WHITE PAPER

Qlik Analytics and Databricks Best Practices Guide

Table of Contents

Summary

The purpose of this document is to cover the following aspects of Integration between Qlik Analytics solutions and Databricks:

- Qlik and Databricks synergy for advanced data and analytics pipelines
- Various architectural options for deploying Qlik and Databricks in tandem
- Exploration of high-level concepts and practical applications
- Guidance on choosing the appropriate approach based on scenarios
- Ongoing updates to align with emerging capabilities in Databricks and Qlik platforms
- Assumes reader's proficiency in both Qlik and Databricks technologies

Introduction

Qlik Analytics solutions set the benchmark for third-generation analytics platforms, empowering everyone in your organization to make data-driven decisions. Built on our unique Associative Engine, it supports a full range of users and use cases across the life cycle from data to insight: self-service analytics, interactive dashboards, conversational analytics, custom, and embedded analytics, mobile analytics, reporting and alerting. It augments and enhances human intuition with AI-powered insight suggestions, automation, and natural language interaction. Qlik Sense offers unmatched performance and governance, with the convenience of SaaS on-premises deployment, or both Qlik Sense is a complete data and analytics platform enabling users of all levels to explore data with agility and high performance. Databricks Data

Intelligence Platform is a cloud-based, massively scalable platform that provides effective management of enterprise-class data.

Qlik and Databricks together provide a balance to optimize the “speed of thought” exploration capabilities when Qlik’s associative engine with its powerful search and AI capabilities is combined with Databricks’ powerful and scalable database engine technology.

Blending the ideas of “getting the data you need when you need it” and “getting the data how you need it” with two cutting-edge technology platforms creates unique solutions that deliver enterprise analytics, reporting, dashboarding, and data science to the business.

Qlik Analytics Architecture

Qlik Analytics solutions architecture is designed with a flexible and scalable approach, ensuring efficient data processing and visualization capabilities. At its core, Qlik Analytics solutions follow an associative model, where data is loaded into an in-memory engine, allowing for swift and dynamic associations between different data sets. The associative model enables users to explore and analyze data intuitively, discovering insights across various dimensions without predefined hierarchies or limitations.

Additionally, Qlik Analytics solutions employ a hub-based structure where users access shared applications through a centralized hub interface. This collaborative environment fosters teamwork and ensures that insights derived from data are readily available across the organization. The platform's open APIs and extensibility further enhance its adaptability, enabling integration with other data sources, custom extensions, and external web content, making Qlik Analytics solutions a versatile and robust solution for modern data analytics.

Clients have the flexibility to adopt Qlik Analytics solutions based on their preferences and organizational requirements. One approach involves client-managed deployment, where customers take on the responsibility of installing and maintaining the infrastructure necessary to meet their analytics demands. This allows for a customized and controlled environment tailored to specific business needs.

On the other hand, an alternative option is leveraging Qlik Cloud, a Software-as-a-Service (SaaS) model. In this scenario, Qlik Cloud assumes the responsibility for all aspects of analytics services, encompassing security,

THE ASSOCIATIVE DIFFERENCE®

Relational databases and queries were designed in the 1980s for transactional systems, not modern analytics. Query-based tools leave data behind and limit your users to restricted linear exploration, resulting in blind spots and lost opportunities.

Qlik Sense runs on the unique Qlik Associative Engine, enabling users of all skill levels to explore their data freely without limitations. The Qlik Associative Engine brings together unlimited combinations of data — both big and small — without leaving any data behind. It offers unprecedented freedom of exploration through interactive selection and search, instantly recalculating all analytics and revealing associations to your user in green (selected), white (associated), and gray (unrelated). By keeping all visualizations in context together and retaining both associated and unrelated values in the analysis, the Qlik Associative Engine helps your users discover hidden insights that query-based tools would miss.

The Qlik Associative Engine is purpose-built for highly scalable, dynamic calculation and association of massive data volumes for large numbers of users. This unique technology is our primary advantage, with more than 30 years of innovation and investment.

scalability, and performance optimization. This offers a more streamlined and hassle-free solution, allowing organizations to focus on leveraging analytics insights without the burden of managing underlying infrastructure. The Qlik Cloud approach is particularly beneficial for those seeking a scalable and secure analytics solution without the complexities of infrastructure maintenance.

User Interfaces

Access to the Qlik Sense Enterprise SaaS environment is through a zero-footprint web browser interface (known as the Qlik Sense Hub). The Qlik Sense web browser interface makes all aspects of development, drag-and-drop content creation, and consumption possible. Qlik Sense features a responsive design methodology to automatically display and resize visualizations with the appropriate layout and information to fit the device — whether it is a browser on a laptop or desktop, tablet, or smartphone. Built with current standards of HTML5, CSS3, JavaScript®, and web sockets, Qlik Sense enables you to build and consume apps on any device.

In addition to the web-based interface, Qlik Sense supports conversational analytics which integrates with major chat platforms such as Slack and MS Teams, and data alerting capabilities to allow users to subscribe to and be notified of key changes to their data.

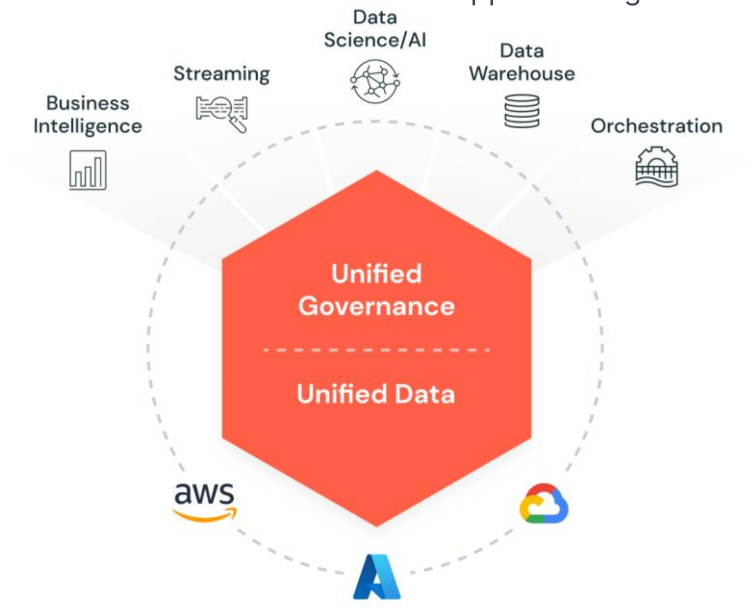
Qlik Mobile extends the power of Qlik's analytics platform to users on the go, providing a seamless and responsive experience across various mobile devices. The mobile application is designed to empower users with the ability to access, explore, and interact with Qlik Sense analytics on their smartphones and tablets. With Qlik Mobile, decision-makers can stay connected to critical insights and data-driven visualizations, making informed decisions anytime, anywhere.

The application ensures a user-friendly interface, optimizing the display of complex visualizations for smaller screens while retaining the interactive and associative nature of Qlik Sense. Qlik Mobile also leverages responsive design principles, adapting to different screen sizes and orientations to enhance the overall user experience. Whether in the office, during meetings, or while traveling, users can leverage the full capabilities of Qlik's associative analytics, fostering a mobile-first approach to data exploration and decision-making.

A quick reference to the entire Qlik Platform including Data Integration capabilities, cataloging, and extending Qlik Sense showcases the power of our integrated suite for Databricks.

Databricks Data Intelligence Platform

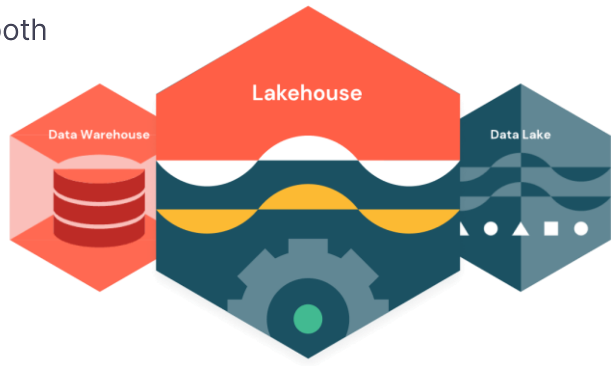
The foundation of the Databricks Data Intelligence Platform lies in lakehouse architecture, a revolutionary blend of data lakes and data warehouses. This innovative approach is geared towards minimizing costs and expediting the realization of data and AI goals.



Embracing open-source principles and adhering to open standards, the lakehouse architecture streamlines data infrastructure by removing the historical barriers that often complicate the realms of data and AI. By doing so, it offers a more cohesive and efficient environment for managing and leveraging your data resources.

Unified

A unified architecture encompassing integration, storage, processing, governance, sharing, analytics, and AI. A singular methodology for handling both structured and unstructured data. A comprehensive perspective on data lineage and provenance from start to finish. A cohesive toolkit accommodating Python and SQL, notebooks and IDEs, batch and streaming processes, across all major cloud providers.



Open

Within the Databricks framework, control over data is consistently maintained, ensuring independence from proprietary formats and closed ecosystems.

The foundation of the Lakehouse architecture relies on widely embraced open-source projects such as Apache Spark™, Delta Lake, and MLflow. It enjoys global support through the Databricks Partner Network. Additionally, the Delta Sharing feature presents an open solution for securely sharing real-time data from the lakehouse to any computing platform. This is achieved without the need for data replication or intricate extract, Transform, Load (ETL) processes.

Scalable

The automatic optimization for performance and storage is meticulously designed to ensure the lowest Total Cost of Ownership (TCO) among data platforms, concurrently achieving world-record-setting performance for data warehousing and Mosaic Artificial Intelligence (AI) use cases. This extends to the application of generative techniques such as Large Language Models (LLMs).

Irrespective of organizational scale, Databricks is engineered to effectively address the operational requirements of businesses, ranging from startups to global enterprises.

High-Level Qlik & Databricks Integration Options

Qlik and Databricks offer a variety of integration capabilities suited for specific scenarios to maximize resources, maintain performance, and fulfill business requirements across the organization. The number of concurrent users, data refresh frequencies, performance, user experience, and total cost will all contribute to deciding which integration options to take advantage of. Qlik provides several ways to load and consume data from Databricks. The table below describes each integration configuration, feel free to use this as a reference guide.

Integration Option	Method	Use Case
In-Memory	Full / Incremental Load (QVD)	Most common and best performing. Batch reloads leveraging a QVD for incremental updates and optimized for analytics engine.
In-Memory + Databricks	Full / Incremental Load (Change Data Feed)	Databricks optimized reload – especially useful when data needs to be compared to current vs historical state and/or for incremental reloads
On-Demand	ODAG	Structured Drill to Detail for access to large data volumes
On-Demand	Dynamic Views	Supporting details on demand inside the existing app as visualization objects

Summary of Qlik & Databricks Usage Options

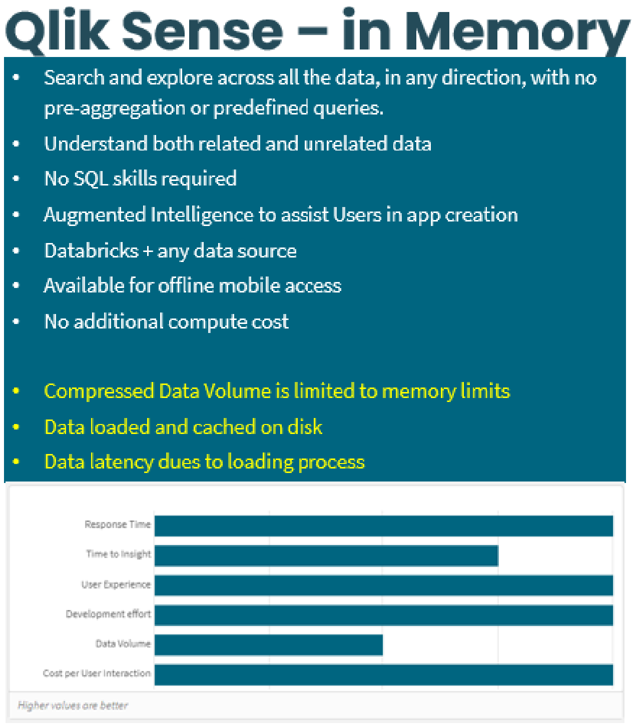
In-Memory options for how to load data into Qlik Sense:

1. Full reload every time on a schedule:

- Could lead to long load times depending on data volume.
- Recommended if the data is highly volatile or has a high number of changes.

2. Incrementally load only new data:

- Initiate delta reloads on a predefined schedule: This functionality can be configured by a Qlik developer within the Extract, Load, Transform (ELT) script. It enables Qlik to exclusively update the modifications in the data. Employing this method facilitates a quasi-real-time refresh of data within the Qlik engine and visualization layer.



Qlik Sense – in Memory

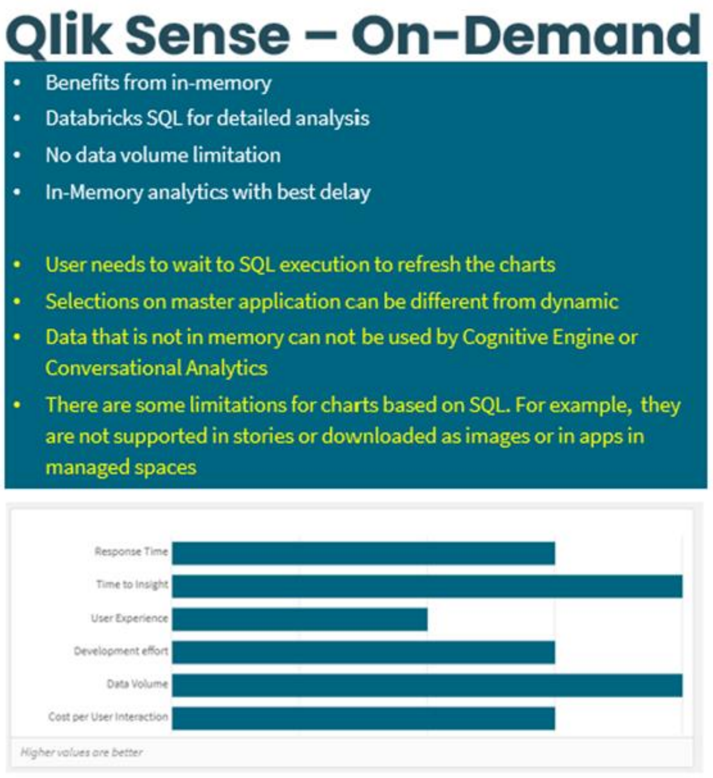
- Search and explore across all the data, in any direction, with no pre-aggregation or predefined queries.
- Understand both related and unrelated data
- No SQL skills required
- Augmented Intelligence to assist Users in app creation
- Databricks + any data source
- Available for offline mobile access
- No additional compute cost

- Compressed Data Volume is limited to memory limits
- Data loaded and cached on disk
- Data latency dues to loading process

On-demand Apps:

1. On Demand App Generation (ODAG)

- On-Demand Detailed Data: This approach is commonly employed in scenarios involving large datasets, where a Qlik app is designed to house summarized data. A Qlik detailed app serves as a template, receiving parameters from the summary app and dynamically querying Databricks for live data. Users can then access the relevant data slice based on their selections. This method is effective for conducting summary-detail analytics.



Qlik Sense – On-Demand

- Benefits from in-memory
- Databricks SQL for detailed analysis
- No data volume limitation
- In-Memory analytics with best delay

- User needs to wait to SQL execution to refresh the charts
- Selections on master application can be different from dynamic
- Data that is not in memory can not be used by Cognitive Engine or Conversational Analytics
- There are some limitations for charts based on SQL. For example, they are not supported in stories or downloaded as images or in apps in managed spaces

2. Dynamic Views

Live Data Visualizations: Employed for scenarios requiring live or near-real-time data in the Qlik app. Utilizing the ODAG framework, live data updates are triggered based on user interactions, reloading data from Databricks in real time when users choose to update their selections.

3. Direct Query applications

Direct Query is a feature in Qlik Cloud Analytics that allows analytics apps to generate SQL queries directly against cloud databases in real time as users interact with visualizations and apply dynamic filters. It eliminates the need to preload data, with no restrictions on data volumes, and performance is dictated by the underlying cloud database.

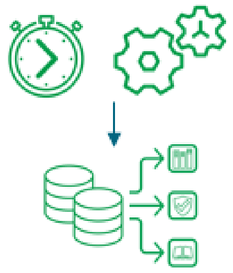
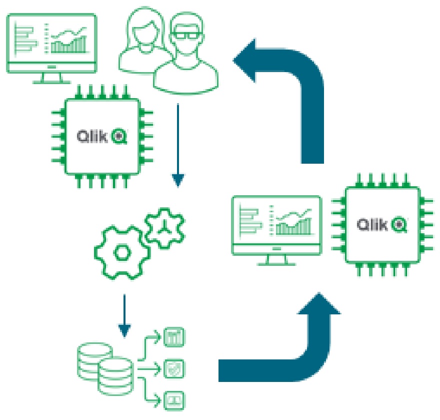
4. Machine Learning integration

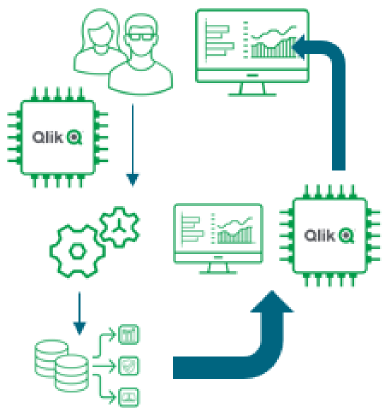
Qlik Cloud Analytics provides analytic connections enabling the establishment of a connection between the Qlik associative engine and third-party machine learning models. This involves creating connections that expose REST-based API endpoints, allowing the seamless exchange of data between Qlik and external machine learning models. Analytic connections provide a bridge for leveraging the power of machine learning within the Qlik Cloud Analytics environment.

Deep Dive Databricks & Qlik Method Decision Tree

In many organizations, there are well-established data pipelines and a user base seeking access to data and insights. The section below aims to provide an overview of various integration options available in Qlik Sense when deploying alongside Databricks, helping you make informed decisions based on your specific requirements.

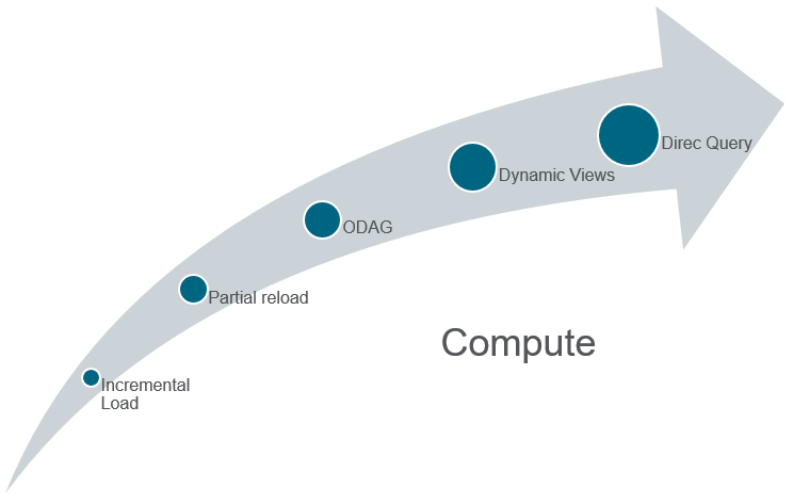
Deciding on the optimal integration option involves considering factors such as the total number of concurrent users, system performance thresholds, costs, and specific business requirements. To guide our discussion, let's first establish a high-level understanding of each Qlik integration option before delving into their potential applications in various business use cases.

Incremental Load (Scheduled Data Load) Cached In-Memory	An Incremental Load is a scheduled process in which Qlik loads the latest data for required tables, merges change with existing QVDs (previously loaded data/cache), and rewrites those QVDs/Cache. This ensures all applications and users can access the updated information for those tables. Particularly effective with large data sources, this approach strikes a balance between freshness and speed, loading only the changed data on a predefined schedule after the initial bulk load. 
Partial Reload Cached In-Memory + New Data Added In-Memory (On-Demand or Scheduled)	In a full or incremental load, all tables in the existing data model are deleted, and the load script is executed. Conversely, a partial reload retains all tables in the data model, executing only Load and Select statements with an Add, Merge, or Replace prefix. This approach loads changed data for a subset of tables into the Associative Engine, offering a balance between freshness and speed. Users initiate the action, boosting confidence in viewing the most recent values, especially when the data source provides details about changes (inserts, updates, and deletes).
On-Demand Application Generation - ODAG (Live Data Load) Cached In-Memory For selections + Live Data based on Selections in the second	On-Demand Application Generation is a process where end users provide their selections to a pre-defined template application. Qlik copies the template, loads data for the specific cohort, and presents it as a new application to the end user. The calling application, fully in-memory with aggregates, enables rapid actions, while the spawned application retrieves live details only if and when needed. 

application when the user asks	
Dynamic Views (Live Data Load) Cached In-Memory For selections + Charts from Live Data based on Selections when the user asks	Dynamic Views is a process similar to ODAG but instead of surfacing a new application to the end user, selected charts from a template application are displayed in the application they are currently using. If the user changes their selections they can ask for the Dynamic Views to be refreshed.  This option capitalizes on the functionality of ODAG, while also rendering the "live" cohort of data in the context of the application where the users made their selections.
Direct Query	You may use this approach when your data volumes are huge or change too often which would make it difficult to load the data into the Qlik Engine
Analytics Connections	When you need to integrate with ML models deployed in the Data Intelligence Platform. This option can integrated at the script level (scheduled reload) or during the visualization at the chart level (real-time)

So which one is right for YOU?

As a primer to choosing the right solution for the right problem, let's begin with some very high-level guidance starting with a focus on how much computation and thus cost is driven by each of the solutions. In many cases the difference between "We WANT Live Data" and "1 minute old will be FINE" may come down to the implementation cost. In another use case, Live Data is a MUST-have regardless of the computing resources and cost associated with solving the problem. The following chart contains no scale on purpose. It is simply to portray the concept that each technology will have increasing amounts of computing requirements associated with it.



Asking the question "Why does each solution increase the amount of compute/cost?" Well, doing an Incremental Load means that Qlik only asks for the changes from our source 1 time per scheduled reload. On Demand Application Generation (ODAG) applications will load all of the data for a defined cohort even if it hasn't changed in years. Dynamic Views require the same data loading as ODAG applications, and as they can be refreshed as often as desired by the end user to keep in sync with their selections they require more computing power. From one usability perspective, Dynamic Views tend to be more used than ODAG since the user sees the app as only one data source. The user is not always aware that SQL queries are executed compared to ODAG when the user hits the “generate new app” button. And using Direct Query all the queries are pushed down to the Databricks Data Intelligence Platform for every selection the user might use.

Remember though, that was only a general guide. It is entirely possible that one of your business use cases only requires a single SQL for an aggregate in an application that is only used by a small set of end users. This brings us to the next point: Business-driven use cases are often complex and require the answers to many questions.

The following scenarios walk through some hypothetical use case that lines up with each of the architectural solutions.

Scenario 1 – ODAG: Drill to Detail Reporting/Analysis

Starting with a summary application of key metrics, a user chooses a selection of criteria which then is passed to a secondary application that is generated on demand. A customer example of when to use ODAG:

A Human Resources department has an existing application that is used by 1,290 HR administrators, managers, and employees throughout the company who make a lot of selections. The application is currently on a one-hour incremental load schedule. They want to use real-time data instead of waiting for reloads to see potential overtime issues. The details are a known subset of content in a specific format that shows potential issues, therefore ODAG provides an interim reload ability in an easily consumable detailed Qlik application.

Scenario 2 – Dynamic Views: Transactional Details in Context

This scenario is best suited when details for specific transactions need to be viewed in context with the original Qlik Sense application as a chart inside the application. A customer example of when to use Dynamic Views:

A customer has several hundred billion records from a transactional system stored in Databricks which is too much for a single Qlik application. This data is reloaded on a schedule with the aggregates of KPIs and other relevant data but not the transactions themselves. Dynamic Views are used to get the transactions of a cohort of dimensional values that limit the records to a threshold (say under 100k rows) to be analyzed in any chart on demand. The users of the application need to be able to see the details in the context of the cohort selected.

Scenario 3 – Partial Reload/Merge: Closing Books / Financial Reporting

Finance leaders closing the books at the end of the month need access to up-to-the-minute general ledger details. The Qlik application contains very complicated calculations, hierarchies, and transformations not easily replicated with SQL. The application will only be used by Accountants and Sr Executives in the corporation which equates to about 25 people. Data is changing rapidly throughout the close cycle and the users need to see where the company stands at any point in time. End users have a very high level of interactivity and are required to analyze the data to find issues.

Performance Considerations / Best Practices by Technique

As previewed, Qlik and Databricks together offer a range of ingestion techniques with varying levels of loading data into memory with Qlik's analytics engine and purely loading near/real-time data. There are clear strengths and advantages for each technique in addition to some having limited use cases.

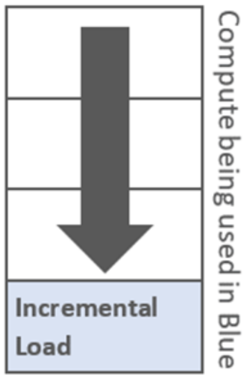
It's necessary to understand the caching process between Databricks and Qlik to distinguish how to optimize each system to support your analytics needs.

Incremental Load Options

The following sections will discuss in detail the above strategies on how to leverage Databricks data depending on the use case.

1. Incremental Load using QVDs

This is a prevalent technique in Qlik environments to efficiently load only the changed or new data. QVD, a binary and optimized data format, allows for snapshotting, data structure reuse, and incremental loading. The process involves:



1. identifying a sizable table
2. loading all data initially and storing it as a QVD file.
3. Subsequent loads involve identifying new records based on a specific column
4. merging the new data from Databricks using a WHERE clause with the identified variable
5. overwriting the existing QVD.

This method optimizes data reloading, enhancing performance and enabling snapshot-based data analysis. For a detailed explanation, refer to this Qlik Community Post: https://community.qlik.com/t5/Qlik-Design-Blog/Qlik-Sense-Incremental-Load-using-Merge/ba-p/1944225

2. Incremental Load using Partial Load

A full reload begins by removing all tables in the current data model and then executes the load script. In contrast, a partial reload retains all tables in the data model and only executes Load and Select statements with an Add, Merge, or Replace prefix.

Partial Load is helpful when you have new data to append to an existing table but do not want to load the rest of the tables. This is helpful when you have one table with a lot of real-time chances and a large data set from many sources in the rest of the data model.

Statement	Full reload	Partial reload
Load ...	Statement will run	The statement will not run
Add/Replace/Merge Load ...	Statement will run	Statement will run
Add/Replace/Merge Only Load ...	The statement will not run	Statement will run

Examples:

ADD LOAD * from Databricks.FactTable where createdatetime > $(lastreloaddatetime);

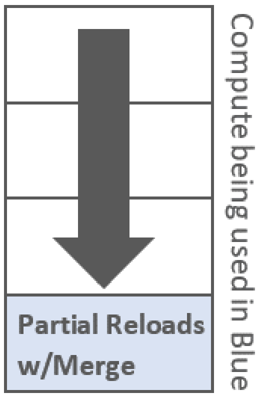
The above will run when the “Partial” option has been set on a reload AND will run when the partial flag is not set.

ADD ONLY LOAD * from Databricks.FactTable where createdatetime > $(lastreloaddatetime);

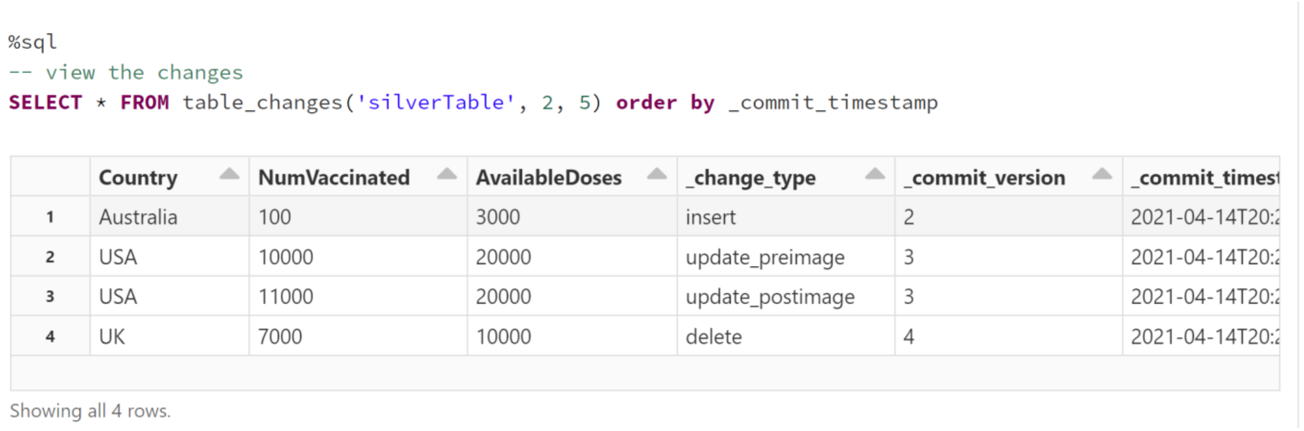
The above will run ONLY when the Partial flag has been set and will *not* be run when partial is set to false in the reload.

Change data feed and MERGE

Traditionally Qlik developers have utilized a Last Modified Timestamp field so that they could load only the changed values. However, there are many times that data tables don't have a Last Modified Timestamp so then Full Reloads were the only option.



Databricks offers a Change data feed. It's the ability to simply get the Change Data Capture information returned in a query. When it is enabled (by table), it adds metadata columns behind the scenes to your tables that allow you to modify your typical SQL Selection using a "where" clause to retrieve only changes since the last execution.



```
%sql
-- view the changes
SELECT * FROM table_changes('silverTable', 2, 5) order by _commit_timestamp
```

	Country	NumVaccinated	AvailableDoses	_change_type	_commit_version	_commit_timest
1	Australia	100	3000	insert	2	2021-04-14T20:
2	USA	10000	20000	update_preimage	3	2021-04-14T20:
3	USA	11000	20000	update_postimage	3	2021-04-14T20:
4	UK	7000	10000	delete	4	2021-04-14T20:

Showing all 4 rows.

We can ignore records where _change_type = 'update_preimage' since we are only interested in the new values from updated records.

Combining this CDC capture information from Databricks with the Qlik Merge function offers a highly performant way to handle Incremental Loads. The Merge function provides a simple way of automatically "merging" changes into an in-memory table. You simply pass the function a few parameters and the results of the Databricks Changes query for the table.

An example where Change Data Feed is enabled in a table

```
[silvertable]: // full load
SELECT
   Country,
    NumVaccinated,
    AvailableDoses
FROM
   `default`.silvertable;
```

```
If IsPartialReload() then
  //Sets a default starting point in time if this is the very first time
  Let Last_commit = Alt(chr(32)&Timestamp(Last_commit,'YYYY-MM-DD hh:mm:ss.ff')&chr(32),1);
  // Incremental load
  Merge only (_commit_timestamp, Last_commit) on Country Concatenate(silvertable)
  SELECT
     left(_change_type,1) as Operation, // We need only the first character
     Country,
     NumVaccinated,
     AvailableDoses,
     _commit_timestamp
  FROM table_changes('silverTable', $(Last_commit))
  WHERE _change_type !='update_preimage'
  ORDER BY _commit_timestamp;

  Drop field _commit_timestamp from [silvertable];//Optional
End if
```

This functionality:

1. Speeds up traditional Incremental Load applications.
2. Can be used to handle Incremental Loads for tables without Last Modified Timestamp columns where you may be doing full reloads.
3. Partial Reloads allow end users to quickly bring the latest and greatest changes very quickly into memory so that they can analyze them.

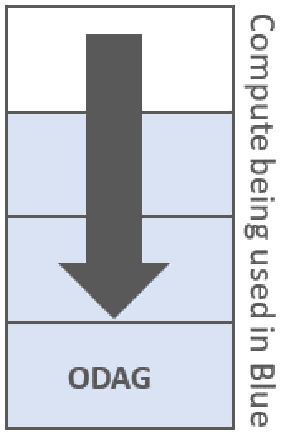
Details about Databricks Change Data Feed can be found here: https://docs.databricks.com/delta/delta-change-data-feed.html

Details about the Qlik Merge function can be found here: https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/Scripting/ScriptPrefixes/Merge.htm

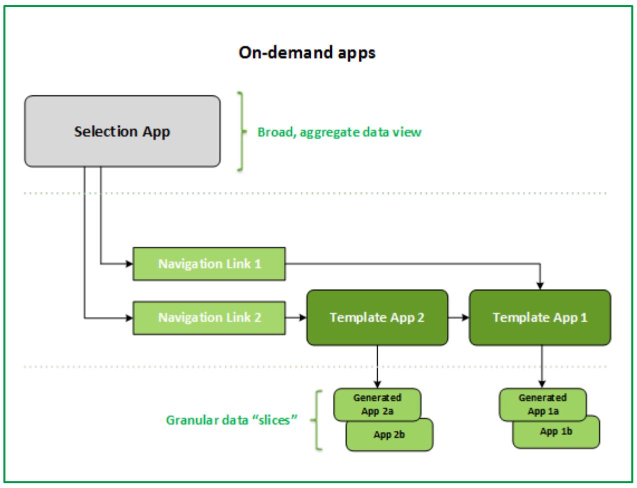
On-Demand Options

The following sections will discuss in detail the strategies to leverage real/near-time data from Databricks.

On-Demand App Generation (ODAG)

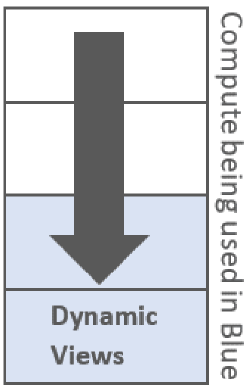


Often referred to as ODAG, this method involves bringing in a specific subset of data for analysis, particularly in scenarios with substantial data volumes where loading all data into a Qlik app may be impractical. This approach is commonly employed when the fact table exceeds 500-800 million rows (a rough estimate depending on various factors). For instance, in a "shopping cart app," users initially explore high-level aggregated data and then refine their analysis by making selections. Once they confirm the manageable dataset with desired parameters, they can transition to the "analysis app," pre-defined with chart objects and layout, or load the data into a blank sheet if preferred.



Here is the official help document for creating and managing On-Demand Apps: https://help.qlik.com/en-US/sense/May2022/Subsystems/Hub/Content/Sense_Hub/DataSource/Manage-big-data.htm

Dynamic Views



Dynamic views empower users to actively manage the analytic sources they wish to investigate and control when data refreshes in visualizations.

These views allow users to query and display specific subsets of extensive datasets in charts that dynamically update with each selection. This functionality ensures that visualizations remain current, making it ideal for scenarios involving large data volumes or rapidly changing data.

Dynamic views provide the capability to link a base app to another app, allowing the incorporation of master visualizations from the latter into the former. This empowers app creators to leverage master visualizations as dynamic charts in different apps, without any imposed limit on the number of dynamic views within the base app.

Dynamic views consist of three primary components:

1. Dynamic Views: These are features integrated into base apps, connecting to template apps and allowing app creators to incorporate master visualizations from the template app into the base app.
2. Dynamic View Template Apps: These are Qlik Sense apps containing connections to data sources, including cloud databases, serving as the source for dynamic views.
3. Dynamic Charts: These are master visualizations within the dynamic view template app that can be added to base apps. Users have the flexibility to manually refresh these dynamic charts.

Dynamic views offer the flexibility to work with disparate data sets between the template app and the base app. For instance, if your base app focuses on customer purchases, you can add a dynamic view from a template app that contains weather data, allowing exploration of potential correlations.

To ensure the queried data from the template app aligns with the selections in your base app, binding expressions in the template app's script are utilized. This enables the dynamic view to selectively query a subset of data from the template app's data sources based on the choices made in the base app. As an example, you might bind the SalesDate field in the base app to the DailyTemperatureReadingDate field in the template app.

For more detailed guidance on creating and managing Dynamic Views, refer to the official help document.

Direct Query applications

Direct Query gives the user more options on how they want to access their data to suit their individual needs. Accessing the data through Direct Query allows the user to keep the data in its underlying data source. This increases the speed at which users can interact with their data in exchange for some of the flexibility that an in-memory Qlik Cloud app offers.

In general, it is recommended that you import your data into Qlik Cloud whenever possible. In-memory Qlik Cloud apps allow you to better customize your experience and get the most out of your data. However, if you find yourself in a situation where your goals cannot be met by importing data, Direct Query might be the solution for you. Due to their streamlined functionality, Direct Query apps can also help new users take their first step towards creating fully functional and fast in-memory apps.

You might consider using Direct Query when:

- Handling big data sources that would not fit in Qlik Engine memory
- Exploring new databases and tables.
- Near real-time data, for example, to see how many orders have come in during the last hour.
- Prototyping your dashboard in the initial phases before production.
- Extracting slices of data into the Qlik Sense engine through ODAG apps.

Please refer to the Supported Capabilities section of Qlik Help to be informed about which features are available when you are creating Direct Query apps.

Databricks Mosaic Machine Learning Integration

The Data Intelligence platform offers Machine Learning capabilities as a unified platform It enables you to build, train, and deploy machine learning models at scale using the most popular frameworks, such as Llama, TensorFlow, PyTorch, and sci-kit-learn or even using the AutoML capabilities to add classification or forecasting to your Qlik Analytics.

Integrating Qlik Cloud Analytics with Databricks Mosaic Machine Learning unlocks a powerful data analytics solution, combining the strengths of both platforms. Qlik Cloud Analytics facilitates seamless access and visualization of data from diverse sources, including Databricks tables and Delta Lake. Databricks Mosaic Machine Learning excels at handling intricate data processing and modeling tasks on extensive datasets, with results seamlessly delivered back to Qlik Sense for in-depth analysis and presentation.

This chapter will guide you through the integration process of Qlik Cloud Analytics and Mosaic AI Foundational Models in Databricks Mosaic AI using the Qlik-Databricks Mosaic ML connector. We will also showcase practical use cases and highlight the benefits of this integration across various industries and scenarios.

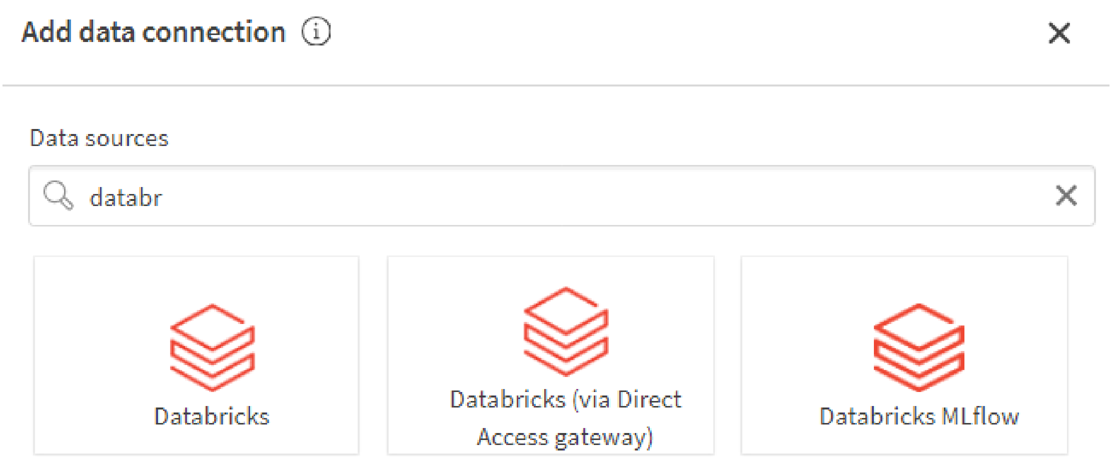
Deploying a Foundational Model in Databricks Mosaic AI

When data scientists deploy Databricks ML models in Databricks Mosaic AI, they make decisions based on their specific needs. For use cases requiring numerous predictions with no stringent latency constraints, the common approach involves batch inference. This entails providing the model with substantial data, generating predictions, and recording them in a table on the Lakehouse. Qlik Analytics can read the results from tables populated by the model. For scenarios demanding low-latency predictions, like responding to user actions in an application or serving analytics use cases deploying ML models as REST endpoints is recommended. This allows applications to send requests to a continuously available endpoint and receive predictions instantly.

Databricks streamlines this process with a turnkey service called Mosaic AI Model Serving. Mosaic AI Model Serving is a unified service for deploying, governing, querying and monitoring models fine-tuned or pre-deployed by Databricks like Llama 2, MosaicML MPT or BGE, or from any other model provider like Azure OpenAI, AWS Bedrock, AWS SageMaker and Anthropic.

Databricks ML Connector

Qlik Cloud Analytics provides three different connections to the Databricks Mosaic AI



- Databricks – used to extract data from the Lakehouse
- Databricks (via Direct Access gateway) – used to extract data from the Lakehouse using a gateway to implement higher security level (firewalls, WAF, etc)
- Databricks MLflow – scope of this chapter, used to interact with Mosaic models

Connections marked as MLFlow Analytic connections can transmit data to machine learning endpoints that are externally accessible. These endpoints perform calculations on the data, and the results are then returned and incorporated into a Qlik Sense application.

There are two primary methods for integrating analytic connections into a Qlik Sense application: data load scripts and chart expressions.

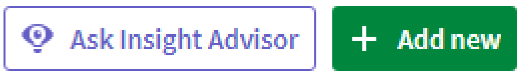
- Data load scripts send data when the application is being reloaded and usually involve a bigger dataset (like sales predictions, churn for a customer list, etc)
- Chart expressions send data in real-time based on the user´s selection showing the results based on that selection.

This process allows Qlik Analytics to leverage the power of external machine learning models, enhancing its analytic capabilities and providing users with more robust and comprehensive insights.

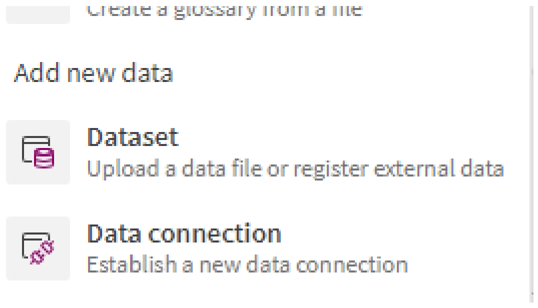
Creating an MLFlow connection

Please refer to the latest documentation from help.qlik.com for any updates. Follow these steps to create a Databricks Mosaic ML connection:

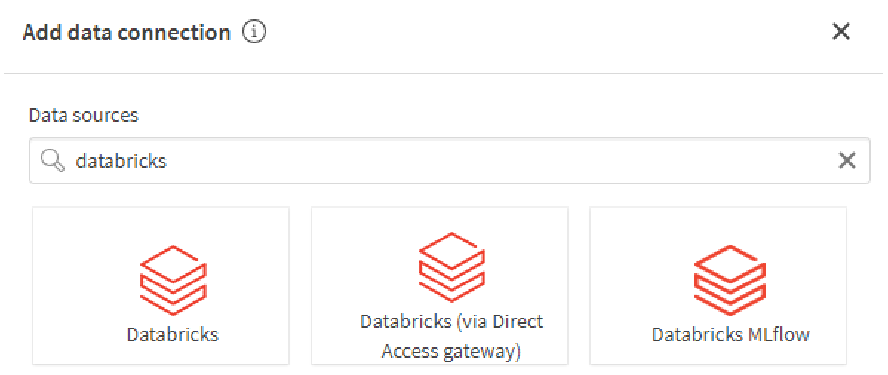
1. Go to your Qlik Cloud Analytics hub and press the button + Add New



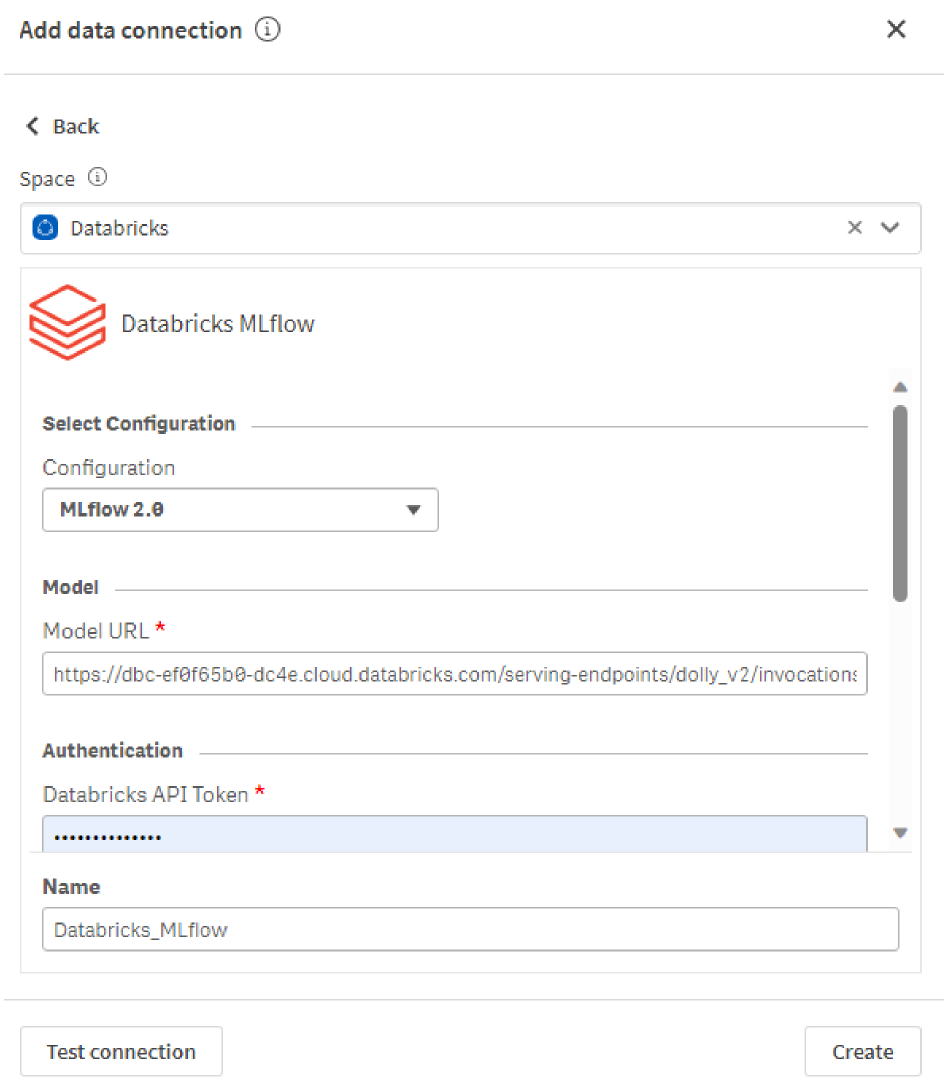
2. Press the option Data Connection


Create a glossary from a file
Add new data
Dataset
Upload a data file or register external data
Data connection
Establish a new data connection

3. In the Data Sources box, search for *Databricks* and select the option Databricks MLflow



4. Fill the Model URL field with the serving point URL (this information is available on the Databricks console)



Space: Databricks

Databricks MLflow

Select Configuration

Configuration: MLflow 2.0

Model

Model URL *: https://dbc-ef0f65b0-dc4e.cloud.databricks.com/serving-endpoints/dolly_v2/invocations

Authentication

Databricks API Token *: ....

Name: Databricks_MLflow

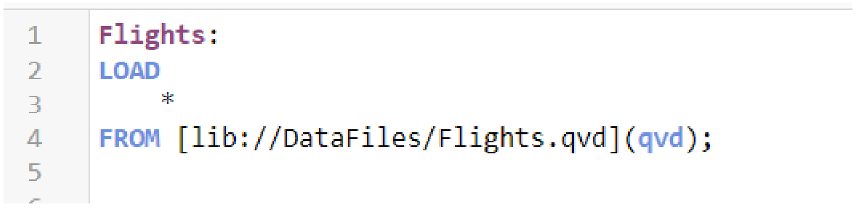
5. Fill the Databricks API Token with your Databricks token (refer to Obtaining Databricks credentials and connection details to learn how to get one)

6. Test and save your connection.

Using an MLFlow connection at the script level

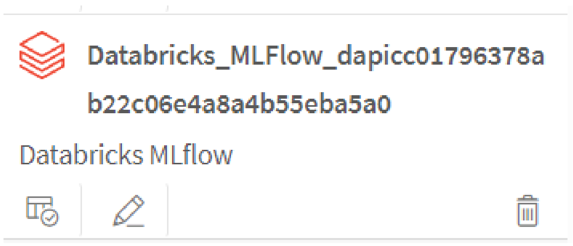
As we discussed in the Databricks ML Connector section, it is possible to interact with the MLFlow connection at reload time to send bulks of data to the Databricks serving point and use the results as a field in our Qlik Sense app data model. To send data to the endpoint the data needs to be previously loaded in your Qlik Sense app.

1. Please import data into your Qlik Sense application. In this instance, we're utilizing a flight cancellation forecasting model to anticipate whether a flight will undergo cancellation.



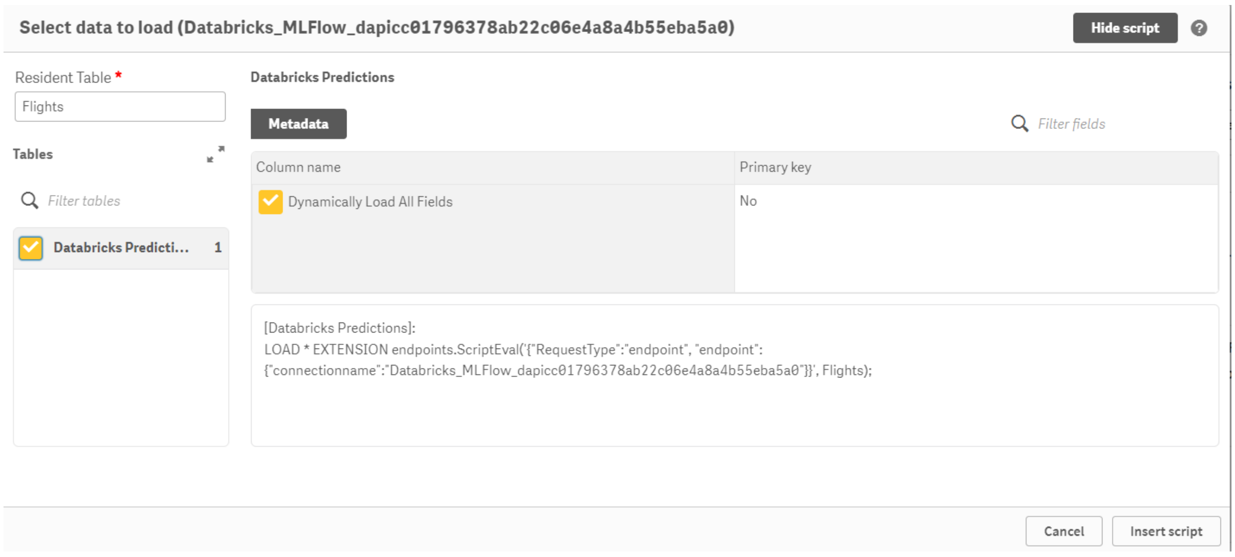
```
Flights:
LOAD
    *
FROM [lib://DataFiles/Flights.qvd](qvd);

```

2. Let´s use the ML_Flow wizard to create your script



Databricks_MLFlow_dapicc01796378a
b22c06e4a8a4b55eba5a0
Databricks MLflow

3. Fill the Resident Table with your table name (from step1) and then press Insert Script


Select data to load (Databricks_MLFlow_dapicc01796378ab22c06e4a8a4b55eba5a0)

Resident Table *

Flights

Tables

Databricks Predicti... 1

Databricks Predictions

Column name	Primary key
✓ Dynamically Load All Fields	No

```
[Databricks Predictions]:
LOAD * EXTENSION endpoints.ScriptEval('{"RequestType":"endpoint", "endpoint":
{"connectionname":"Databricks_MLFlow_dapicc01796378ab22c06e4a8a4b55eba5a0"}}', Flights);
```

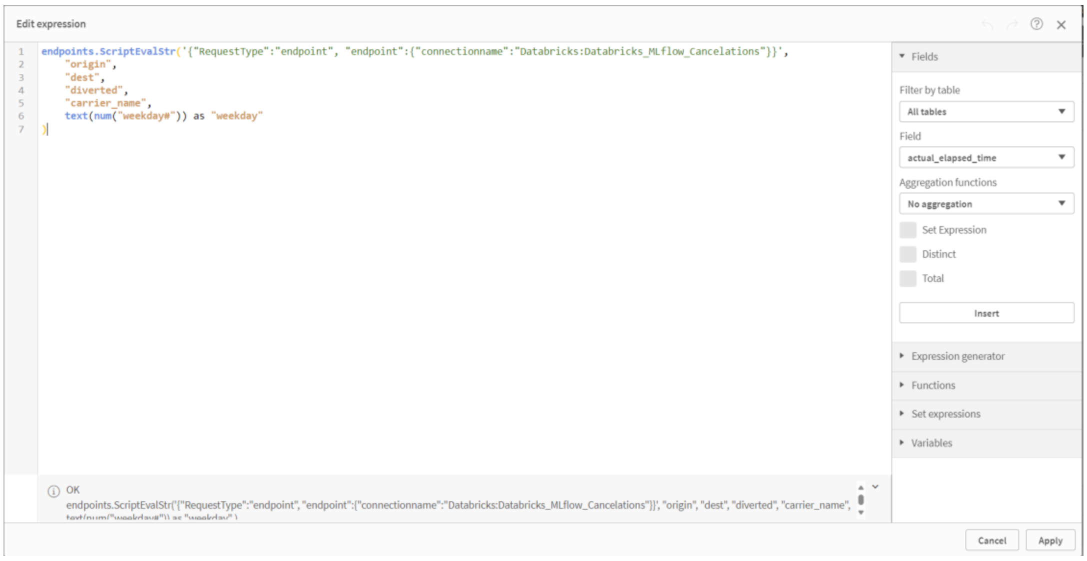
4. Reload your application and check if a new table was added to your data mode with the results from the Databricks Model

5. You can use the predictions in your application as a regular field

Using an MLFlow connection at the chart level

As previously discussed in the Databricks Mosaic ML Connector section, it is feasible to engage with the MLFlow connection in real-time by transmitting segmented data (filtered data based on user selection) to the Databricks serving point and exhibiting the outcomes within any Qlik Sense object (inclusive of 3rd party extensions). A specific syntax exists to establish this connection. Qlik Sense offers distinct functions, and the choice among them hinges on the data types being transmitted to the connection, as well as whether you prefer Qlik Engine to aggregate the data before transmitting it to the connection or to make a call for each dimension value. A more comprehensive discourse on these functions is accessible in the Server Side Extensions Syntax document.

As an example, below we are connecting to the same endpoint to get one cancellation predictions passing the parameters the model needs do perform the forecast



```
endpoints.ScriptEvalStr('{"RequestType":"endpoint", "endpoint":{"connectionname":"Databricks:Databricks_MLflow_Cancelations"}}',
    "origin",
    "dest",
    "diverted",
    "carrier_name",
    text(num("weekday#")) as "weekday"
)
```

```
OK
endpoints.ScriptEvalStr('{"RequestType":"endpoint", "endpoint":{"connectionname":"Databricks:Databricks_MLflow_Cancelations"}}', "origin", "dest", "diverted", "carrier_name",
text(num("weekday#")) as "weekday" )
```

Additional Resources

Please refer to the list below for a more detailed discussion of the specific items addressed in this document.

- Loading Databricks data - https://help.qlik.com/en-US/cloud-services/Subsystems/ODBC_Connector_help/Content/Connectors_ODBC/Databricks/Databricks-connector.htm
- Create a Databricks connection - https://help.qlik.com/en-US/cloud-services/Subsystems/ODBC_Connector_help/Content/Connectors_ODBC/Databricks/Create-Databricks-connection.htm
- Creating a Databricks MLflow connection - https://help.qlik.com/en-US/cloud-services/Subsystems/Hub/Content/Sense_Hub/LoadData/ac-databricks-mlflow-create.htm
- Databricks MLflow analytics source - https://help.qlik.com/en-US/cloud-services/Subsystems/Hub/Content/Sense_Hub/LoadData/ac-databricks-mlflow-overview.htm
- What are SQL Warehouses? - https://docs.databricks.com/en/compute/sql-warehouse.html
- Mosaic AI - https://www.databricks.com/product/machine-learning