Databricks Certified Data Analyst Associate Video Course

Jun Shan

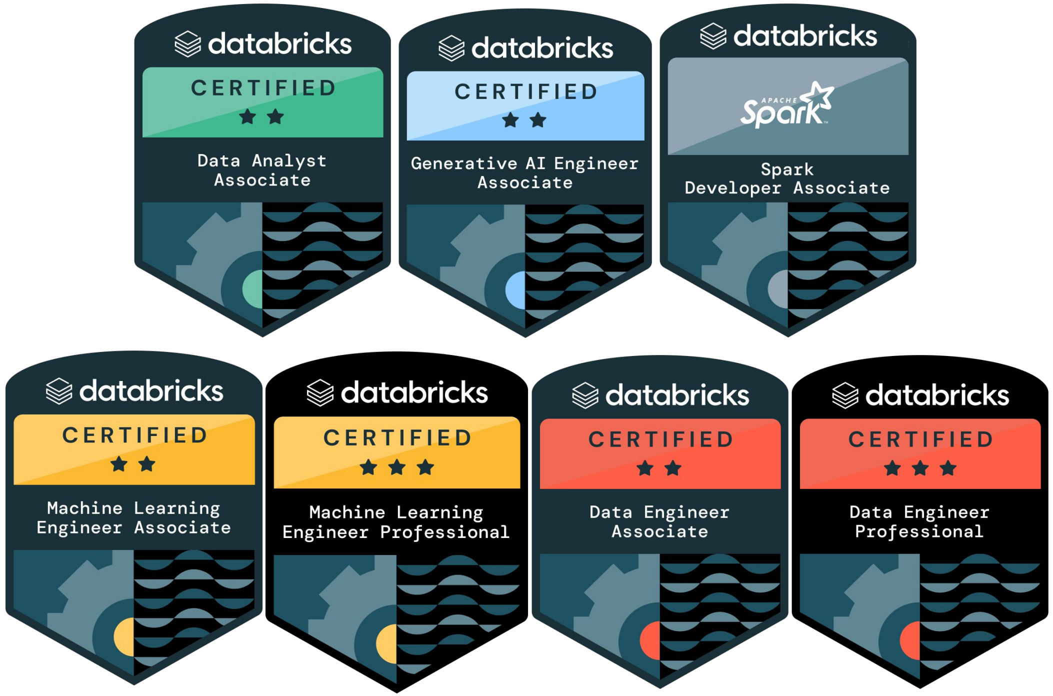
Module 0.1:
Introduction to Course

About This Course

Prepares you for the “Databricks Certified Data Analyst Associate” certification exam

- Databricks:
  a. Data analytics company based on Apache Spark
  b. Cloud-based platform for data and analytics
- The usage is getting momentum in recent years
  a. Leader in the 2023 Gartner Magic Quadrant™ for Cloud Database Management Systems
  b. Partnership with major cloud vendors
  c. More than 50% YOY growth for year 2023

About Databricks Certifications


databricks CERTIFIED Data Analyst Associate
databricks CERTIFIED Generative AI Engineer Associate
databricks APACHE Spark Spark Developer Associate
databricks CERTIFIED Machine Learning Engineer Associate
databricks CERTIFIED Machine Learning Engineer Professional
databricks CERTIFIED Data Engineer Associate
databricks CERTIFIED Data Engineer Professional

About This Certification Exam

- The Databricks Certified Data Analyst Associate certification
  - evaluate your Databricks data analytics skill set
  - boost your credibility as a data analyst
  - enhance your career prospects

Target Audience

- Entry level SQL Analyst, Data Analysts, and Business Analysts responsible for analyzing data and providing actionable insights
- Entry level Professionals who want to leverage Databricks SQL for creating dashboards and running queries
- Entry level Data engineers interested in acquiring analytical skills to complement their expertise
- Students or professionals new to data analytics who want to showcase their proficiency with Databricks tools

Course Content

- Help you learn/enhance the topics covered by the certification exam
    1. Databricks SQL
    2. Data Management
    3. SQL in the Lakehouse
    4. Data Visualization and Dashboarding
    5. Analytics applications
- Familiarize you with the terms and contexts of the certification exam questions

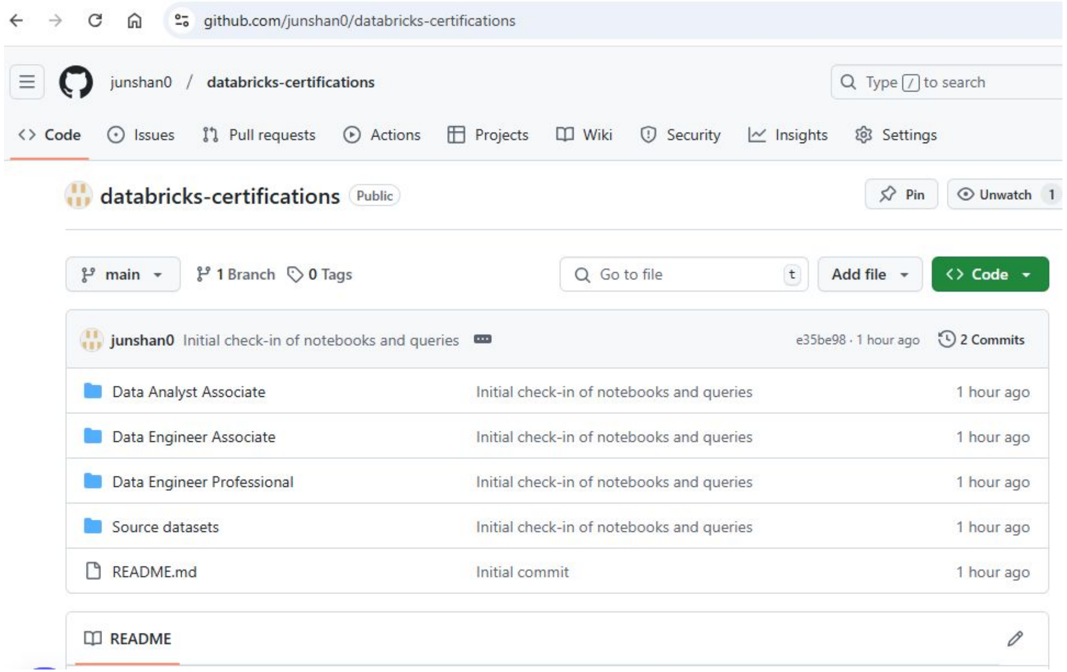
About the Instructor

- Cloud data architecture advisor with more than 20 years of professional experience.
- Teaches data analytics at several universities in New York area.
- Databricks Certified Data Engineer Professional and Certified Data Analyst Associate myself.
- Certified Professional Solution Architect in AWS, Azure, and GCP. Certified Professional Data Engineer in AWS, Azure, and GCP
- Author of *SQL for Data Analytics: Harness the power of SQL to extract insights from data, Third Edition*.

Module 0.2:
Related Resources

Github Repository for this course

Github Repository


github.com/junshan0/databricks-certifications

junshan0 / databricks-certifications

databricks-certifications Public

1 Branch 0 Tags

junshan0 Initial check-in of notebooks and queries e35be98 · 1 hour ago 2 Commits

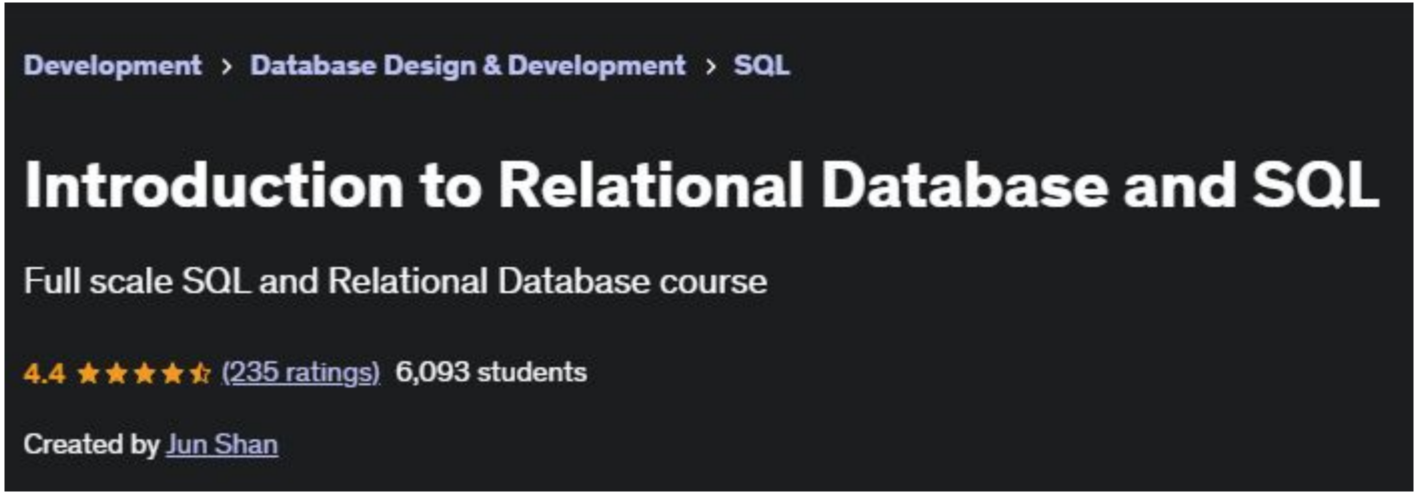
Data Analyst Associate	Initial check-in of notebooks and queries	1 hour ago
Data Engineer Associate	Initial check-in of notebooks and queries	1 hour ago
Data Engineer Professional	Initial check-in of notebooks and queries	1 hour ago
Source datasets	Initial check-in of notebooks and queries	1 hour ago
README.md	Initial commit	1 hour ago

README

Prerequisites

6+ months of hands-on experience performing the data analysis tasks outlined in the exam guide

[Introduction to Relational Database and SQL]



Development › Database Design & Development › SQL

Introduction to Relational Database and SQL

Full scale SQL and Relational Database course

4.4 ★★★★☆ (235 ratings) 6,093 students

Created by Jun Shan

Other Databricks Courses from Me

Databricks Data Engineer Associate Certification Course, O'Reilly

Databricks Data Engineer Associate Certification Course: Data Engineering and Data Science on a Unified Platform

by Jun Shan
Released December 2024
Publisher(s): O'Reilly Media, Inc.
ISBN: 0642572061357

Module 0.3:
Databricks Setup

Databricks Environment

Databricks is a cloud based data analytics platform

1. Databricks account
2. SaaS platform
   - Cloud account (classic compute setup)
   - Databricks serverless compute (starting 2024)
3. (Optionally) other cloud storage accounts

Setting up Databricks Environment

1. Databricks interface
2. We will use Azure Databricks in this course
    a. Azure account setup
    b. Databricks workspace setup
    c. Access to Databricks workspace UI

Module 1: Databricks SQL

Learning Objectives

Module 1: Databricks SQL

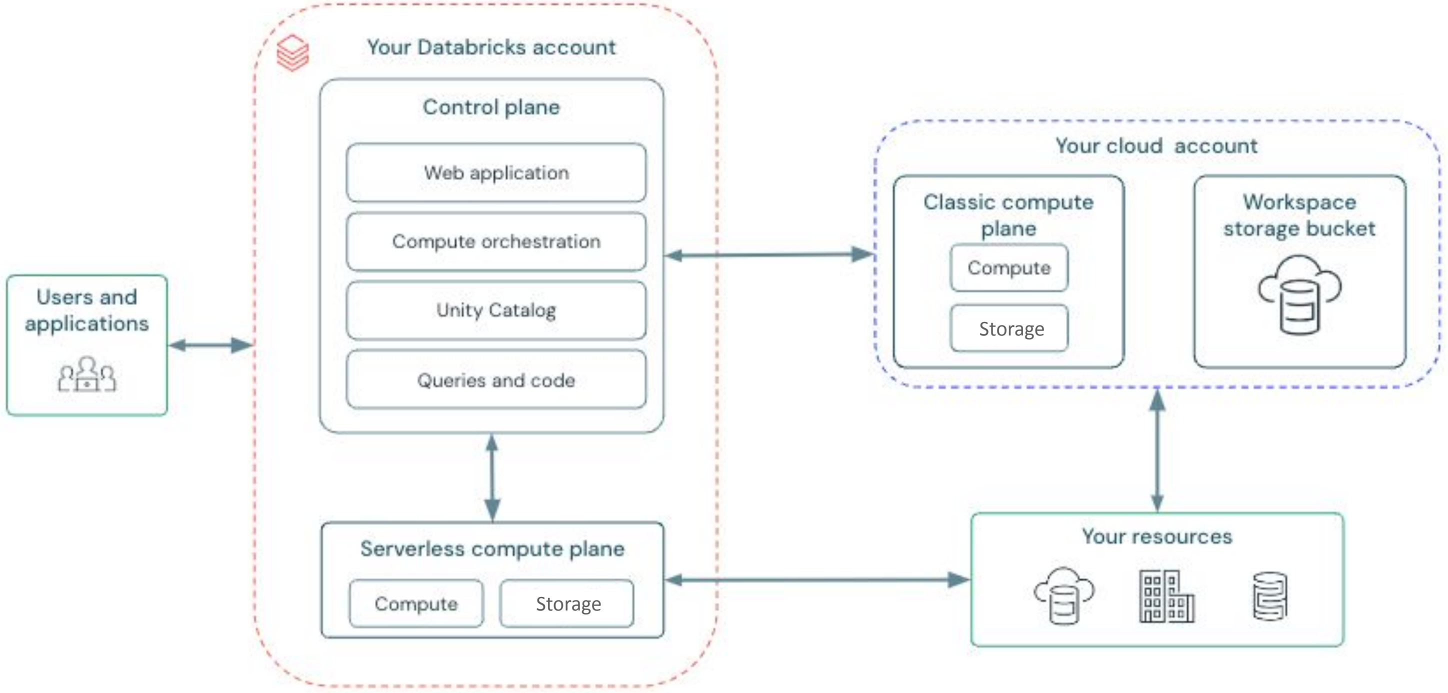
By the end of this module, you will understand the diverse audiences of Databricks SQL, including data analysts, engineers, and stakeholders. You will learn the user-friendly interface of Databricks SQL, which supports SQL queries, schema browsing, and dashboards. Key benefits of the Databricks SQL that you learn here include a scalable medallion architecture for clean data pipelines, support for batch and streaming data, and integrations with tools like Tableau, Power BI, and Fivetran to enhance BI workflows and data ingestion processes.

Module 1.1: Databricks SQL Introduction

Audience of Databricks SQL

- Key Audiences for Databricks SQL: SQL analysts, data analysts, and business analysts to analyze data, create reports, and build dashboards
- Side Audiences for Databricks SQL:
  - Data engineers to validate pipelines, ensure data quality
  - Data scientists for exploratory data analysis or to validate hypotheses
  - Business Users and Decision-Makers for viewing dashboards and reports

Databricks Architecture (2024)

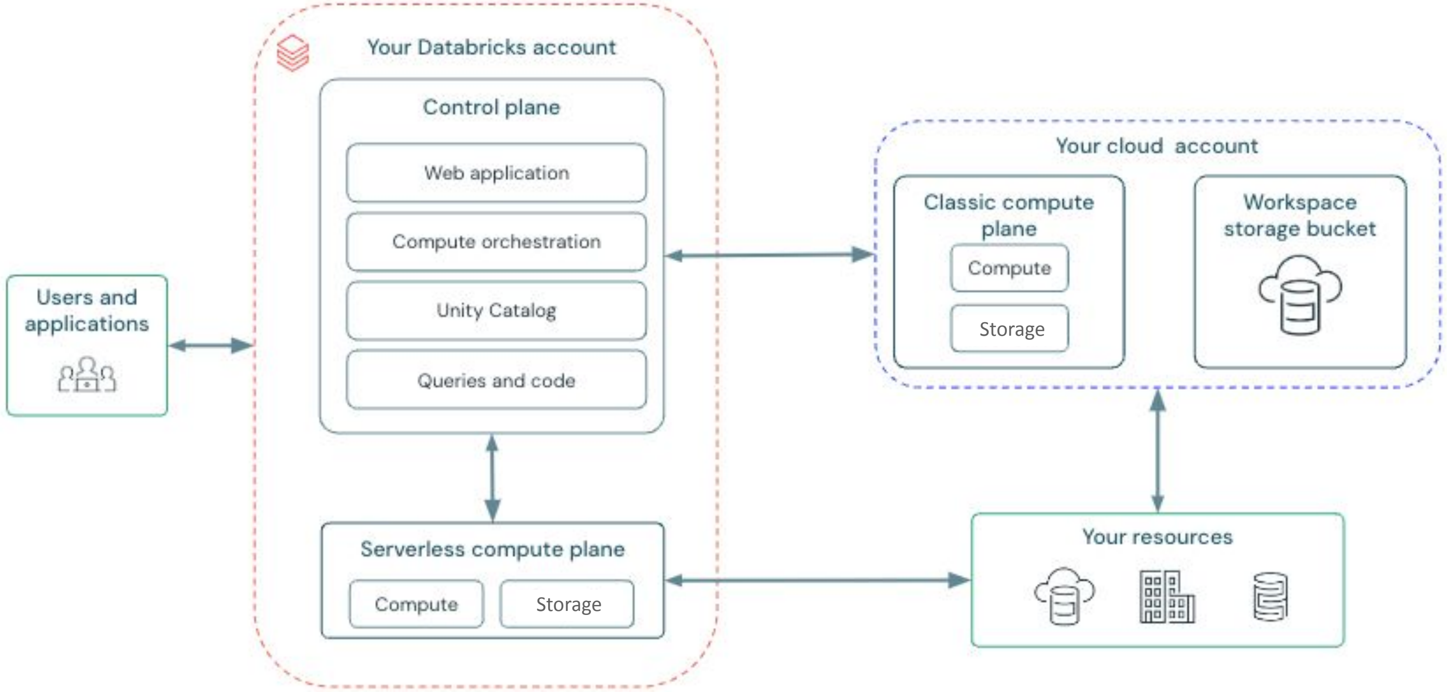


Benefits of Databricks SQL

- Centralized Analytics Platform
  - Integrated data access across platforms
  - Rich Data Transformation Capabilities
  - Seamless Integration with BI Tools
- Open formats and standard ANSI SQL
  - Delta Lake Integration
  - Optimized for Big Data
  - Seamless Access to Structured and Unstructured Data
- Data Governance and Security
  - Unity Catalog integration
  - Access Control
  - Audit and Monitoring

Module 1.2:
Databricks SQL Components

Databricks Architecture (2024)



Databricks SQL Warehouse

SQL Warehouse: compute resource that lets you query and explore data on Databricks

Demo:

- Overview of Databricks SQL user interface
- Defining SQL Warehouse: Cost vs Capacity
- Serverless Databricks SQL Warehouse
    - Instant and elastic compute
    - Minimal management overhead
    - Lower total cost of ownership (TCO)

Databricks SQL User Interface

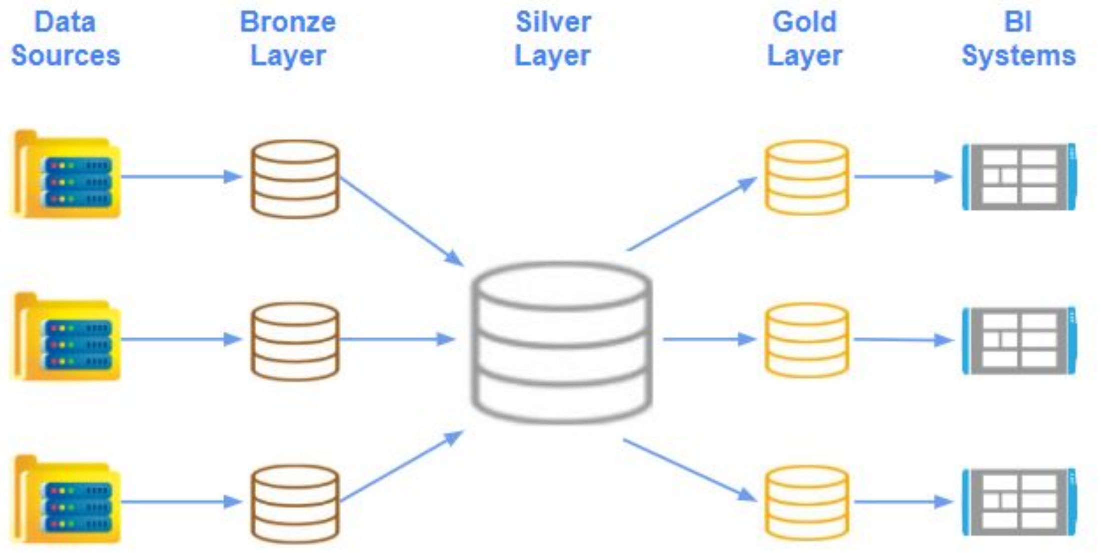
Demo:

- Use SQL editor to write and run SQL code
- Save queries
- Show Dashboard interface (will discuss in a dedicated section)

Module 1.3:
Data Loading

Medallion Architecture

- Design pattern to organize data in a lakehouse
- Arrange data into three layers based on their lifecycle stage and quality:
    - Raw data in into bronze layer
    - Transformed data in silver and gold layers



Data Import for Databricks SQL

Demo of data creation

- Upload small text files via UI
- Import from object storage using Databricks SQL
    - Databricks SQL can ingest directories of files that are of the same type

Streaming Data

As new rows coming in, they are continuously added (appended) to the unbounded table

- readStream/writeStream: continuous operation
  - vs Batch mode (read/write): only triggered when called
- Streaming is expensive
  - Use batch processing if cost-sensitive: Once per day, once per hour, etc.
- Auto Loader and Delta Live Stream Table (DLT)

```sql
CREATE OR REFRESH STREAMING TABLE stream_table AS
SELECT * FROM read_files("<storage location>", "csv");
```

Module 1.4:
Partner Connect

Partner Connect

Complementary tool for BI partner tool workflows

- Create trial accounts with select Databricks technology partners and connect your Databricks workspace to partner solutions from the Databricks UI
- Provides a simpler alternative to manual partner connections by provisioning the required Databricks resources on your behalf, then passing resource details to the partner.

Demo:

- Fivetran
- Power BI

Module 2: Data Management

Learning Objectives

Module 2: Data Management

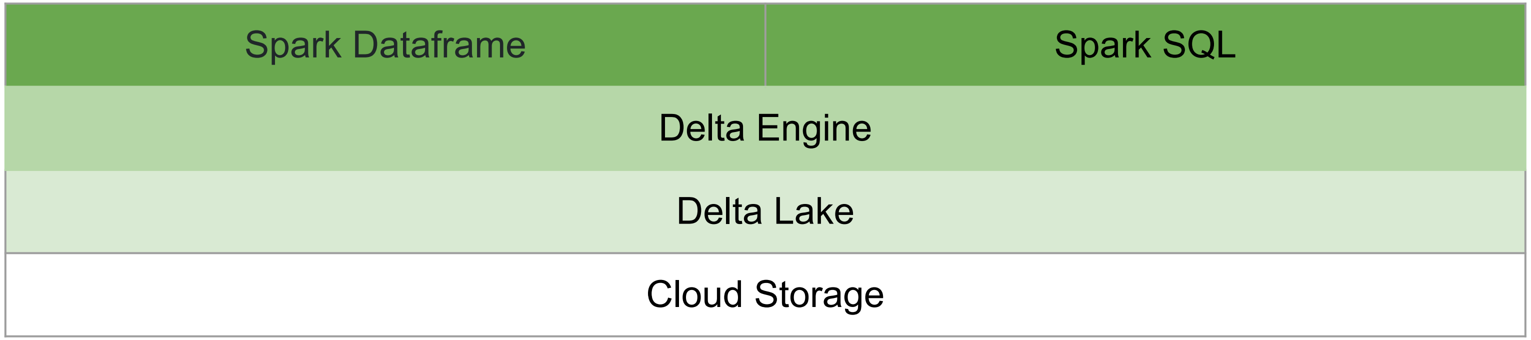
By the end of this module, you will learn the Delta Lake within the Databricks Lakehouse, which provides data file management, table metadata handling, and version history. You will learn how to create, modify, and manage both managed and unmanaged tables. You will use tools like Data Explorer to enable data exploration, previewing, security management, and access control.

Module 2.1:
Delta Lake

Databricks Lakehouse

Delta Lake:

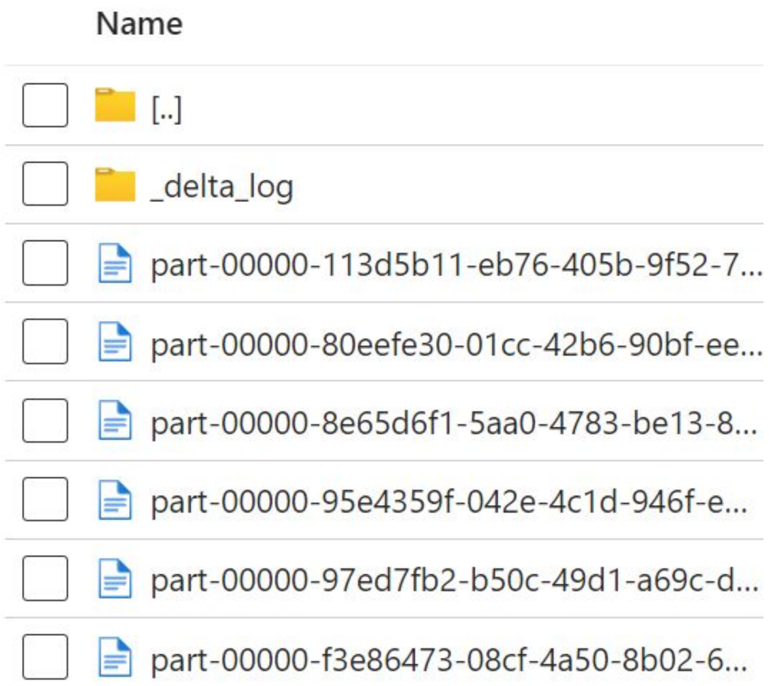
- Data management and governance layer on top of the cloud data platforms
- Open source, open format. Enables a large variety of functionalities
- Storage unit: Delta table, a structured storage



Delta Table Creation

Delta table: structured storage

- Parquet file: Apache Parquet is an open source data file format designed for efficient data storage and retrieval
    - Column-oriented
    - Compressed
    - Has its own schema definition
    - Only format used by Delta tables
- Can have subfolders for partition
- _delta_log folder: Delta Lake transaction log (also known as the DeltaLog)



Name
[..]
_delta_log
part-00000-113d5b11-eb76-405b-9f52-7...
part-00000-80eefe30-01cc-42b6-90bf-ee...
part-00000-8e65d6f1-5aa0-4783-be13-8...
part-00000-95e4359f-042e-4c1d-946f-e...
part-00000-97ed7fb2-b50c-49d1-a69c-d...
part-00000-f3e86473-08cf-4a50-8b02-6...

ACID Guarantee

Delta Lake’s ACID Guarantee

- **A**tomicity: Complete all operations or none
- **C**onsistency: Ensure data validity
- **I**solation: Prevent interference in concurrent transactions
- **D**urability: Persist data changes despite failures

Benefits of ACID:

- Ensure reliability and integrity
- Prevent inconsistent state and partially completion

Module 2.2: Table Storage

External Location

Storage path within a cloud storage platform but outside of Databricks associated cloud path, with a storage credential that authorizes access to that path

- Catalog
    - Catalog can be defined on top of external location
    - Table/volume defined in external location is still managed table/volume, if not specified as external table/volume.
- Table
    - Managed table
    - External table: defined with LOCATION keyword
- Volume: Volume of storage of non-tabular datasets
    - Managed volume
    - External volume: defined with LOCATION keyword

Storage and Table Definitions

Data Format	Delta table			Other formats
Table Type	Managed table		External Table	
Delta Lake	Databricks Default Delta Lake	Databricks User Managed Delta Lake	External to Delta Lake	
Storage	Databricks Managed Storage	External Cloud Storage (External Location)		

Table Storage Demo

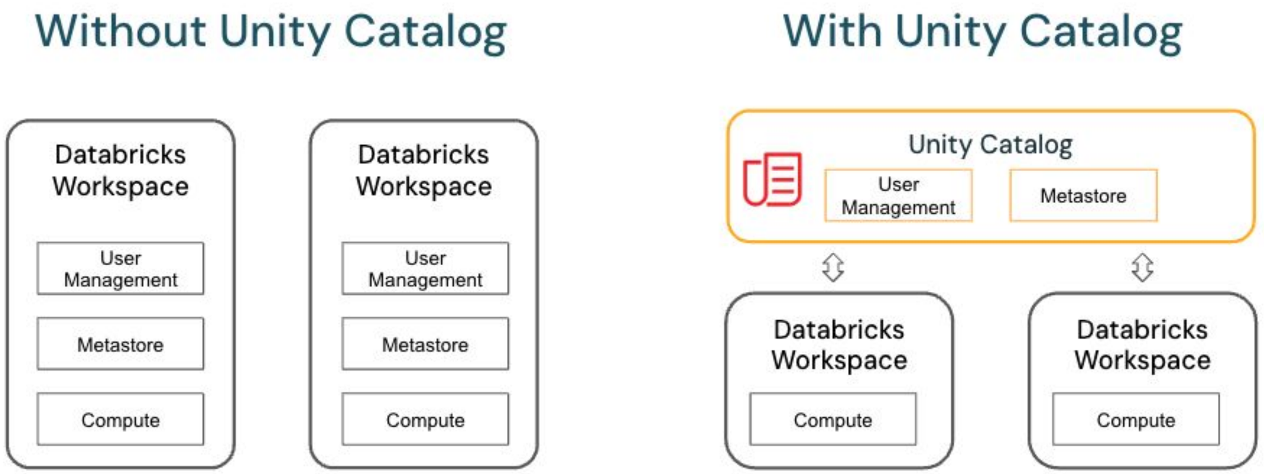
Demo

- Understand external location and storage credential
- Create and use catalog
- Create managed tables
- Create external tables
- Check whether a table is managed or unmanaged through catalog browser
- Rename table
- Drop managed table: Data folder/file will be removed
- Drop external table: Data folder/file will remain untouched

Module 2.3: Catalog Explorer

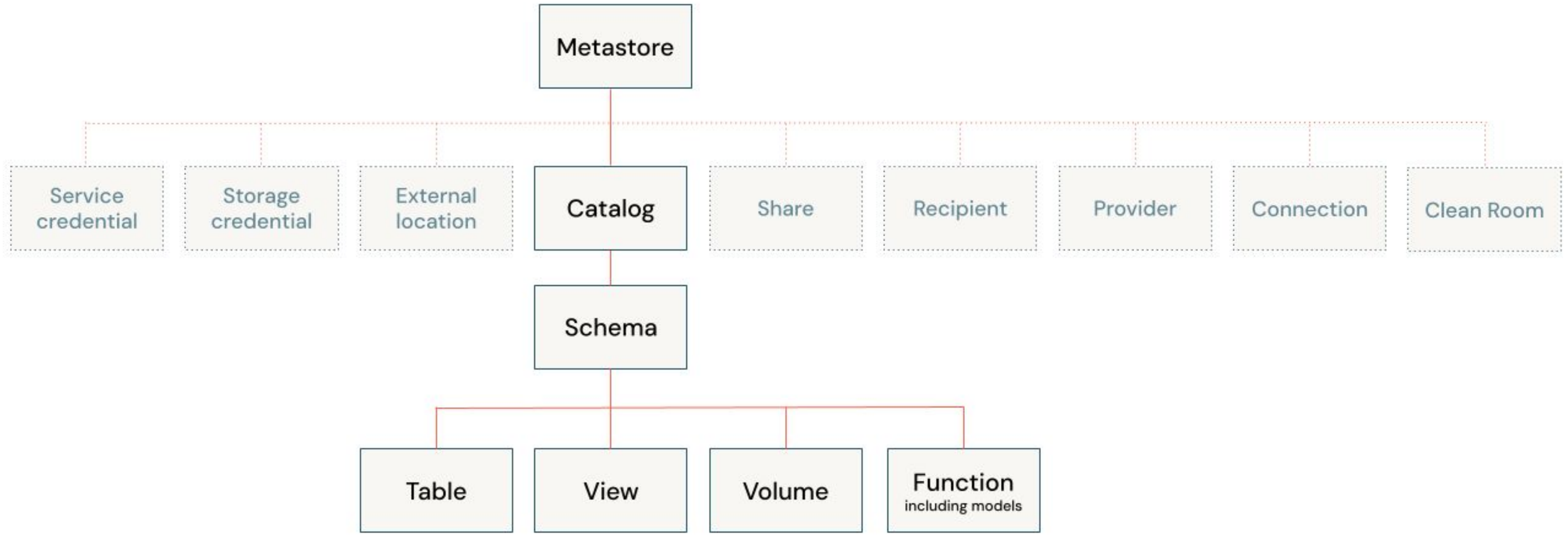
Unity Catalog

Centralized, cross-workspace control over access control, auditing, lineage, and data discovery



Unity Catalog Object Hierarchy

Catalog.Schema.Table



Data Security

- The responsibilities of a table owner.
- Grant access via GRANT statement
- Considerations of Personally Identifiable Information (PII) data
  - Row level security
  - Column masking

Table, View, and Temp View

	Permanent Existence	Physical Storage	Usage
Table	✓	✓	Fundamental unit of data storage Physical existence of data
View	✓	✗	Data pulled as real time query Can be used for access management or data masking
Temporary View	✗	✗	Exists for the lifespan of a user session. Can be used in job pipelines such as ETL process

Catalog Explorer

- Was called Data Explorer before 2024
    - Exam guide has not got updated

Demo

- Explore, preview, and secure data using Catalog Explorer.
- Identify the table owner using Catalog Explorer.
- Change access rights to a table using Catalog Explorer.

Module 3:
SQL in the Lakehouse

Learning Objectives

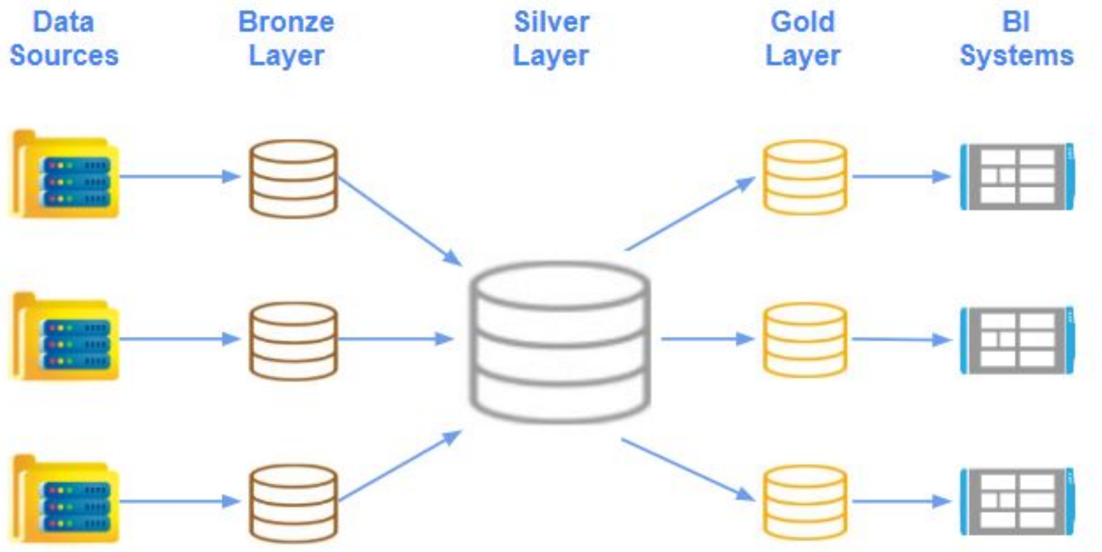
Module 3: SQL in the Lakehouse

By the end of this module, you will learn the use of ANSI SQL in Databricks, enabling efficient data retrieval, aggregation, and transformation with techniques like subqueries, joins, and windowing. You will learn to work with nested data, clean silver-level data, and apply advanced aggregations using roll-up, cube, and Spark SQL functions for optimized performance.

Module 3.1:
SELECT Statement

Medallion Architecture

- Design pattern to organize data in a lakehouse
- Arrange data into three layers based on their lifecycle stage and quality:
  - Raw data in into bronze layer
  - Transformed data in silver and gold layers



SELECT Statement

Databricks uses ANSI SQL as the standard in the Lakehouse.

- The output of SELECT/SELECT DISTINCT
- WHERE clause
- Functions

COUNT

Different ways of counting:

- `COUNT(*)`: Count all rows in the table
- `COUNT(<expression>)`: Count when the `<expression>` is not NULL (duplicates are also counted)
- `COUNT(DISTINCT <expression>)`: Count the unique occurrence of `<expression>`
- `COUNT_IF(<condition>)`: Count when the `<condition>` is true

```
SELECT COUNT_IF(Name LIKE ‘T%’) FROM myTable
```

Timestamp

`CAST(<Value> as <Data_Type>)` function for data type conversion

- Cast string to timestamp
- Cast timestamp to string

`EXTRACT(<field> FROM <Timestamp_Value>)` function to extract calendar data from a timestamp

- YEAR, Y, MONTH, MON, WEEK, W, DAY, D
- HOUR, H, MINUTE, M, SECOND, S

String Manipulation

Data type conversion

- `CAST(<Value> as <Data_Type>)`: data type conversion
- `STRING(<Value>)`: a shortcut to convert values into string

Other string functions

- `concat(expr1, expr2[, …])`
- `contains(expr, subExpr)/instr(expr, subExpr)`
- `left(str, len)/right(str, len)`
- `len(expr)/length(expr)`
- `substr(expr, pos[, len])`

UDF

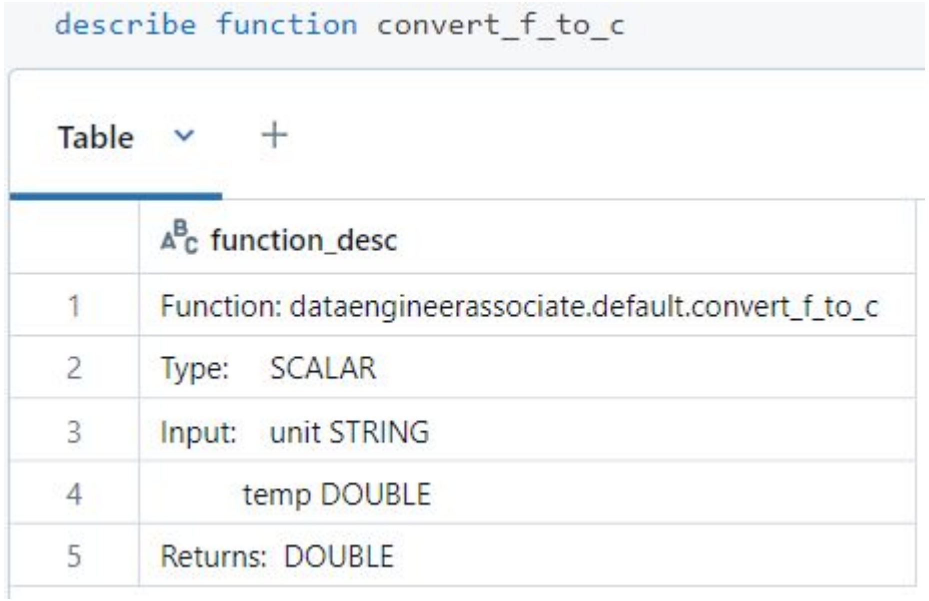
Use UDFs to perform specific tasks, such as complex calculations, transformations, or custom data manipulations

```
CREATE FUNCTION convert_f_to_c(unit STRING, temp DOUBLE)
RETURNS DOUBLE
RETURN
  CASE
    WHEN unit = "F" THEN (temp - 32) * (5/9)
    ELSE temp
  END;
```

Function Metadata

Returns the basic metadata information of an existing function:

```
DESCRIBE FUNCTION
<function_name>;
```



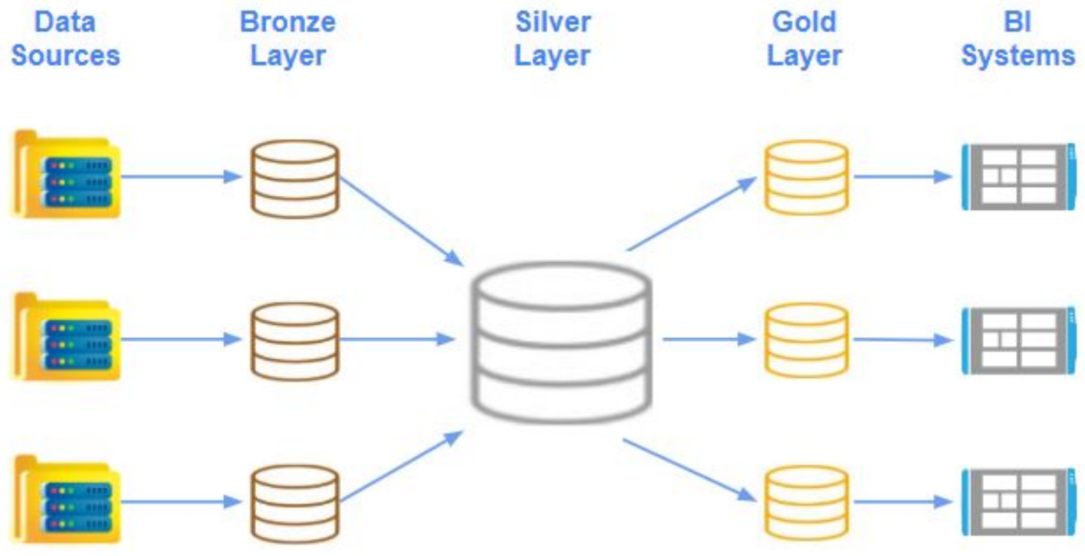
```
describe function convert_f_to_c
```

	function_desc
1	Function: dataengineerassociate.default.convert_f_to_c
2	Type: SCALAR
3	Input: unit STRING
4	temp DOUBLE
5	Returns: DOUBLE

Module 3.2:
Dataset Operations

Medallion Architecture

- Design pattern to organize data in a lakehouse
- Arrange data into three layers based on their lifecycle stage and quality:
  - Raw data in into bronze layer
  - Transformed data in silver and gold layers



CTE and Subqueries

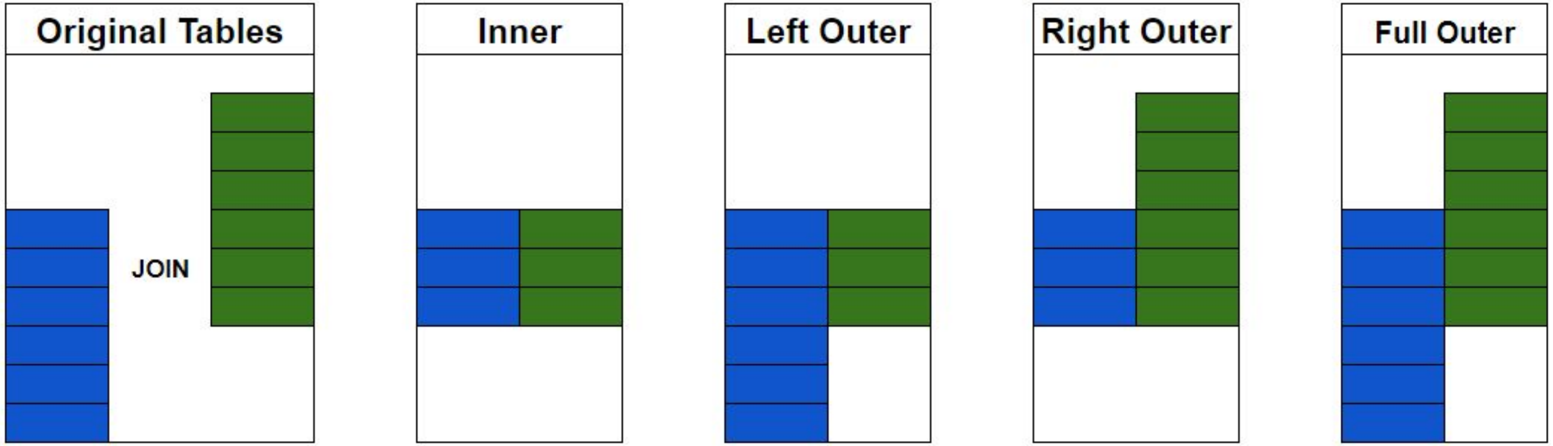
```
common_table_expression
      { subquery | set_operator }

subquery
      { SELECT clause |
        VALUES clause |
        ( query ) |
        TABLE [ table_name | view_name ]}
```

JOIN Types

Databricks uses standard SQL syntax for outer joins

```
FROM table1 INNER JOIN table2 ON table1.join_key = table2.join_key
FROM table1 LEFT  JOIN table2 ON table1.join_key = table2.join_key
FROM table1 RIGHT JOIN table2 ON table1.join_key = table2.join_key
FROM table1 FULL  JOIN table2 ON table1.join_key = table2.join_key
FROM table1 CROSS JOIN department
```



SEMI JOIN and ANTI JOIN

- `[ LEFT ] SEMI`: Returns values from the left table that has a match with the right. It is also referred to as a left semi join.
    - Difference with LEFT OUTER JOiN/INNER JOIN: SEMI JOIN only returns matched results which is similar to INNER JOIN, but only return result from the left table, not the right table
- `[ LEFT ] ANTI`: Returns the values from the left table that have no match with the right table reference. It is also referred to as a left anti join.

GROUP BY with Cube/Rollup

- `GROUP BY group expression`
  `GROUP BY Column1, Column2`

- `GROUP BY { group_expression | { ROLLUP | CUBE | GROUPING SETS } ( grouping_set [, ...] ) } [, ...]`
  - `grouping_set`: Groups the rows for each grouping set specified after GROUPING SETS
  - `GROUP BY group_expression [, ...] [ WITH ROLLUP | WITH CUBE ]`: Groups the rows for each grouping set rolled up with `ROLLUP/CUBE`
  - `GROUP BY ALL`

Columns in the SELECT clause must either be in the GROUP BY clause or be an aggregation.

Window Functions

Window function: Identify the characteristics of each row within its dataset/window

- Scalar function: Identify the characteristics of each row by itself
- Aggregation function: Identify the characteristics of each dataset

```
function OVER { window_name | ( window_name ) | window_spec }
    function: { ranking function | analytic function | aggregate function }
    over_clause: OVER { window_name | ( window_name ) | window_spec }
    window_spec: ( [ PARTITION BY partition [ , ... ] ] [ order_by ] [
    window_frame ] )
```

Module 3.3:
Semi-Structured Data

Types of Data

- Structured: Highly organized and formatted, typically fits into rows and columns, following a predefined schema or data model
  Examples: Relational databases — usually data that has been processed
- Semi-structured: Do not have rigidly defined structure as traditional relational data but still retains some form of structure
  Examples: JSON data, array
- Unstructured: No clear organization or format
  Examples: Free text documents (e.g., Word files, PDFs), social media posts, multimedia content (e.g., image, audio, video)

JSON and Array

JSON and array processing can get really complex. Generally, use Python for them. However, Databricks also provides some SQL functions for JSON/Array processing.

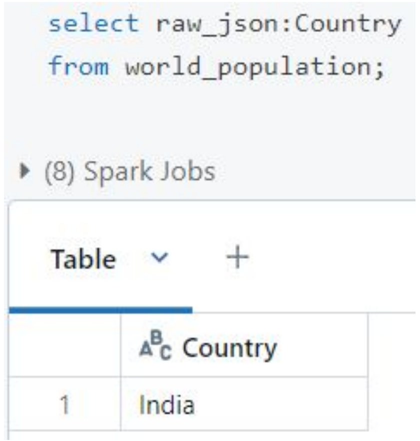
- JSON processing
  - Name-value pair
  - Going deep into nested structure
- Array processing
  - Organizing elements

JSON and Array

Dump JSON into a dataset (table) with only one column. The JSON content is in this column. Use `column:attribute` to access each individual attribute

```
CREATE TABLE world_population AS
select '{"Country":"India","Population":"1428627663"}' as
raw_json;

select raw_json:Country
from world_population;
```



```
select raw_json:Country
from world_population;
```

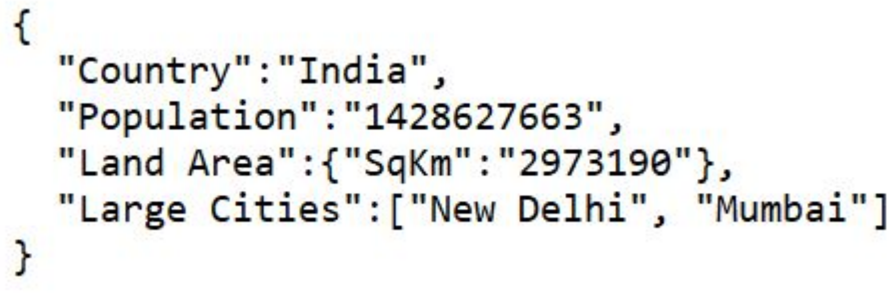
▸ (8) Spark Jobs

	Country
1	India

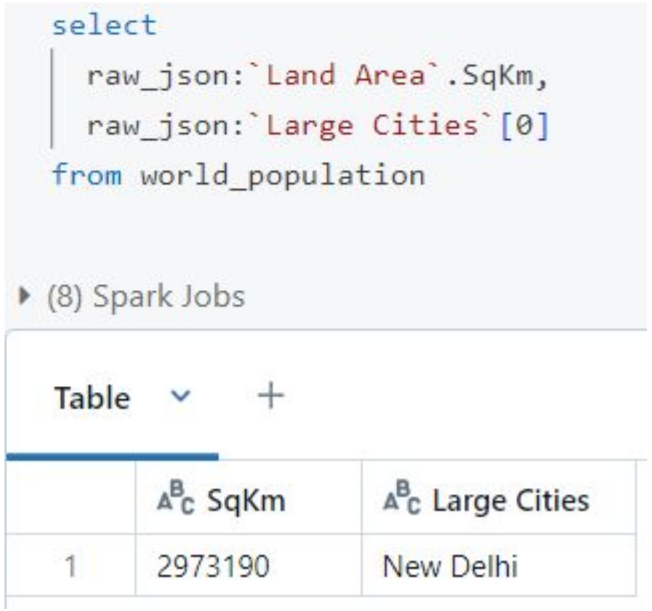
Accessing JSON and Array

Use dot operator for nested structure

Use array index (0-based) for array elements



```
{
  "Country":"India",
  "Population":"1428627663",
  "Land Area":{"SqKm":"2973190"},
  "Large Cities":["New Delhi", "Mumbai"]
}
```



```
select
  raw_json:`Land Area`.SqKm,
  raw_json:`Large Cities`[0]
from world_population
```

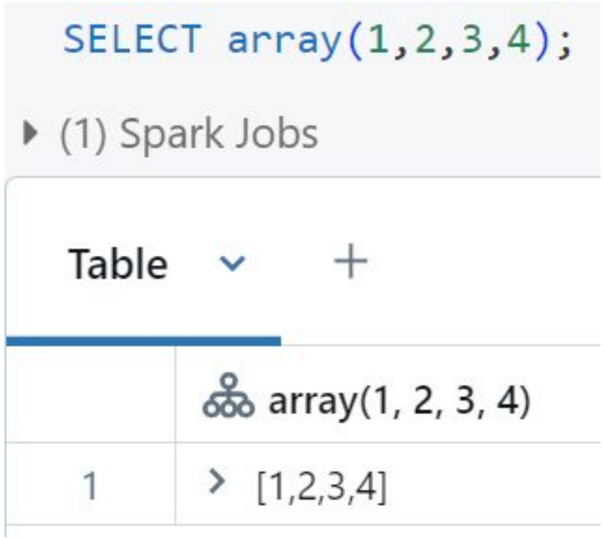
(8) Spark Jobs

	SqKm	Large Cities
1	2973190	New Delhi

Array Manipulation

Array operations handle the arrangement of elements:

- Merge multiple elements into one array: array()
- Merge multiple arrays into one array: flatten()

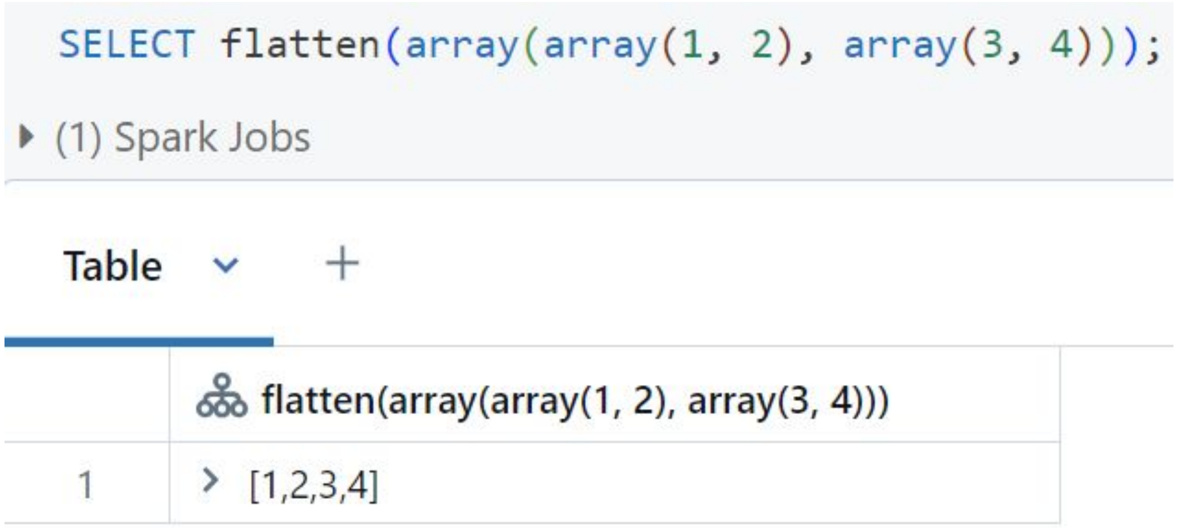


```
SELECT array(1,2,3,4);
```

▸ (1) Spark Jobs

Table ⌄ +

	array(1, 2, 3, 4)
1	› [1,2,3,4]



```
SELECT flatten(array(array(1, 2), array(3, 4)));
```

▸ (1) Spark Jobs

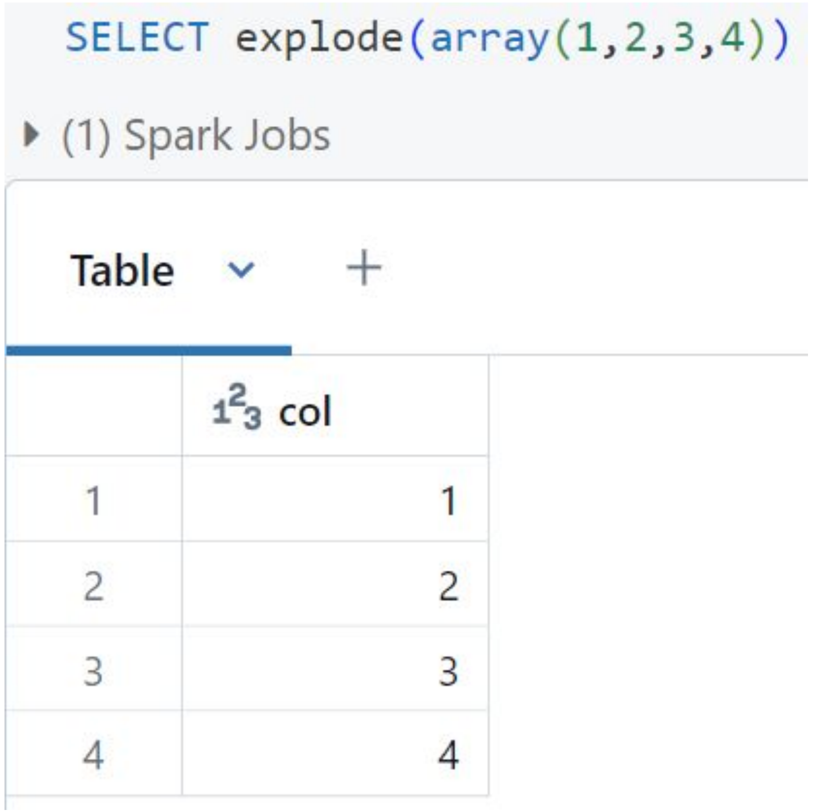
Table ⌄ +

	flatten(array(array(1, 2), array(3, 4)))
1	› [1,2,3,4]

Array Manipulation

Array operations handle the arrangement of elements:

- Split array into a list: `explode()`



```
SELECT explode(array(1,2,3,4))
```

	col
1	1
2	2
3	3
4	4

Functional Programming Functions

Allow you to define functions that manipulate arrays in SQL: higher-order function takes an array, implements how the array is processed, and what the result of the computation will be. It delegates to a lambda function how to process each item in the array

- Higher-order functions
- Anonymous (lambda) functions

```
SELECT   key,
         values,
         REDUCE(values, 0, (value, acc) -> value + acc, acc -> acc) total,
FROM     table_with_key_and_multiple_values
```

Module 3.4: Populating Changing Data

Changing Data

`INSERT`: Inserts new rows into a table and optionally truncates the table or partitions

`MERGE`: Merges a set of updates, insertions, and deletions based on a source table into a target Delta table

`COPY INTO`: Loads data from a file location into a Delta table

INSERT

`INSERT INTO with REPLACE WHERE` **:** Existing rows that meet that condition will be removed. New row is inserted.

```
INSERT INTO [ TABLE ] table_name
    REPLACE WHERE predicate
    query
```

```
INSERT INTO events
REPLACE WHERE eventId = 3
VALUES (3, 'Mary piano', 'Piano', '2024-09-30 12:00:00', '2024-09-30');
```

MERGE

`MERGE`: Check the data existence in two tables and perform actions based on whether the data exists in any or both of the tables. Used for tasks such as deduplicating data, upserting change data, and applying SCD Type 2 operations

```
MERGE INTO target_table_name
  USING source_table_reference ON merge_condition
    WHEN MATCHED [ AND matched_condition ] THEN action |
    WHEN NOT MATCHED [BY TARGET] [ AND ... ] THEN action |
    WHEN NOT MATCHED BY SOURCE [ AND ... ] THEN action
```

MERGE Logic

```
MERGE INTO target USING source
  ON target.key = source.key
    WHEN NOT MATCHED THEN INSERT *
    WHEN NOT MATCHED BY SOURCE THEN DELETE
    WHEN MATCHED THEN UPDATE SET target.updated_at = source.updated_at
```

If the record is in source but not in target, insert into target

If the record is not in source but in target, delete from target

If the record is both in source and in target, update the field in target to match source

COPY INTO

Bulk loading from files using `COPY INTO`

```
COPY INTO target_table [ BY POSITION | ( col_name [ ... ] ) ]
  FROM { source_clause | ( SELECT expression_list FROM source_clause ) }
  FILEFORMAT = data_source
  [ VALIDATE [ ALL | num_rows ROWS ] ]
  [ FILES = ( file_name [, ...] ) | PATTERN = glob_pattern ]
  [ FORMAT_OPTIONS ( { data_source_reader_option = value } [, ...] ) ]
  [ COPY_OPTIONS ( { copy_option = value } [, ...] ) ]
```

Not recommended by Databricks any more but still in exam guide

COPY INTO

Simple example:

```
COPY INTO target_table
  FROM (SELECT col1, col2 FROM '<external location>')
  FILEFORMAT = CSV
  PATTERN = 'folder1/file_[a-g].csv'
  FORMAT_OPTIONS('header' = 'true')
```

- Supports a variety of formats: CSV, JSON, XML, parquet, AVRO, BINARYFILE, ORC, and TEXT
- Supports pattern matching for file names
- Only-once copying: Once copied, Databricks will not load the file again when rerunning COPY INTO command

Module 4: Data Visualization and Dashboarding

Learning Objectives

Module 4: Data Visualization and Dashboarding

By the end of this module, you will learn using Databricks SQL to create and customize diverse visualizations. You will learn to use dashboards to combine multiple visualizations with formatting options, query parameters, and refresh schedules. You will also learn how to share dashboards.

Module 4.1: Visualization Fundamentals

Visualization Overview

Exploratory data analysis (EDA): a process of identifying patterns in data, such as outliers and unexpected features

- exploring data sets
- summarize their main characteristics
- identify any patterns with the data.
- learn about a data set to determine its readiness for analysis
- Identify what techniques to apply for data preparation
- determine which algorithms you choose to apply for training ML models.

Visualization Overview

Exploratory data analysis (EDA):

- Create data visualizations in a Databricks SQL notebook
  - Demo
- Create data visualizations in a Databricks SQL dashboard
  - Demo

Basic Visualization Types

Basic Visualization Types

- Bar chart: shows change/difference in metrics over time or across categories
  - Histogram: plots the frequency that a given value occurs in a dataset.
- Line: presents the change in one or more metrics over time
- Pie: shows proportionality between metrics
- Scatter visualizations: shows the relationship between two numerical variables
  - Bubble chart: scatters charts where the size of each point marker reflects a relevant metric.

Extended Visualization Types

Extended Visualization Types

- Area visualization: shows sales funnel changes through time.
- Box chart: compares the value ranges across categories and visualize the statistics characteristics
- Counter chart: displays a single value prominently
- Heatmap chart: visualizes numerical data using colors

Geographical Visualization Types

Geographical Visualization Types

- Point map: displays quantitative data as symbols at specific map locations
- Marker map:: places a marker at a set of coordinates on the map
- Choropleth map: colored geographic localities such as countries or states according to the values

Combo Visualization Types

Combo Visualization Types

- Combo chart: combine line and bar charts to present the changes over time with proportionality.
- Dual-axis combo chart: show two different y-axes

Advanced Visualization Types

Advanced Visualization Types

- Cohort: examines the outcomes through a set of stages
- Funnel visualization: the change in a metric at different stages
- Sankey diagram: visualizes the flow from one set of values to another
- Sunburst diagram helps visualize hierarchical data using concentric circles
- Word cloud: represents the frequency a word occurs in the data

Customizable Table

- Table visualization: shows data in a standard table but allows you to manually reorder, hide, and format the data
- Pivot: aggregates records from a query result into a tabular display

Visualization Customization

Create customized data visualizations to aid in data storytelling.

Demo

- Create a dashboard using multiple existing visualizations from Databricks SQL Queries.
- Add visual appeal through formatting
- Change the colors of the visualizations in a dashboard.

Visualization Customization

- Notebook Visualization Types
- Dashboard visualizations types

Module 4.2: Dashboard Management

Dashboard Parameter

- Query parameters: Change the output of underlying queries within a dashboard
    - "Query Based Dropdown List": create a query parameter from the distinct output of a different query.
- Dashboard parameter

Sharing Dashboard

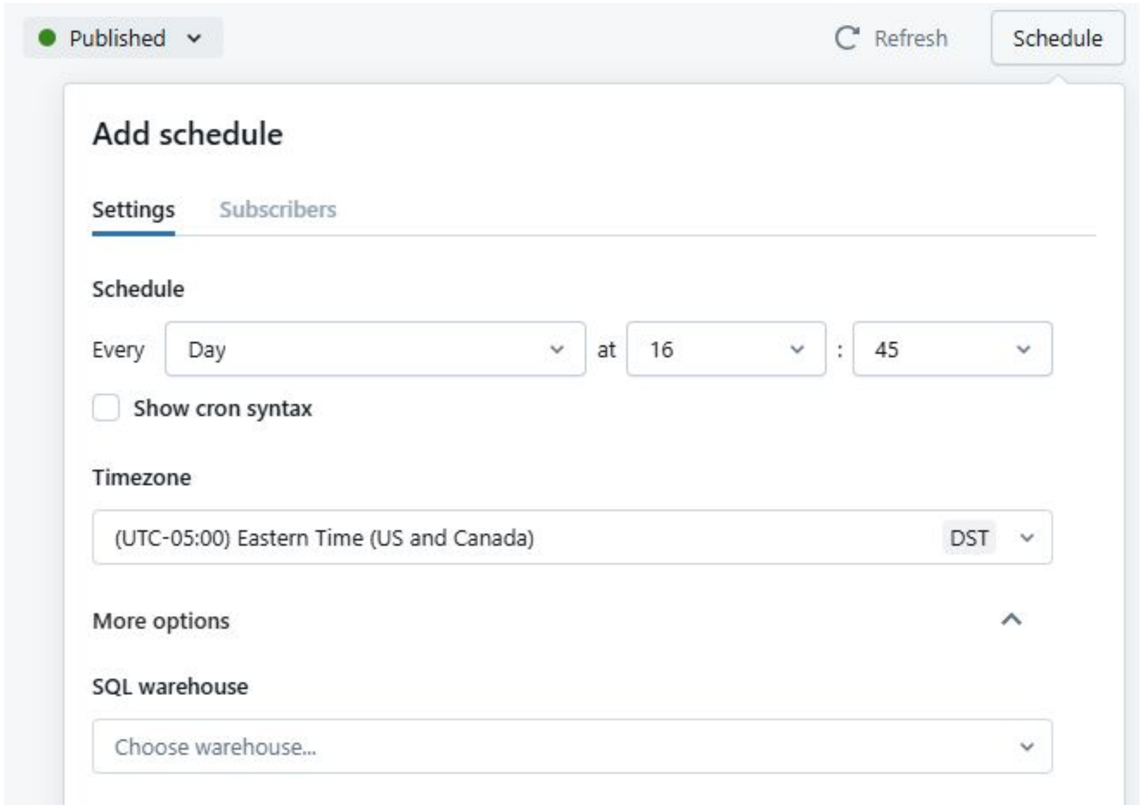
- Share dashboards with users and groups in your Databricks account
  - Built into the Databricks IAM platform, which integrates directly with many popular IDPs so that you can share your analysis with anyone in your organization
  - Recommended to only share the published versions
  - Share with designated audience
    - Can share with All Workspace Users. Workspace users with access to your dashboard can access both the draft or published versions
    - Anyone in my account can view: View-only copy of a dashboard published with embedded credentials
- Shared privileges: Manage, Edit, Run, and View

Publishing and Embedding Dashboard

- Enabling dashboard embedding
- Credentials
  - Embedded credential
  - Not Embedded: dashboard viewer’s own data and compute credentials
    - Can only used by users in the workspace
- Getting embedding code

Scheduled Refresh

Must be published with embedded credentials



Add schedule

Schedule

Every Day at 16 : 45

☐ Show cron syntax

Timezone

(UTC-05:00) Eastern Time (US and Canada) DST

More options

SQL warehouse

Choose warehouse...

Module 5:
Analytics Applications

Learning Objectives

Module 5: Analytics Applications

By the end of this module, you will learn the key statistic terms and key metrics of distributions for summarizing and comparing data, including discrete and continuous statistics. You will learn data enhancement and blending for enriching datasets and combining information from multiple sources and perform last-mile ETL to enable project-specific transformations. Finally, you will learn how to configure alerts, and monitor and troubleshoot the system by checking query history.

Module 5.1: Statistics Overview

Discrete and Continuous statistics

Aspect	Discrete Statistics	Continuous Statistics
Data Type	Countable, distinct values	Any value within a range
Representation	Bar charts, pie charts	Histograms, line graphs
Example	Number of students in a class	Heights of students
Statistical Tools	PMF, chi-square tests	PDF, regression analysis

Descriptive Statistics vs Inferential Statistics

Aspect	Descriptive Statistics	Inferential Statistics
Purpose	Summarizes data	Makes predictions/inferences
Data Scope	Limited to given data	Generalizes to the population
Tools	Central tendency, variability	Hypothesis testing, estimation
Use of Probability	Not involved	Central to methods

Key Statistics Metrics

Metric	What It Describes
Mean, Median, Mode	Central location of data
Range, Variance, SD, IQR	Spread or variability of data
Skewness, Kurtosis	Shape of the distribution
Percentiles, Quartiles	Relative position of data points
PDF, PMF, CDF	Probabilities for values or ranges

Module 5.2: Analytics Flow

Data Enhancement

Enriching or augmenting datasets to improve their quality, utility, or value for analytical and operational purposes

- Data Cleaning
  - Handling Missing Values
  - Removing Duplicates
  - Standardizing Formats
- Data Transformation
  - Feature Engineering: Create new features from existing data
  - Normalization and Scaling
  - Imputation: Replace missing or invalid values with statistically derived values
- Imputation: Replace missing or invalid values with statistically derived values
  - Joining External Data

Data Blending

Combining data from multiple sources to create a unified, enriched dataset for analysis

- Data Sources:
  - Structured (e.g., relational databases, CSV files).
  - Semi-structured (e.g., JSON, XML).
  - Unstructured (e.g., text, images).
  - Streaming data (e.g., Kafka, Event Hubs).
- Blending Operations:
  - Joining: Combining datasets based on a common key.
  - Union: Appending data with the same schema.
  - Aggregation: Summarizing data across multiple sources.
  - Transformation: Standardizing and aligning data for blending.

Last-mile ETL

Ensures that the final dataset is clean, enriched, and ready for specific business use cases

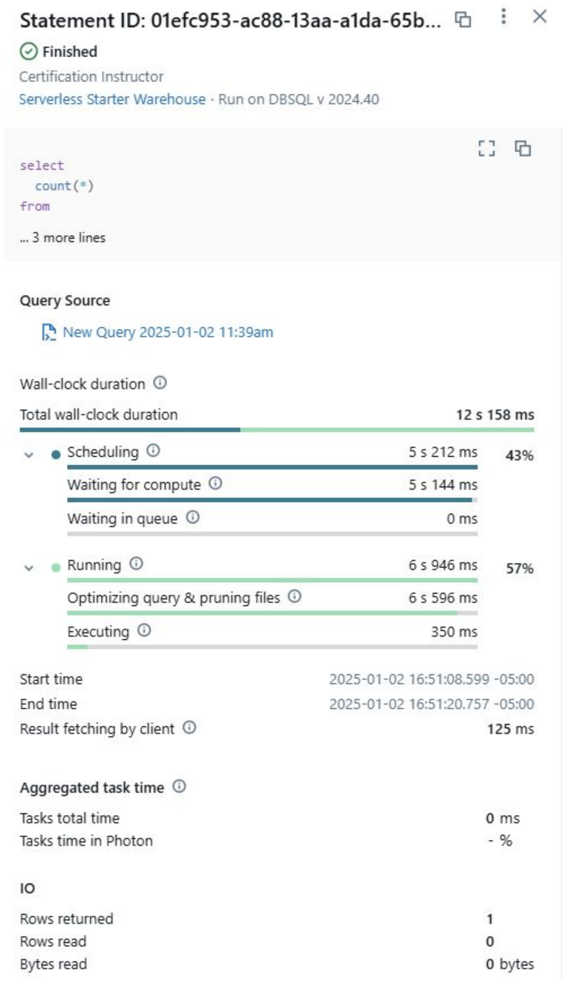
- Dashboard-Specific Transformation
- Data is optimized to enable quick queries and smooth interaction in dashboards
  - Create a pre-aggregated data cube
  - Calculate as many metrics as possible
- Data is formatted to match the requirements of the BI tool

Module 5.3:
Query Management

Query History

Demo: Query History

Databricks SQL UI cache: Per user caching of all query and dashboard results in the Databricks SQL UI. When users first open a dashboard or SQL query, the Databricks SQL UI cache displays the most recent query result, including the results from scheduled executions.


Statement ID: 01efc953-ac88-13aa-a1da-65b...

Finished

Certification Instructor

Serverless Starter Warehouse · Run on DBSQL v 2024.40

```
select
  count(*)
from
```

... 3 more lines

Query Source

New Query 2025-01-02 11:39am

Wall-clock duration

Total wall-clock duration	12 s 158 ms	
Scheduling	5 s 212 ms	43%
Waiting for compute	5 s 144 ms	
Waiting in queue	0 ms	
Running	6 s 946 ms	57%
Optimizing query & pruning files	6 s 596 ms	
Executing	350 ms	

Start time	2025-01-02 16:51:08.599 -05:00
End time	2025-01-02 16:51:20.757 -05:00
Result fetching by client	125 ms

Aggregated task time

Tasks total time	0 ms
Tasks time in Photon	- %

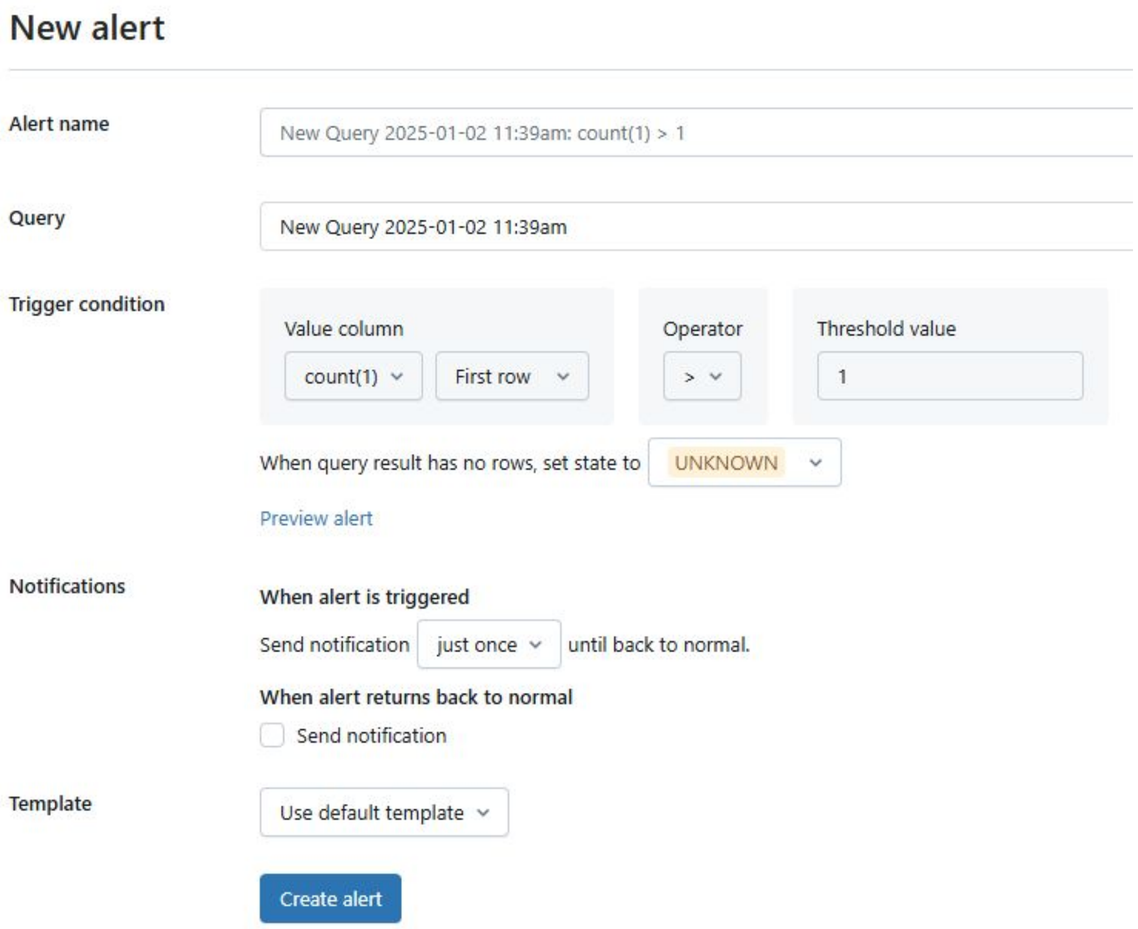
IO

Rows returned	1
Rows read	0
Bytes read	0 bytes

Alert

Demo:

- Create alert
- Apply notification
- Add schedule
- Define destination



New alert

Alert name: New Query 2025-01-02 11:39am: count(1) > 1

Query: New Query 2025-01-02 11:39am

Trigger condition:
Value column: count(1), First row
Operator: >
Threshold value: 1

When query result has no rows, set state to UNKNOWN

Preview alert

Notifications:
When alert is triggered
Send notification just once until back to normal.
When alert returns back to normal
☐ Send notification

Template: Use default template

Create alert

Ready, Set, Pass!

Exam Information

- Official Exam Website:
  - 90 min, 45 questions
  - Fee: $200
  - Exam registration
    - Remote
  - All questions are multiple choice questions (one correct answer out of 4 or 5 choices)
- Available resources
  - Official Exam Guide
  - This course (with Github repository)

Good Luck!