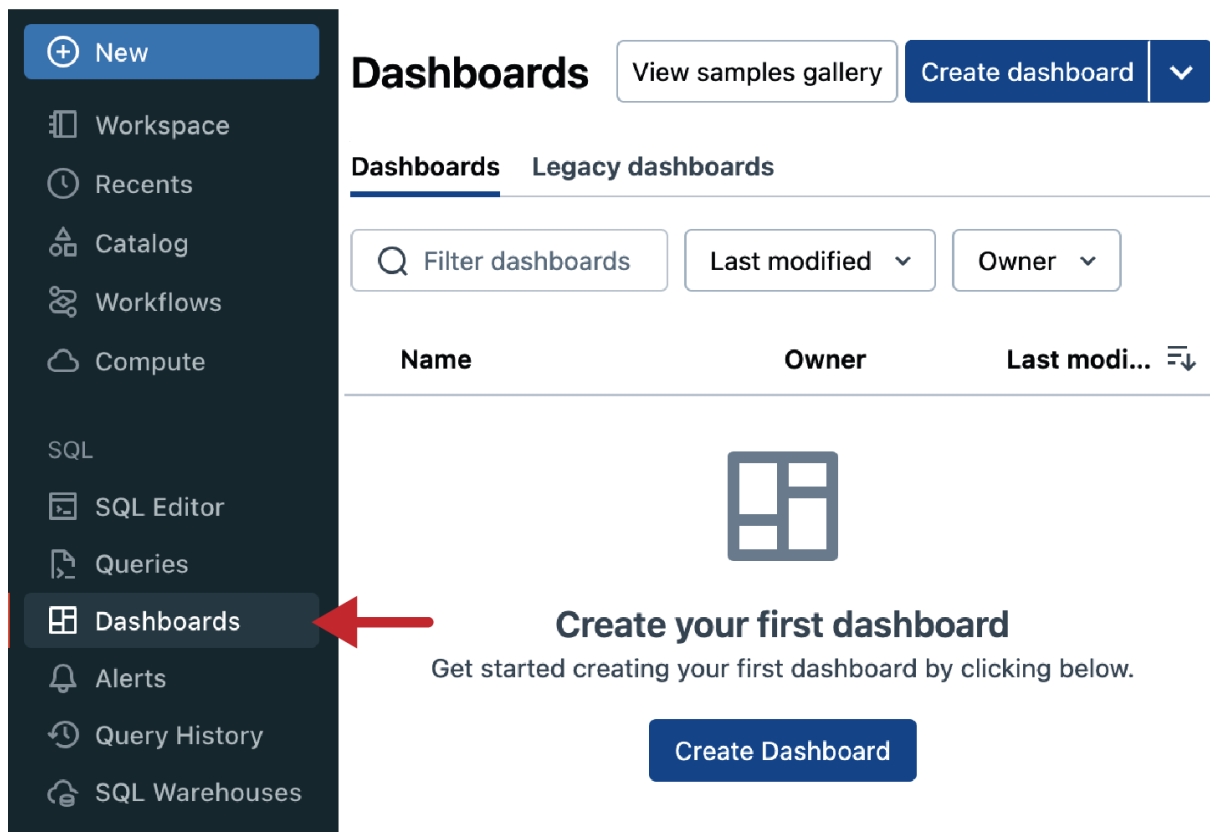


Figure 7-7. Dashboards page in Databricks SQL

Creating a New Dashboard

To create a new dashboard, click the “Create dashboard” button in the upper-right corner of the page. This opens the dashboard editor interface, which consists of two panes: the Canvas pane and the Data pane, as illustrated in [Figure 7-8](#):

Canvas pane

This is where you design your dashboard. You can drag and drop different visual elements, such as charts, graphs, and tables, to create a visually appealing and informative layout.

Data pane

In this pane, you define the source datasets for your dashboard. There, you can specify the data sources that will feed data to your visualizations.

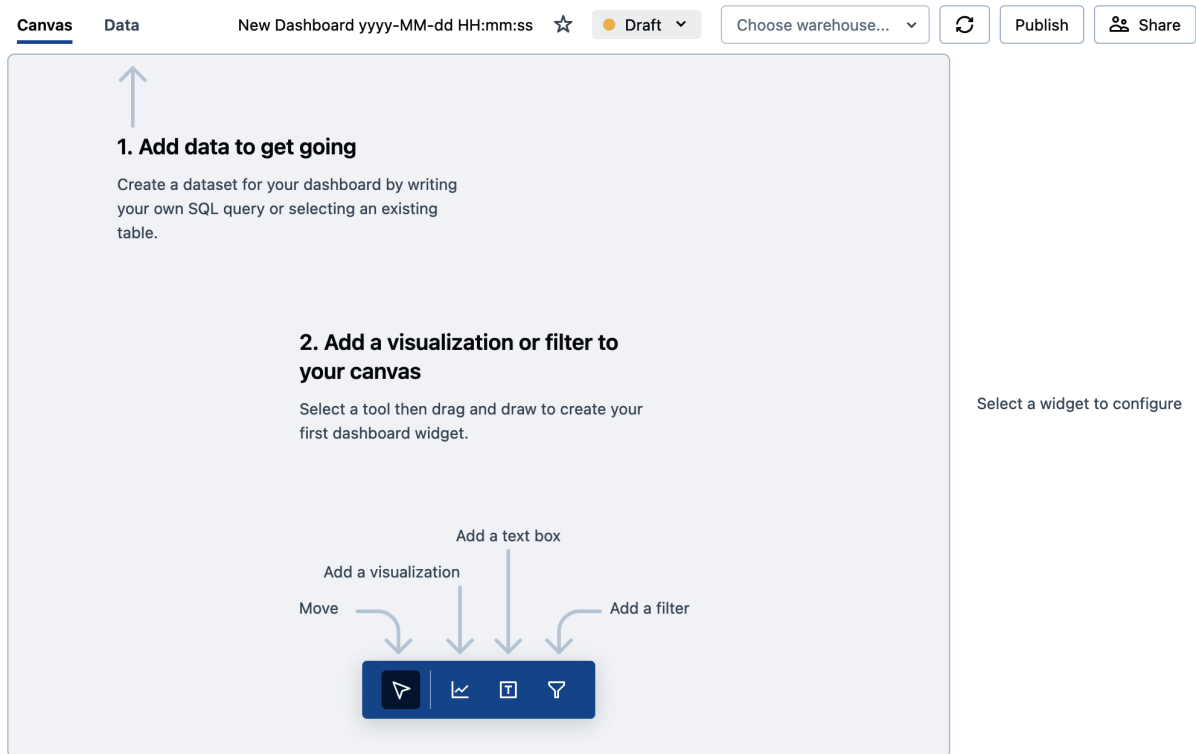


Figure 7-8. Dashboard editor interface

At the top of the dashboard editor interface interface, you will see a default name assigned to the dashboard (*New Dashboard yyyy-MM-dd HH:mm:ss*). Click this name to rename it to something more descriptive, such as “School Demo Dashboard.” This gives our dashboard a clear and descriptive title. Before you can start building your dashboard, make sure you have a SQL warehouse connected, as displayed in [Figure 7-9](#).

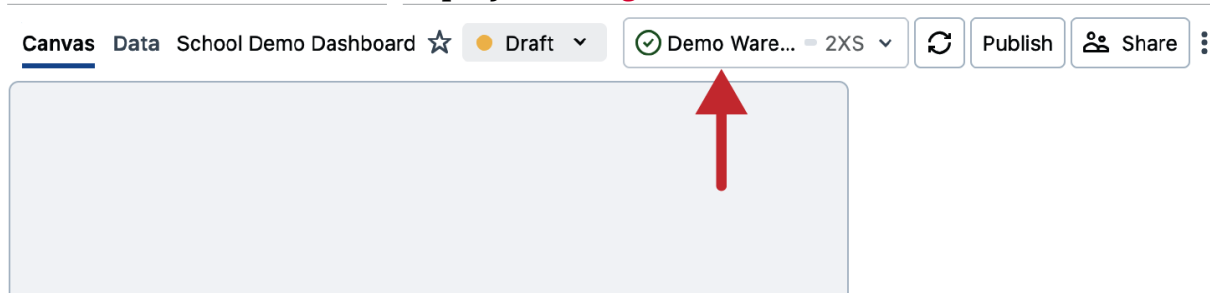


Figure 7-9. Connecting a running SQL warehouse to the dashboard

The dashboard’s functionality depends on the SQL warehouse, as it is responsible for retrieving data and displaying the visualizations.

Creating data sources

The Data pane in the dashboard editor interface allows you to define the source datasets for your dashboard, as displayed in [Figure 7-10](#). This is a crucial step in building a functional dashboard, as these datasets provide the input data for your visualizations.

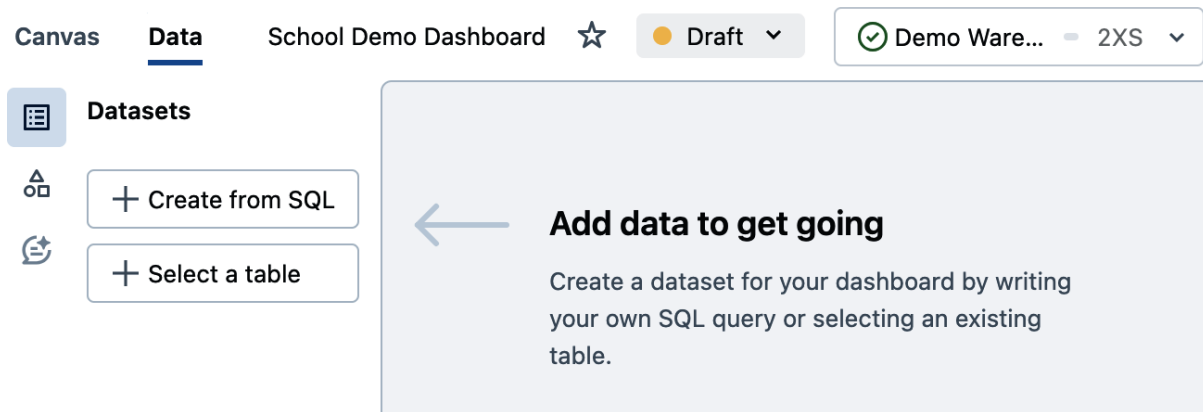


Figure 7-10. Dashboard Data pane

Each dataset is defined as either a SQL query or a table, which provides flexibility and power in shaping your data insights. Let's add a new dataset by clicking the “+ Create from SQL” button in the left panel. This opens a SQL editor to write our custom query, as shown in [Figure 7-11](#).

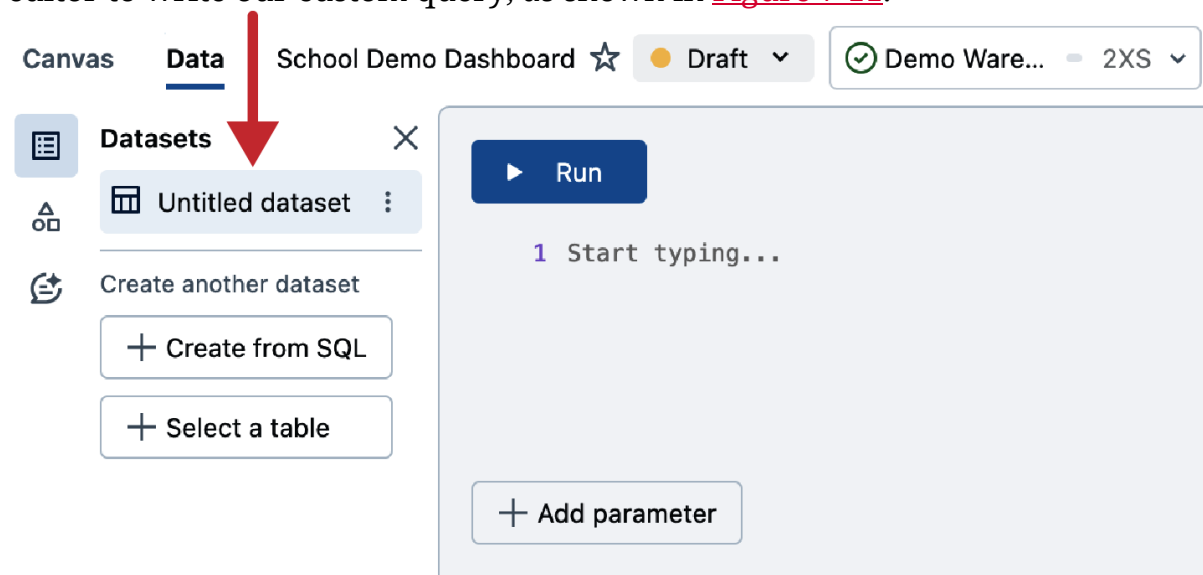


Figure 7-11. Dashboard Data pane

For this demonstration, we will write a SQL query that combines the two gold tables created previously by the “School” DLT pipeline. But before we dive into the query, let's rename the dataset to a meaningful name that can be easily identified in the Canvas pane. To do this, simply double-click the default dataset name (*Untitled dataset*) in the left panel and enter a new name, such as “Daily Student Courses.”

Now, let's write our SQL query that consolidates the school statistics data from two regions: the United Kingdom and France.

```
SELECT "United Kingdom" region, student_id, f_name, l_name,
       order_date, courses_counts
FROM hive_metastore.school_dlt_db.uk_daily_student_courses

UNION
```

```
SELECT "France" region, student_id, f_name, l_name,  
       order_date, courses_counts  
FROM hive_metastore.school_dlt_db.fr_daily_student_courses
```

After writing the query, click the Run button to view and verify the query output. This ensures that your dataset is correctly defined and ready for use in your dashboard.

Remember, you can add additional datasets depending on your requirements, allowing you to incorporate multiple data sources and insights into your dashboard. These datasets can also be queries that leverage functions during execution or provide advanced filtering logic, which can be very useful for gaining deeper insights into your data.

With our dataset created, we can now switch to the Canvas pane to start building our dashboard. This is where we will design and lay out our visualizations, leveraging the data from our newly created dataset.

Designing visualizations

The Canvas pane enables you to design your dashboard by adding visualizations and filters. From the toolbar at the bottom, you can pick a widget to add, including a visualization, text box, or filter, as illustrated in [Figure 7-12](#).

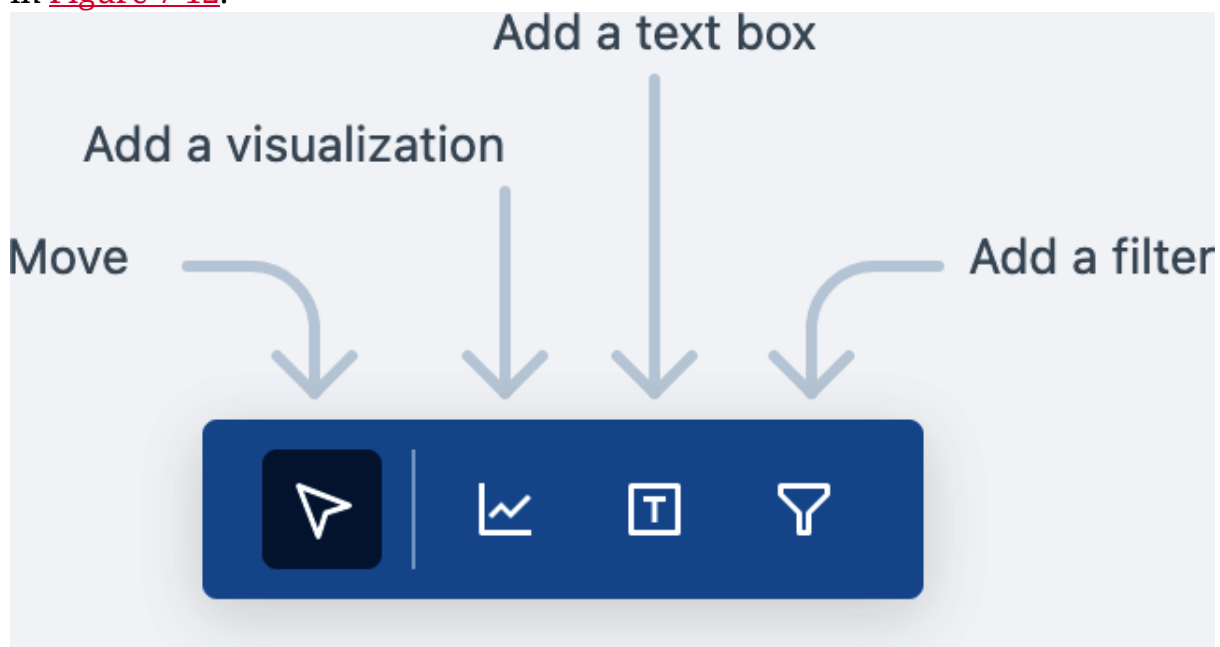


Figure 7-12. Toolbar in the Canvas pane

To create a visualization, simply pick the visualization widget and place it on the Canvas pane. Databricks SQL supports a wide range of visualization types, including area, bar, combo, counter, heatmap, histogram, line, pie, pivot, scatter, and table charts. This flexibility allows you to choose the most

appropriate and effective visualization type to communicate your data insights.

Once you've added a visualization widget to the Canvas pane, a configuration panel for this visualization will appear on the right side, as illustrated in [Figure 7-13](#). Here, you can specify the details of your visualization, including the dataset, visualization type, aggregation, and formatting options.

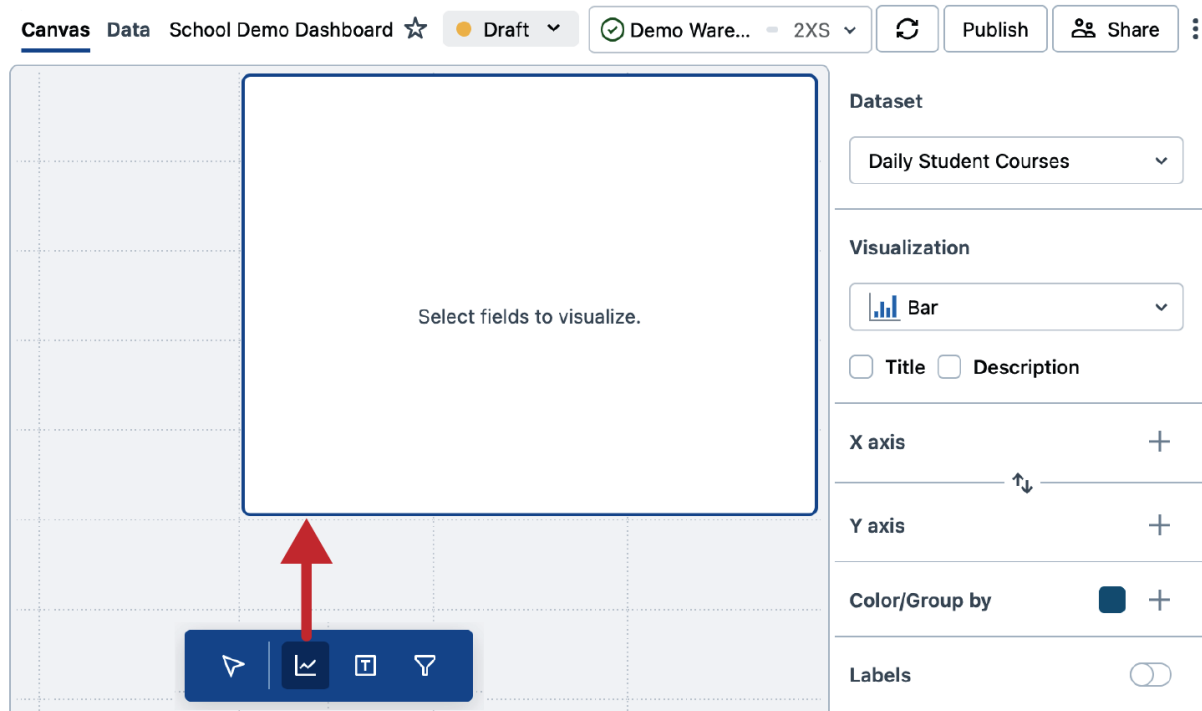


Figure 7-13. The configuration panel of a visualization widget

Let's configure this visualization widget to display a pie graph that shows the total course counts per region. To achieve this, complete the following steps:

1. **Selecting the dataset:** First, select the dataset "Daily Student Courses" that we created in the Data pane. This dataset contains the data we want to visualize.
2. **Choosing the visualization type:** Next, select the visualization type. In this case, we want to create a pie graph, so we choose the "Pie" option from the list of supported visualizations.
3. **Customize the visualization settings:** For a pie graph, we need to set the angle field and group by field. To do this, we do the following:
 1. Set the angle field to `SUM(courses_counts)` by clicking the plus (+) button and selecting the `courses_counts` field. This will default to the sum aggregation function, which calculates the total number of courses. If needed, choose an alternative aggregation function to match your application.
 2. Choose the "region" field as the color/group by field to display multiple categories in the graph.

After configuring these settings, you should see a pie graph that displays the total course counts per region, as illustrated in [Figure 7-14](#).

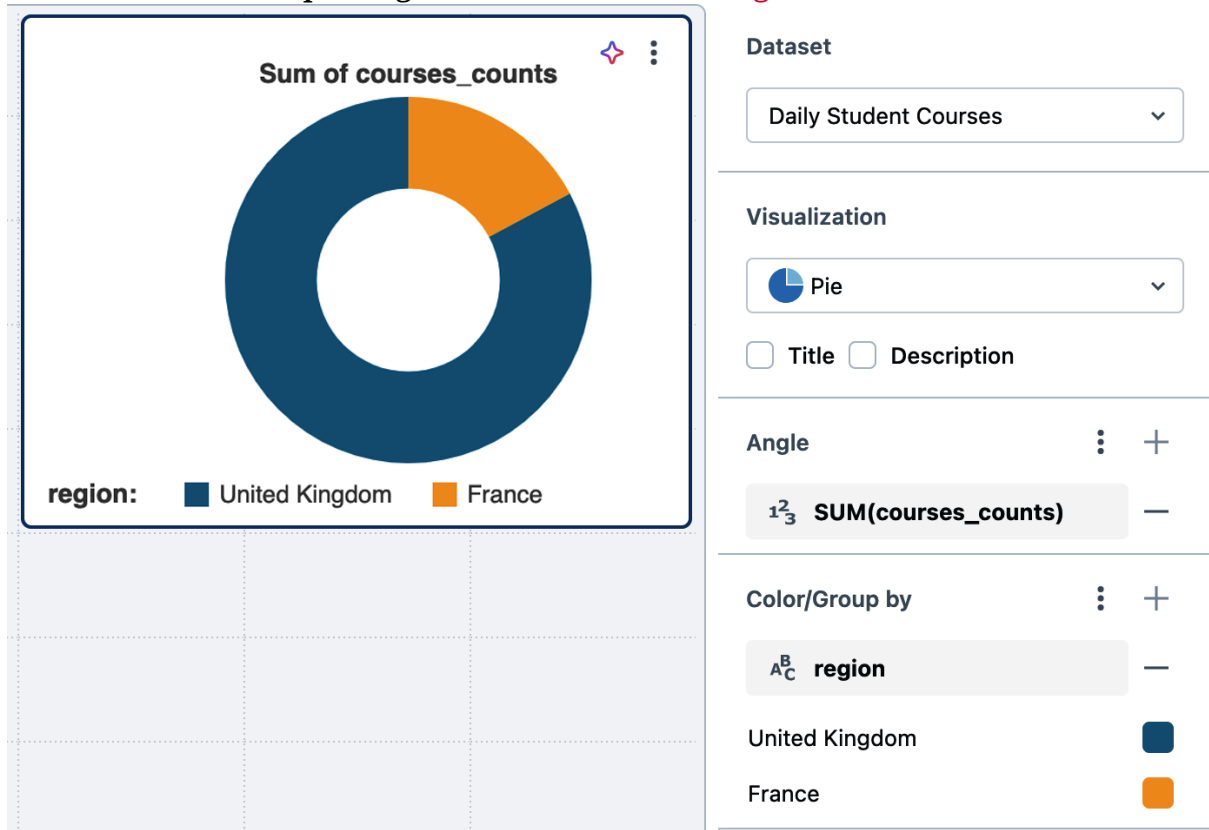


Figure 7-14. Pie graph for the total course counts per region

You can further customize this visualization by giving it a descriptive title, modifying its color scheme, and adjusting its size and position. This helps make the visualization more visually appealing and easier to understand for viewers.

In the same way, you can add other visualizations of different types depending on your business needs. By combining various visualizations, you can create a rich and informative dashboard that provides a comprehensive view of your data.

Defining filters

In addition to visualizations, Databricks SQL allows you to add dynamic filters to your dashboard. These filters enable viewers to refine the data presented in visualizations by filtering on specific fields. To add a filter to your dashboard, simply pick the filter widget from the toolbar at the bottom and place it on the Canvas pane, as shown in [Figure 7-15](#).

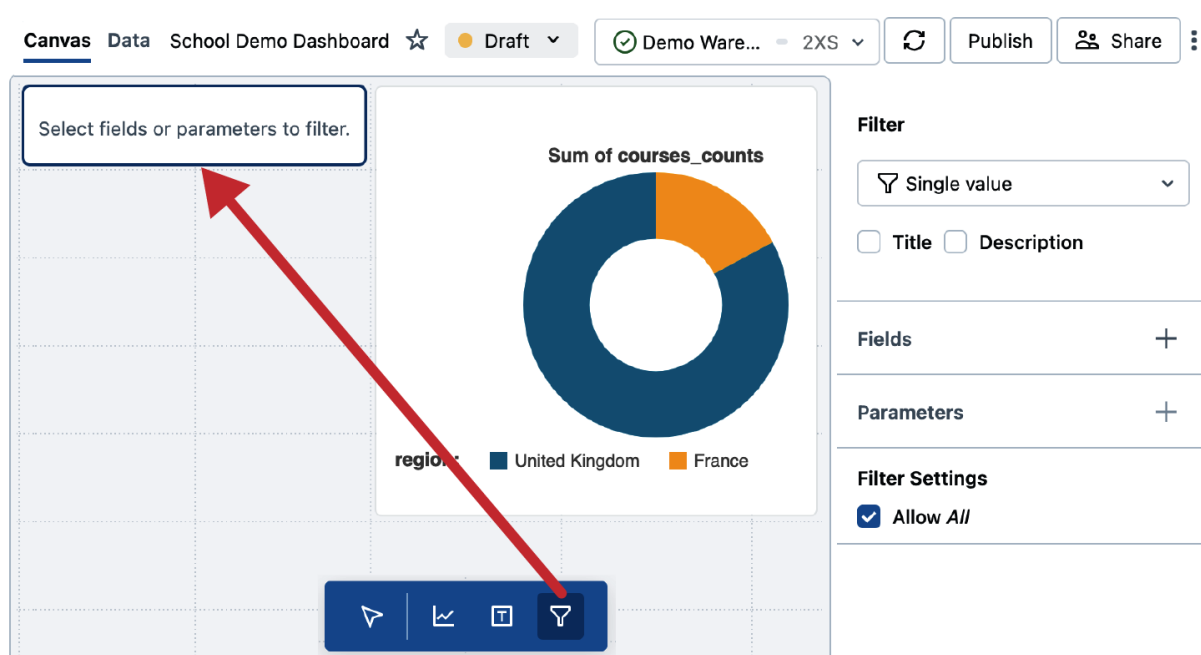


Figure 7-15. Adding a filter to the dashboard

Next, we need to configure the widget to filter based on a specific field—in this case, the “region” field. To achieve this, follow these simple steps:

1. Navigate to the right-hand panel of the selected filter widget.
2. Locate the Fields section and click the plus (+) button.
3. Select the “region” field.

Figure 7-16 displays the updated canvas after configuring the region filter.

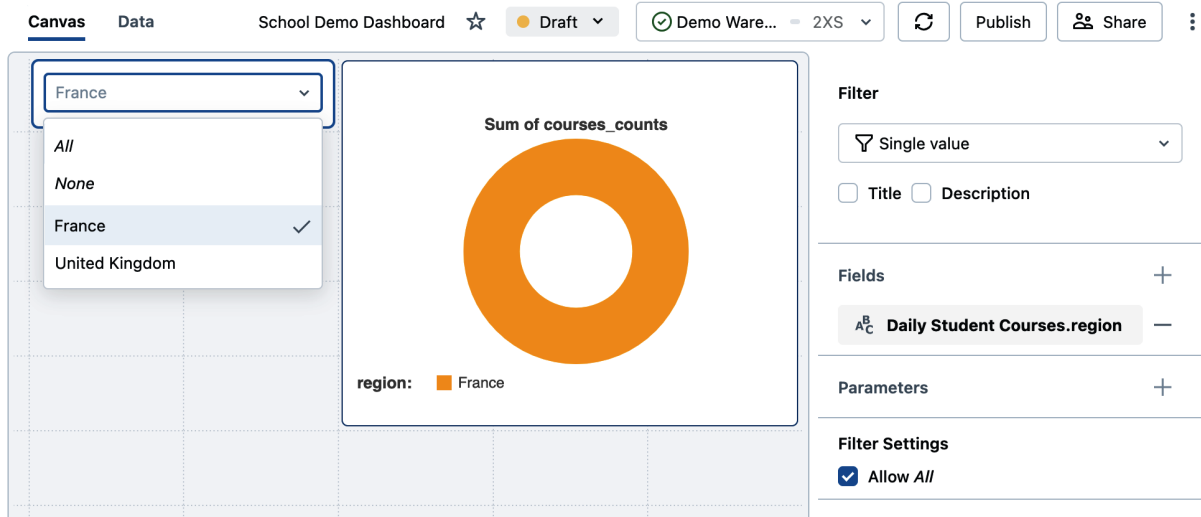


Figure 7-16. Dashboard view after applying the region filter

With this filter in place, our visualizations will update dynamically depending on the selected region. This means that when a user selects a specific region from the filter, the visualizations will automatically refresh to display only the data relevant to that region. This interactive filtering capability enables users to quickly and easily explore the data from different perspectives.

Sharing a Dashboard

When you create a new dashboard, it is initially saved as a draft, indicated by the Draft label in the top bar, as shown in [Figure 7-17](#). This draft status allows you to refine and finalize your dashboard before sharing it with others.

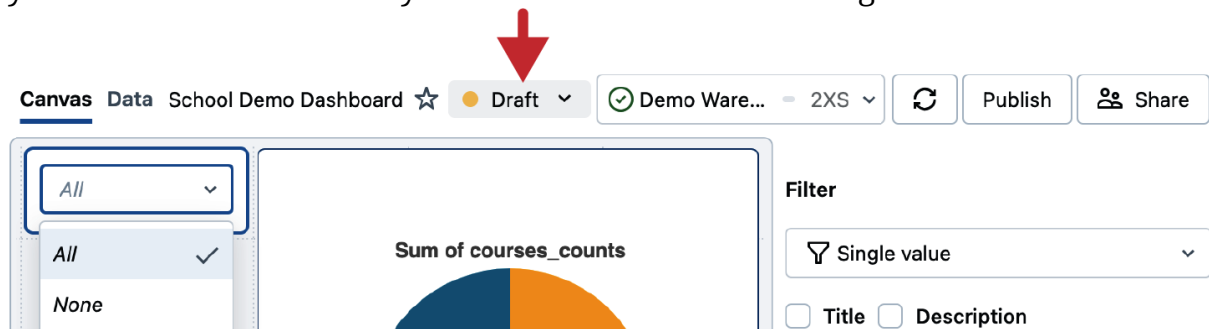


Figure 7-17. Draft version of the dashboard

You can share the draft with other users in your workspace to facilitate collaborative development and feedback. To share a dashboard, simply click the Share button at the top right of the dashboard editor. This opens a window where you can add users and set their permissions, allowing you to control who can manage, edit, run, or view the dashboard draft ([Figure 7-18](#)).

Sharing: School Demo Dashboard



Type to add multiple users, groups or service principals

People with access

Derar Alhussein

Can Manage (inherited)

Admins

Can Manage (inherited)

Sharing settings

Only people with access can view

[Copy link](#)

Figure 7-18. Setting permissions for sharing the dashboard draft

By assigning appropriate permissions, you can control the level of access each user has to your dashboard, ensuring a secure and controlled environment for collaboration. It's worth mentioning that when you share a dashboard draft, all users interact with the data and visualizations using their own credentials. This means that their individual permissions and access levels are applied, preventing unauthorized access to the underlying data.

Publishing a Dashboard

Once you're satisfied with your dashboard draft, you can publish it to create a clean copy that can be shared with any user in your organization. Publishing a dashboard allows others to view and interact with the visualizations and make data-driven decisions.

To publish your dashboard, click the Publish button at the top right, as displayed in [Figure 7-19](#).

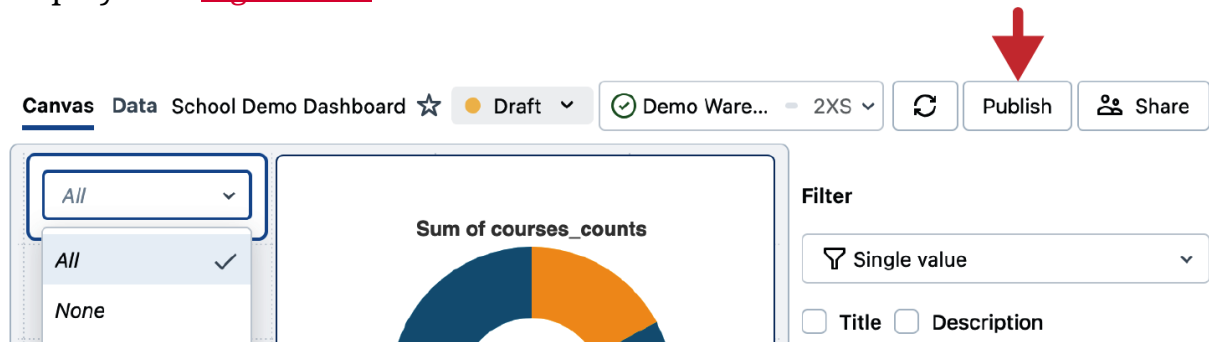


Figure 7-19. Publishing the dashboard

This will open a window that prompts you to choose how the published dashboard will access live data and run queries. You can select to use either your own embedded credentials or those of the viewing user, as shown in [Figure 7-20](#).

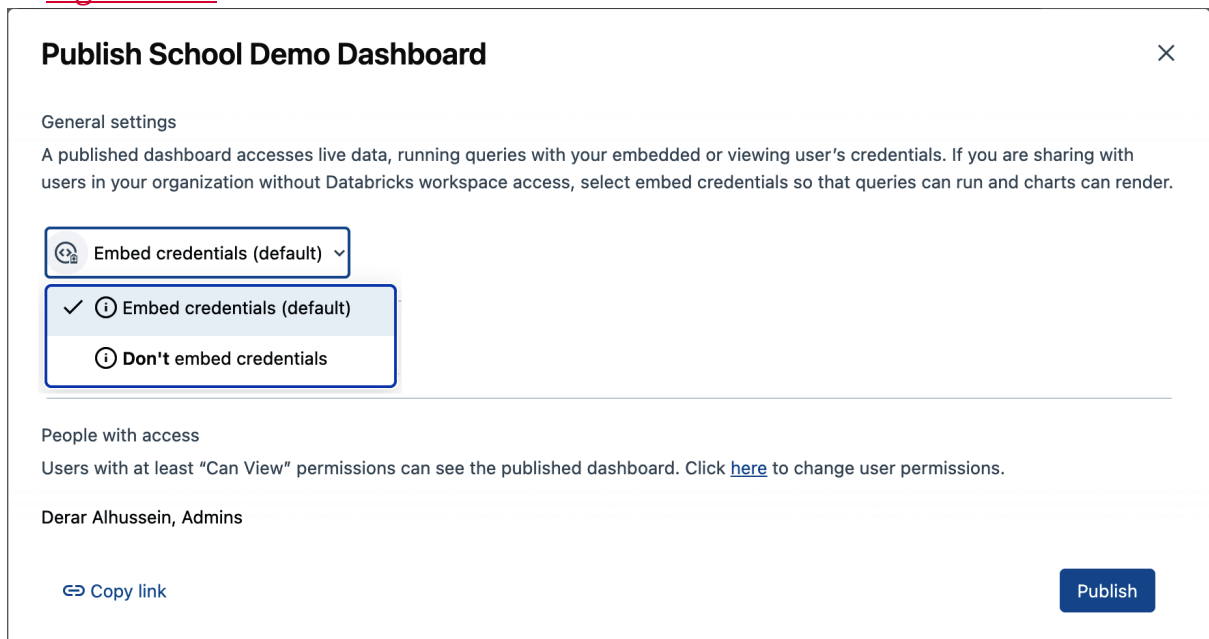


Figure 7-20. Publish options—selecting data access credentials

If you are publishing the dashboard for users in your organization without access to your Databricks workspace, select the option to embed credentials. This ensures that the dashboard's queries can run and visualizations can render properly for these users.

After publishing, you can view the live dashboard by clicking the Draft label at the top bar and selecting the published version, as displayed in [Figure 7-21](#). This indicates that the dashboard is now live and accessible to others via its published link.

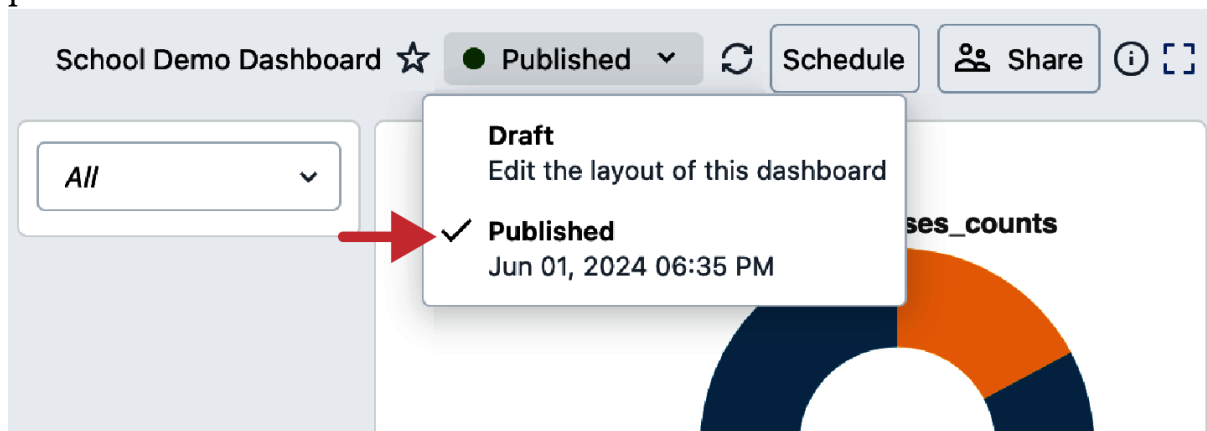


Figure 7-21. The published version of the dashboard

Republishing a New Version

The published version of your dashboard remains unchanged until you publish again. Meanwhile, you can continue making modifications and improvements in the draft version without affecting the publicly shared copy. This allows you to easily design and test a newer version of your dashboard before resharing it with your audience.

To switch from the published to the draft version, simply click the label at the top bar and select Draft, as illustrated in [Figure 7-22](#).

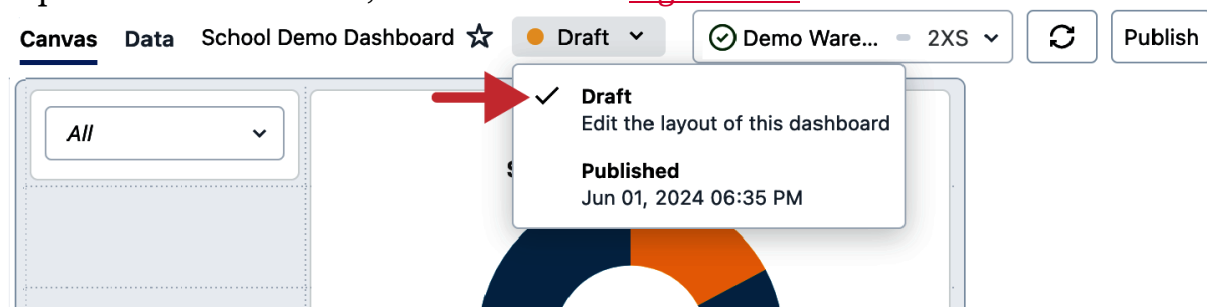


Figure 7-22. Switching between published and draft versions of the dashboard

By effectively managing draft and published versions, you can continuously improve your dashboards while providing a consistent experience for end users.

Managing SQL Queries

Databricks SQL provides a comprehensive solution for managing SQL queries, enabling you to create ad hoc queries, save them in a workspace directory,

and schedule automatic refreshes or alerts based on their results. This flexibility enables efficient data exploration and reporting.

To start creating SQL queries, let's navigate to the SQL Editor tab from the left sidebar of the workspace. This interface provides an integrated environment for writing, executing, and scheduling SQL queries, as illustrated in [Figure 7-23](#).

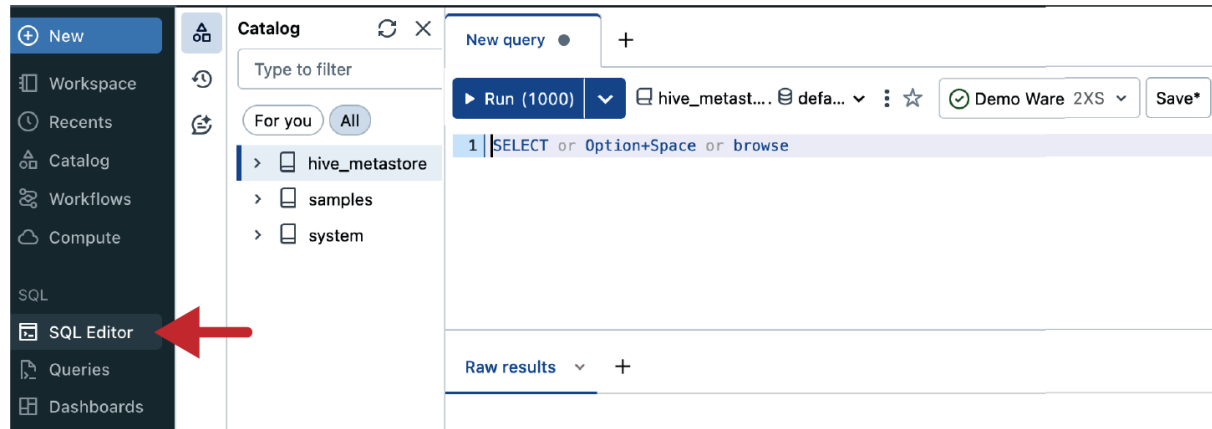


Figure 7-23. SQL editor in Databricks SQL

Before writing any queries, ensure you are connected to a running SQL warehouse, as illustrated in [Figure 7-24](#). This connection provides you with the computational power necessary for executing your SQL queries. If you attach it to a warehouse that has shut down, it will start automatically.

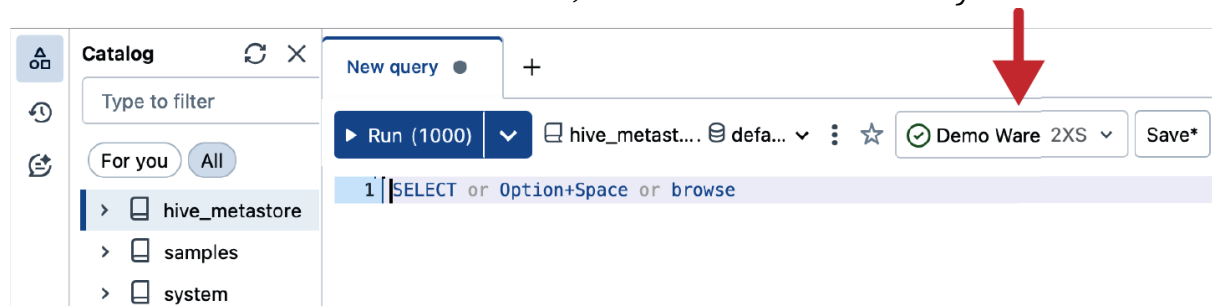


Figure 7-24. Connecting a running SQL warehouse to the SQL editor

On the left panel of the SQL editor, you'll find the schema browser where you can navigate your databases and tables. Expand the `hive_metastore` catalog to view the `school_dlt_db` database, which contains all the tables created by your DLT pipeline.

To view the columns of a table, simply expand the table in the schema browser, as shown in [Figure 7-25](#). This provides a detailed list of columns as well as some indicators for column types, enabling you to understand the structure of your data and write more informed and accurate queries.













- ✓  hive_metastore
 - >  default
 - ✓  school_dlt_db
 - >  fr_daily_student_courses
 - ✓  uk_daily_student_courses
 -  student_id
 -  f_name
 -  l_name
 -  order_date
 -  courses_counts
 - >  enrollments_cleaned
 - >  enrollments_raw

Figure 7-25. Exploring table columns in the schema browser

Writing a SQL Query

The SQL editor simplifies the process of writing queries by allowing you to interactively insert tables and columns into the query text. This feature allows you to quickly add the necessary objects to your query without having to type them out manually.

To insert a table or column into your query text, you can simply hover over the object name and click the double arrows button next to it, as illustrated in [Figure 7-26](#).

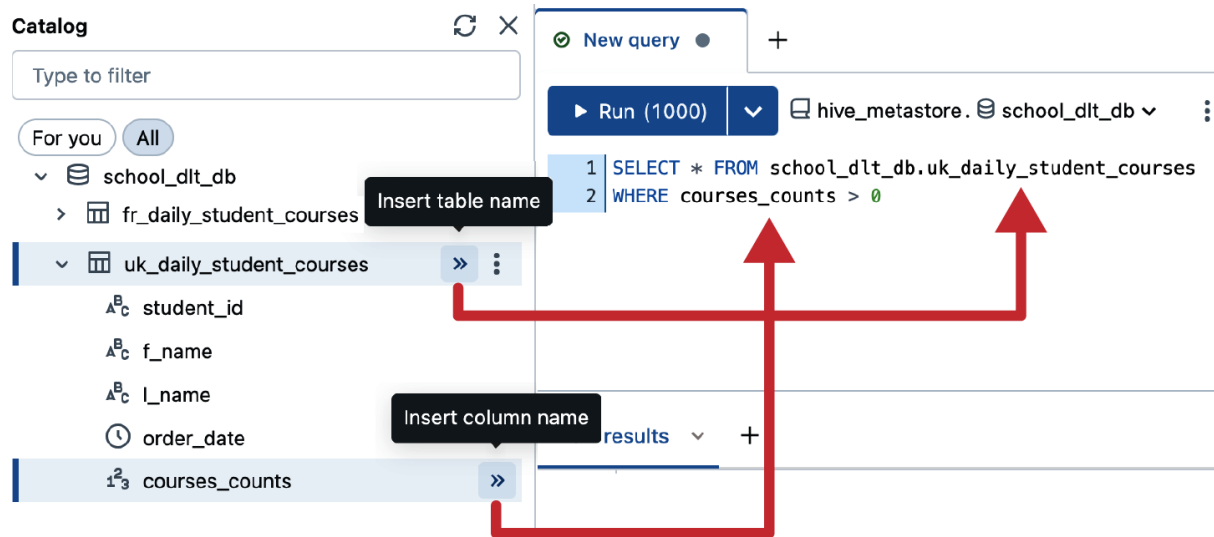


Figure 7-26. Inserting tables and columns into the query text

This convenient feature significantly speeds up query composition, ensuring accuracy and reducing manual input errors.

To illustrate the simplicity of writing queries in Databricks SQL, let's construct a basic query. Here is an example:

```
SELECT * FROM hive_metastore.school_dlt_db.uk_daily_student_courses
WHERE courses_counts > 0
```

Once the query is written, you can execute it by clicking the Run button in the SQL editor. The results will be displayed in the results pane below the editor. [Figure 7-27](#) showcases the results of our query, which returns 123 records that meet the specified criteria.

Raw results					
	student_id	first_name	last_name	order_date	courses_counts
1	S011				1
2	S007				2
3	S008				1
4	S008			2022-07-27T00:00:00.000	1
5	S005			2022-07-16T00:00:00.000	2
6	S010			2022-07-21T00:00:00.000	1
7	S00502	Elke	Feedome	2022-07-26T00:00:00.000	2

Figure 7-27. Query results in the SQL editor

From this pane, you can download the result set or create standalone visualizations, which can easily be added to new or existing dashboards.

Saving a Query

Once we have written and executed our query, we may want to save it for future use. Databricks SQL provides a convenient way to save our queries, making it easy to reuse them or share them with others.

To save our query, we simply click the Save button located at the top bar of the query editor. This will prompt us to give our query a name, as displayed in [Figure 7-28](#). In our case, let's name our query `school_stats`. Note that you can also name your query directly in the query tab, just like other objects in the workspace.

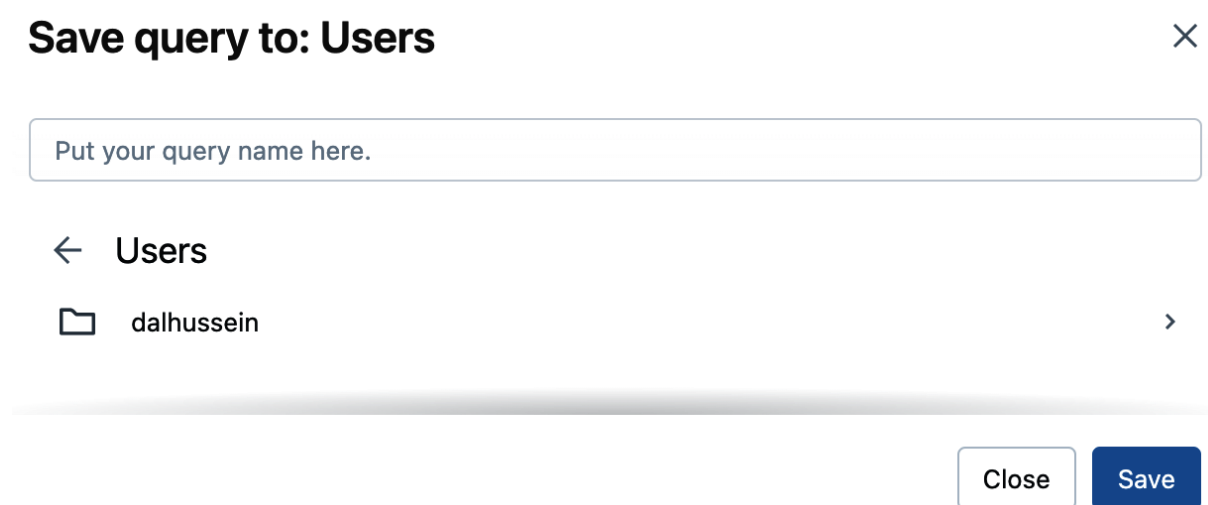


Figure 7-28. Saving a query in the SQL editor

Later in this chapter, we'll discuss how to effectively manage our saved SQL queries and leverage them to set up customizable alerts.

Scheduling a Query

Scheduling queries in Databricks SQL allows you to automate the refresh of query results, ensuring that data is up-to-date without manual intervention. This feature is particularly useful when working with dynamic data that changes frequently, as it allows you to stay informed about the latest insights and trends.

To add a schedule to our query, we click the Schedule button located at the top bar of the query editor and then click “Add schedule,” as shown in [Figure 7-29](#).

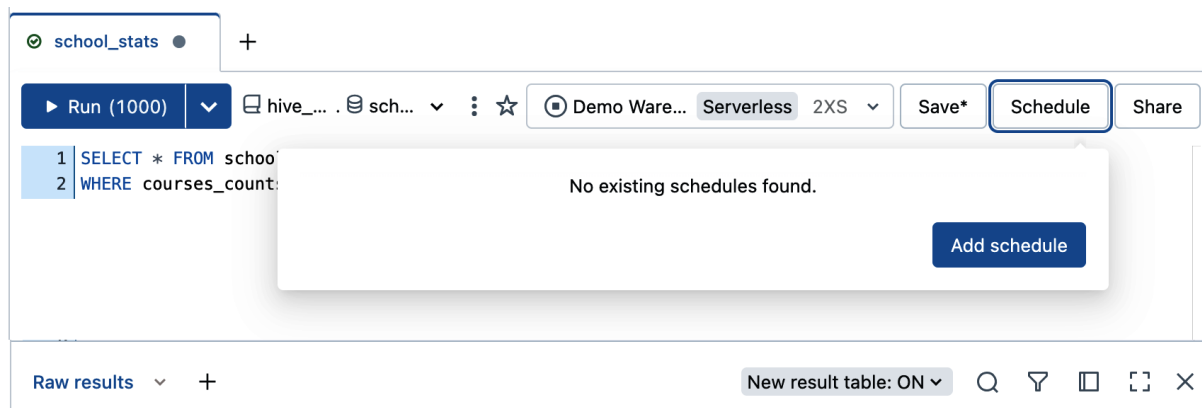


Figure 7-29. Adding a schedule to a query in the SQL editor

This opens a window with scheduling options, allowing us to specify when and how often the query should run, as illustrated in [Figure 7-30](#).

Figure 7-30. Query scheduling options

By scheduling queries to run at regular intervals, you significantly reduce the manual workload involved in maintaining up-to-date datasets. This is particularly useful for reports and other scenarios where fresh insights are critical or where queries may have long running times.

Browsing Saved Queries

Once we have saved our queries, we need a convenient way to access and manage them. Databricks SQL provides a dedicated interface for browsing saved queries, making it easy to find, organize, and reuse our queries.

To explore our saved queries, we navigate to the Queries tab on the left sidebar of the workspace, as shown in [Figure 7-31](#). This tab provides a centralized location for all our saved queries, allowing us to quickly find and access the queries we need.

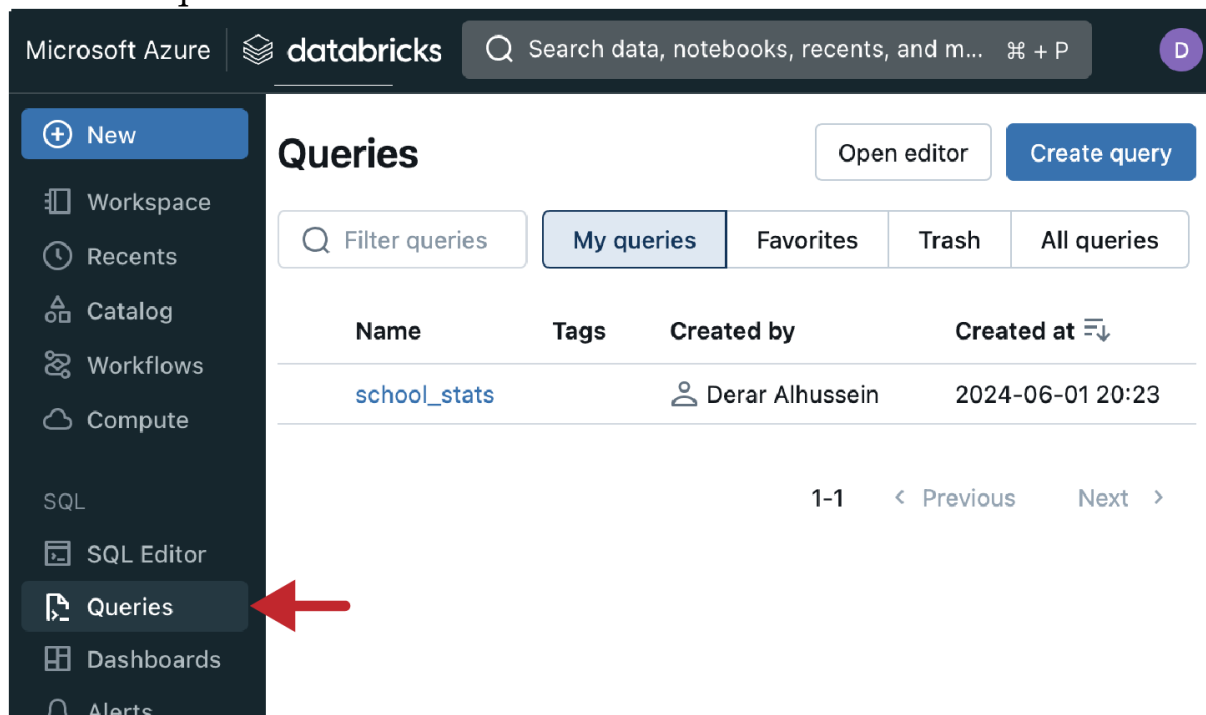


Figure 7-31. Queries interface in Databricks SQL

As shown, this tab displays a list of all our saved queries, along with relevant information such as the query name, creator name, and creation time. We can sort and filter the list to narrow down our search, making it easy to find specific queries or groups of queries.

At the top of the interface, there are also tabs for favorite queries we might reference frequently and trashed queries we want to review from time to time. Overall, the Queries page provides an organized and efficient way to manage our queries, allowing us to reuse our work and accelerate our data analysis tasks.

Setting Up Alerts

In Databricks SQL, we can take our data analysis to the next level by setting up alerts. Alerts are a powerful feature that enables us to monitor our data and receive notifications when reported values deviate from expected thresholds. By leveraging our saved queries, we can create alerts that periodically run queries, evaluate defined conditions, and send notifications when a condition is met.

In this demonstration, we will create an alert that notifies us when there is high demand for courses. This alert will be triggered when the number of course enrollments exceeds a threshold of 1,000, indicating high demand.

Creating an Alert

To create an alert, we navigate to the Alerts page from the left sidebar, as displayed in [Figure 7-32](#). This interface displays all previously created alerts and their most recent statuses, serving as a reference for managing alerts. From here, we click “Create alert” in the top-right corner to start configuring the new alert.

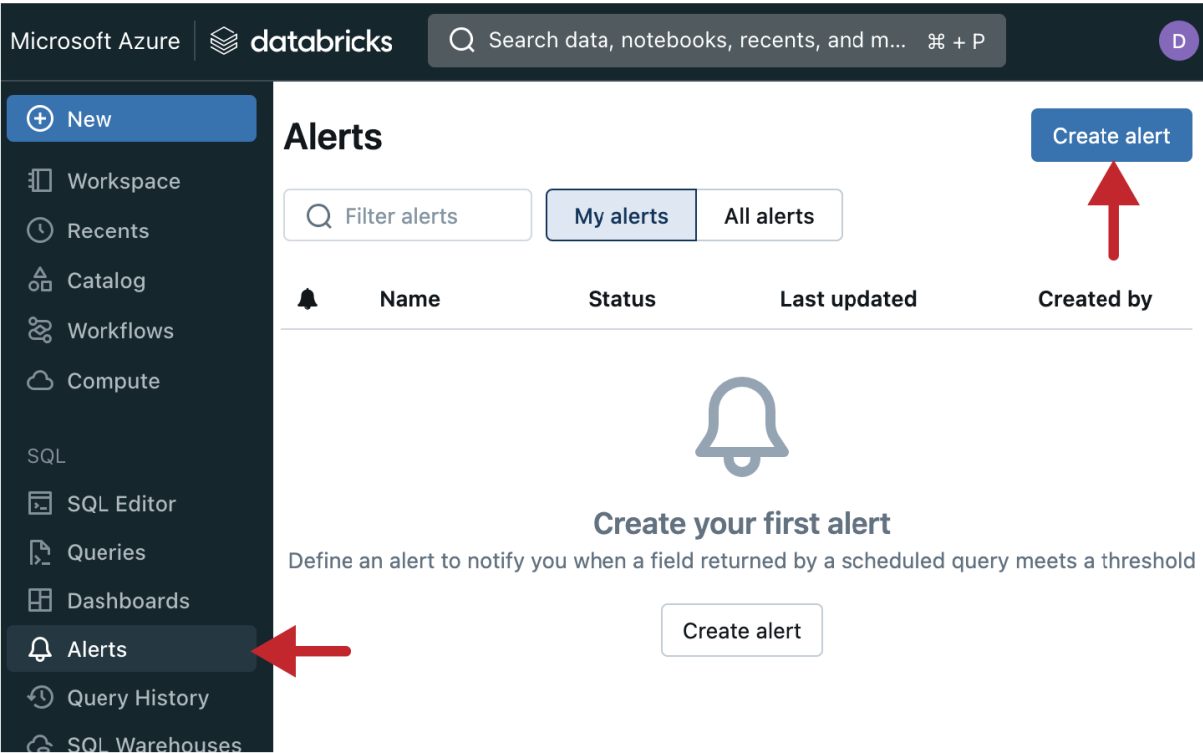


Figure 7-32. Alerts interface in Databricks SQL

First, let’s configure the basis details of the alert, as shown in [Figure 7-33](#).

New alert

Alert name

courses_high_demand

Query

school_stats

Create alert

Figure 7-33. Basic configurations of the new alert

Provide a descriptive name for the alert that reflects its purpose or the specific metric and source it monitors. For instance, let's name our alert `courses_high_demand`. Then, choose the saved query that will be executed to evaluate the alert condition. This query should return the data you want to monitor. In our case, let's choose our saved query titled `school_stats`. Next, let's proceed by defining the condition that will trigger the alert, as illustrated in [Figure 7-34](#).

Trigger condition

Value column

courses_counts ▾

Sum ▾

Operator

> ▾

Threshold value

1000

OK This alert would not trigger with the current data

SUM(courses_counts)	Threshold
328	1000

Template

Use default template ▾

Create alert

Figure 7-34. Condition configurations of the new alert

In this example, we set the value column to `courses_counts` with a threshold of a sum greater than 1,000. This means the alert will trigger if the total count of courses exceeds 1,000. Currently, the value in the table is 328, which is below the defined threshold.

Leave all other options at their [default settings](#), and click “Create alert” to finalize the creation process. [Figure 7-35](#) displays the newly created alert, showcasing the configured settings.

courses_high_demand ☆

UNKNOWN Derar Alhussein

Share

Edit

Refresh

⋮

Last triggered at None

Query

school_stats ↗

Trigger condition

Value column

courses_counts ▾

Sum ▾

Operator

> ▾

Threshold value

1000

Schedules

Destinations

No schedules found

Add schedule

Template Set to default notification template.

Figure 7-35. The `courses_high_demand` alert

Now that we've successfully created our alert, the next essential step is to set its schedule.

Scheduling the Alert

To ensure timely notifications, it is important to schedule how frequently you want to check for alerts. When you schedule an alert, it periodically runs its underlying query and evaluates the alert criteria according to this schedule. If the criteria are met, notifications are sent to defined destinations. Notably, this scheduling is separate from any other schedule that may be associated with the underlying query.

To add a schedule to our alert, we click the “Add schedule” link in the schedules section, as illustrated in [Figure 7-36](#).

courses_high_demand ☆

ShareEditRefresh⋮

UNKNOWNDerar Alhussein

Last triggered atNone

SchedulesDestinations

Queryschool_stats ↗

Trigger condition

Value column

courses_countsSum

Operator>Threshold value1000

No schedules found

Add schedule ←

TemplateSet to default notification template.

Figure 7-36. Adding a schedule to the `courses_high_demand` alert

This opens a window with scheduling options, allowing you to specify when and how often the alert should be evaluated, as shown in [Figure 7-37](#).

Add schedule



Settings Destinations

Schedule

Every 1



Hour



at

54 minutes past the hour



☐ Show cron syntax

Timezone

(UTC+00:00) UTC



More options



Cancel

Create

Figure 7-37. Alert scheduling options

After defining the schedule settings, you need to configure the notification destinations by clicking the Destinations tab, as illustrated in [Figure 7-38](#). You can set up notifications to be sent to multiple email addresses or to [system destinations](#) such as Slack, Microsoft Teams, or any webhook-based service.

Add schedule



Settings

Destinations



☒ Notify destinations when the alert is triggered

Search to add users or destinations



Destination



No Data

Cancel

Create

Figure 7-38. Alert destinations configuration

To add a destination, we use the drop-down menu to select an available notification destination or start typing a username from the workspace to add

their email address. This allows us to customize the notification recipients and ensure that the right users are informed when the alert is triggered.

Finally, let's click Create to save our changes and activate the alert.

Now, on the next scheduled execution, if the total value of `courses_counts` changes and exceeds the defined threshold of 1,000, the alert will be triggered, and we will receive a notification at the configured destinations.

By setting up and scheduling alerts in Databricks SQL, you can ensure that critical business metrics are continuously monitored, and you receive timely notifications whenever important conditions are met. This allows for prompt actions and informed decision-making.

Conclusion

In conclusion, Databricks SQL provides a robust data warehousing solution, offering a comprehensive suite of tools for efficiently managing and analyzing data. Through its intuitive interface and robust features, you can easily create scalable SQL warehouses, design informative dashboards, and manage complex queries. By mastering these skills, you can unlock the full potential of your data and drive business insights at scale.

Sample Exam Questions

Conceptual Questions

1. Which of the following compute resources is required to render a dashboard in Databricks SQL?

1. A job cluster.
2. An all-purpose cluster.
3. A SQL warehouse.
4. A BI engine.
5. None of the above! Databricks dashboards don't require compute resources to render visualizations.

2. A data engineer is setting up an alert in Databricks SQL to ensure timely action is taken when an anomaly is detected in a data flow. They want to notify the data operations team when this alert is triggered.

Which of the following notification destinations is supported in Databricks and can be configured by the data engineer in response to a triggered alert?

1. Webhook-based service.
2. WhatsApp.
3. SMS.
4. All of the above options are valid notification destinations in Databricks.
5. None of the above notification destinations is supported in Databricks.

The correct answers to these questions are listed in [Appendix C](#).