DP-203

Microsoft Azure Data Engineer

**Exam number:**  DP-203

**Exam title: Microsoft Azure Data Engineer**

**Publish date:**

**Language(s) this exam will be available in:**

**Audience** (IT professionals, Developers, Information workers, etc.):

**Technology:**

**Credit type** (example: MCSA)**:**

**Exam provider** (VUE, Certiport, or both)**:**

# Exam Design

**Audience Profile**  
The Azure Data Engineer integrates, transforms, and consolidates data from various structured and unstructured data systems into a structure suitable for building analytics solutions. The Data Engineer explores data to investigate specific data questions posed by stakeholders. The Data Engineer builds and maintains secure and compliant data processing pipelines by using different tools and techniques. This role uses various Azure data services and languages to store and produce cleansed and enhanced datasets for analysis.

A Data Engineer ensures that data pipelines and data stores are high-performing, efficient, organized, and reliable given a set of business requirements and constraints. The Data Engineer deals with unanticipated issues swiftly, and minimizes ~~any~~ data loss. The Data Engineer designs, implements, monitors, and optimizes data platforms to meet the needs of the data pipelines.

The Data Engineer must have strong knowledge of data processing languages using SQL, Python, or Scala. Understand parallel processing, and data architecture patterns.

|  |  |
| --- | --- |
| In scope technologies | Out-of-scope technologies |
| Azure Data Factory  Azure Synapse Analytics  Stream Analytics  Event Hubs  Data Lake  Azure Databricks | Cosmos DB  IoT Hub  Machine Learning  PowerBI  Azure SQL DB  Analysis Services  Data Explorer  HDInsight  Delta Lake  Azure Monitor (Log Analytics)  Blob Storage  Azure KeyVault |

Skills measured

# Design and Implement Data Storage

* 1. Design a data storage structure
     + Design a data lake solution
     + Recommend file types for storage
     + Recommend file types for analytical queries
     + Design for efficient querying
     + Design for data pruning
     + Design a folder structure that represents levels of data transformation
     + Design a distribution strategy
     + Design a data archiving solution
  2. Design a partition strategy
     + Design a partition strategy for files
     + Design a partition strategy for analytical workloads
     + Design a partition strategy for efficiency/performance
     + Design a partition strategy for Synapse Analytics
     + Identify when partitioning is needed in Azure Data Lake Store Gen2
  3. Design the serving layer
     + Design star schemas
     + Design slowly changing dimensions
     + Design a dimensional hierarchy
     + Design a solution for temporal data
     + Design for incremental loading
     + Design analytical stores
     + Design metastores in Azure Synapse and Azure Databricks
  4. Implement physical data storage structures
     + Implement compression
     + Implement partitioning
     + Implement sharding
     + Implement different table geometries with Synapse Analytics Pools
     + Implement data redundancy
     + Implement distributions
     + Implement data archiving
  5. Implement logical data structures
     + Build a temporal data solution
     + Build a slowly changing dimension
     + Build a logical folder structure
     + Build external tables
     + Implement file and folder structures for efficient querying and data pruning
  6. Implement the serving layer
     + Deliver data in a relational star schema
     + Deliver data in parquet files
     + Maintain metadata
     + Implement a dimensional hierarchy

# Design and Develop Data Processing

* 1. Ingest and transform data
     + Transform data by using Apache Spark
     + Transform data by using TSQL
     + Transform data by Azure Data Factory
     + Transform data by using Synapse pipelines
     + Transform data by using Stream Analytics
     + Cleanse data
     + Split data
     + Shred JSON
     + Encode and decode data
     + Configure error handling for the transformation
     + Normalize and de-normalize values
     + Transform data by using Scala
     + Perform data exploratory analysis
  2. Design and develop a batch processing solution
     + Develop batch processing solutions by using Azure Data Factory, Data Lake, Spark, Synapse pipelines, Polybase and Databricks
     + Create data pipelines
     + Design and implement incremental data loads
     + Design and develop slowly changing dimensions
     + Handle security and compliance requirements
     + Scale resources
     + Configure batch size
     + Design and create tests for data pipelines
     + Integrate Jupyter/ipython notebooks into a data pipeline
     + Handle duplicate data
     + Handle missing data
     + Handle late-arriving data
     + Upsert data
     + Regress to a previous state
     + Design and configure exception handling
     + Configure batch retention
     + Design a batch processing solution
     + Debug spark jobs using the Spark UI
  3. Design and develop a stream processing solution
     + Develop stream processing solutions by using Stream Analytics, Databricks, Event Hubs
     + Process data by using Spark structured streaming
     + Monitor for performance and functional regressions
     + Design and create windowed aggregates
     + Handle schema drift
     + Process time series data
     + Process across partitions
     + Process within one partition
     + Configure checkpoints/watermarking during processing
     + Scale resources
     + Design and create tests for data pipelines
     + Optimize pipelines for analytical or transactional purposes
     + Handle interruptions
     + Design and configure exception handling
     + Upsert data
     + Replay archived stream data
     + Design a stream processing solution
  4. Manage batches and pipelines
     + Trigger batches
     + Handle failed batch loads
     + Validate batch loads
     + Manage data pipelines in Azure Data Factory/Synapse pipelines
     + Schedule data pipelines in Azure Data Factory/Synapse pipelines
     + Implement version control for pipeline artifacts
     + Manage Spark jobs in a pipeline

# Design and Implement Data Security

* 1. Design security for data policies and standards
     + Design data encryption for data at rest and in transit.
     + Design a data auditing strategy
     + Design a data masking strategy
     + Design for data privacy
     + Design a data retention policy.
     + Design to purge data based on business requirements.
     + Design Azure role-based access control (Azure RBAC) and POSIX-like ACL for Azure Data Lake Storage Gen2
     + Design row-level and column-level security
  2. Implement data security
     + Implement data masking.
     + Encrypt data at rest and in motion.
     + Implement row-level and column-level security
     + Implement Azure RBAC
     + Implement POSIX-like ACLs for Azure Data Lake Storage Gen2
     + Implement a data retention policy
     + Implement a data auditing strategy
     + Manage identities, keys, and secrets across different data platform technologies
     + Implement secure endpoints (private and public)
     + Implement resource tokens within Databricks
     + Load a DataFrame with sensitive information
     + Write encrypted data to tables or Parquet files
     + Manage sensitive information

# Monitor and Optimize Data Storage and Data Processing

* 1. Monitor data storage and data processing
     + Implement logging used by Azure Monitor
     + Configure monitoring services
     + Measure performance of data movement
     + Monitor and update statistics about data across the system
     + Monitor data pipeline performance
     + Measure query performance
     + Monitor cluster performance
     + Understand custom logging options
     + Schedule and monitor pipeline tests
     + Interpret Azure Monitor metrics and logs
     + Interpret a Spark DAG
  2. Optimize and Troubleshoot Data Storage and Data Processing
     + Compact small files
     + Re-write user defined Functions (UDFs)
     + Handle skew in data
     + Handle data spill
     + Tune shuffle partitions
     + Find shuffling in a pipeline
     + Optimize resource management
     + Tune queries using indexers
     + Tune queries using cache
     + Optimize pipelines for analytical or transactional purposes
     + Optimize pipeline for descriptive versus analytical workloads
     + Troubleshoot a failed spark job
     + Troubleshoot a failed pipeline run