DP-750 Exam Prep

Overview

As a candidate for this Microsoft Certification, you should have subject matter expertise in integrating and modeling data, building and deploying optimized pipelines, and troubleshooting and maintaining workloads in Azure Databricks. You should also have experience applying data quality and data governance best practices in Unity Catalog.

You need to know how to ingest and transform data by using Structured Query Language (SQL) and Python. You need experience with software development lifecycle (SDLC) practices, including Git. Additionally, you should be familiar with Microsoft Entra, Azure Data Factory, and Azure Monitor.

Your responsibilities for this role include:

- Setting up and configuring an Azure Databricks environment.
- Securing and governing Unity Catalog objects.
- Preparing and processing data.
- Deploying and maintaining data pipelines and workloads.

You work closely with administrators, platform architects, solution architects, data scientists, and data analysts to design, deploy, and secure data engineering solutions by using Azure Databricks.

Skills at a glance

- Set up and configure an Azure Databricks environment (15–20%)
- Secure and govern Unity Catalog objects (15–20%)
- Prepare and process data (30–35%)
- Deploy and maintain data pipelines and workloads (30–35%)

Set up and configure an Azure Databricks environment (15–20%)

- **Select and configure compute in a workspace**
    - Choose an appropriate compute type, including job compute, serverless, warehouse, classic compute, and shared compute
    - Configure compute performance settings, including CPU, node count, autoscaling, termination, node type, cluster size, and pooling
    - Configure compute feature settings, including Photon acceleration, Azure Databricks runtime/Spark version, and machine learning
    - Install libraries for a compute resource
    - Configure access permissions to a compute resource
- **Create and organize objects in Unity Catalog**
    - Apply naming conventions based on requirements, including isolation, development environment, and external sharing
    - Create a catalog based on requirements
    - Create a schema based on requirements
    - Create volumes based on requirements
    - Create tables, views, and materialized views
    - Implement a foreign catalog by configuring connections
    - Implement data definition language (DDL) operations on managed and external tables
    - Configure AI/BI Genie instructions for data discovery

Secure and govern Unity Catalog objects (15–20%)

- **Secure Unity Catalog objects**
    - Grant privileges to a principal (user, service principal, or group) for securable objects in Unity Catalog
    - Implement table- and column-level access control and row-level security
    - Access Azure Key Vault secrets from within Azure Databricks

    - Authenticate data access by using service principals
    - Authenticate resource access by using managed identities
- **Govern Unity Catalog objects**
    - Create, implement, and preserve table and column definitions and descriptions for data discovery
    - Configure attribute-based access control (ABAC) by using tags and policies
    - Configure row filters and column masks
    - Apply data retention policies
    - Set up and manage data lineage tracking by using Catalog Explorer, including owner, history, dependencies, and lineage
    - Configure audit logging
    - Design and implement a secure strategy for Delta Sharing

Prepare and process data (30–35%)

- **Design and implement data modeling in Unity Catalog**
    - Design logic for data ingestion and data source configuration, including extraction type and file type
    - Choose an appropriate data ingestion tool, including Lakeflow Connect, notebooks, and Azure Data Factory
    - Choose a data loading method, including batch and streaming
    - Choose a data table format, such as Parquet, Delta, CSV, JSON, or Iceberg
    - Design and implement a data partitioning scheme
    - Choose a slowly changing dimension (SCD) type
    - Choose granularity on a column or table based on requirements
    - Design and implement a temporal (history) table to record changes over time
    - Design and implement a clustering strategy, including liquid clustering, Z-ordering, and deletion vectors

    - Choose between managed and unmanaged tables
- **Ingest data into Unity Catalog**
    - Ingest data by using Lakeflow Connect, including batch and streaming
    - Ingest data by using notebooks, including batch and streaming
    - Ingest data by using SQL methods, including CREATE TABLE … AS (CTAS), CREATE OR REPLACE TABLE, and COPY INTO
    - Ingest data by using a change data capture (CDC) feed
    - Ingest data by using Spark Structured Streaming
    - Ingest streaming data from Azure Event Hubs
    - Ingest data by using Lakeflow Spark Declarative Pipelines, including Auto Loader
- **Cleanse, transform, and load data into Unity Catalog**
    - Profile data to generate summary statistics and assess data distributions
    - Choose appropriate column data types
    - Identify and resolve duplicate, missing, and null values
    - Transform data, including filtering, grouping, and aggregating data
    - Transform data by using join, union, intersect, and except operators
    - Transform data by denormalizing, pivoting, and unpivoting data
    - Load data by using merge, insert, and append operations
- **Implement and manage data quality constraints in Unity Catalog**
    - Implement validation checks, including nullability, data cardinality, and range checking
    - Implement data type checks
    - Implement schema enforcement and manage schema drift
    - Manage data quality with pipeline expectations in Lakeflow Spark Declarative Pipelines

Deploy and maintain data pipelines and workloads (30–35%)

- **Design and implement data pipelines**
  - Design order of operations for a data pipeline
  - Choose between notebook and Lakeflow Spark Declarative Pipelines
  - Design task logic for Lakeflow Jobs
  - Design and implement error handling in data pipelines, notebooks, and jobs
  - Create a data pipeline by using a notebook, including precedence constraints
  - Create a data pipeline by using Lakeflow Spark Declarative Pipelines
- **Implement Lakeflow Jobs**
  - Create a job, including setup and configuration
  - Configure job triggers
  - Schedule a job
  - Configure alerts for a job
  - Configure automatic restarts for a job or a data pipeline
- **Implement development lifecycle processes in Azure Databricks**
  - Apply version control best practices using Git
  - Manage branching, pull requests, and conflict resolution
  - Implement a testing strategy, including unit tests, integration tests, end-to-end tests, and user acceptance testing (UAT)
  - Configure and package Databricks Asset Bundles
  - Deploy a bundle by using the Azure Databricks command-line interface (CLI)
  - Deploy a bundle by using REST APIs
- **Monitor, troubleshoot, and optimize workloads in Azure Databricks**
  - Monitor and manage cluster consumption to optimize performance and cost

- Troubleshoot and repair issues in Lakeflow Jobs, including repair, restart, stop, and run functions
- Troubleshoot and repair issues in Apache Spark jobs and notebooks, including performance tuning, resolving resource bottlenecks, and cluster restart
- Investigate and resolve caching, skewing, spilling, and shuffle issues by using a Directed Acyclic Graph (DAG), the Spark UI, and query profile
- Optimize Delta tables for performance and cost, including OPTIMIZE and VACUUM commands
- Implement log streaming by using Log Analytics in Azure Monitor
- Configure alerts by using Azure Monitor