SNOWPRO™ ADVANCED: DATA ENGINEER EXAM STUDY GUIDE DEA-C01

Last Updated: September 27, 2022

SNOWPRO™ STUDY GUIDE OVERVIEW

This study guide highlights concepts that may be covered on Snowflake's **SnowPro™ Advanced: Data Engineer Certification** exam.

This document introduces relevant information that may appear on the **SnowPro Advanced: Data Engineer** Exam. This document should serve as an introduction to the knowledge and skills required to guide your preparation. The material contained within this study guide is not intended to guarantee a passing score on any Snowflake certification exam.

For an overview and more information on the SnowPro™ Core Certification exam or SnowPro™ Advanced Certification series, please navigate here.

TABLE OF CONTENTS

SNOWPRO™ ADVANCED: DATA ENGINEER CERTIFICATION OVERVIEW

The **SnowPro™ Advanced: Data Engineer** tests advanced knowledge and skills used to apply comprehensive data engineering principles using Snowflake.

This certification will test the ability to:

- Source data from Data Lakes, APIs, and on-premises
- Transform, replicate, and share data across cloud platforms
- Design end-to-end near real-time streams
- Design scalable compute solutions for DE workloads
- Evaluate performance metrics

Target Audience:

2 + years of data engineering experience, including practical experience using Snowflake for DE tasks; Candidates should have a working knowledge of Restful APIs, SQL, semi-structured datasets, and cloud native concepts. Programming experience is a plus.

SNOWPRO™ ADVANCED: DATA ENGINEER SUBJECT AREA BREAKDOWN

This exam guide includes test domains, weightings, and objectives. It is not a comprehensive listing of all the content that will be presented on this examination. The table below lists the main content domains and their weighting ranges.

Domain	Estimated Percentage Range of Exam Questions
1.0 Data Movement	25-30%
2.0 Performance Optimization	20-25%
3.0 Storage and Data Protection	10-15%
4.0 Security	10-15%
5.0 Data Transformation	25-30%

SNOWPRO™ ADVANCED: DATA ENGINEER PREREQUISITE

Eligible individuals must hold an active SnowPro Core Certified credential. If you feel you need more guidance on the fundamentals, please see the SnowPro Core Study guide.

Snowflake recommends that examinees have at least 2 years of hands-on practical Snowflake implementation experience prior to attempting any of the SnowPro Advanced exams.

For the SnowPro Advanced: Data Engineer Certification exam, **we recommend individuals have at least 2 + years of hands-on Snowflake Practitioner experience in a Data Engineering role prior to attempting this exam.** The exam will assess skills through scenario-based questions and real-world examples.

RECOMMENDATIONS AND USING THE GUIDE

This guide will show the Snowflake topics and subtopics covered on the exam. Following the topics will be additional resources consisting of videos, documentation, blogs, and/or exercises to help you understand data engineering with Snowflake.

Estimated length of study guide: 10 – 13 hours

STEPS TO SUCCESS

1. Review Data Engineer Exam Guide
2. For training, attend Snowflake's Instructor Led Data Engineering Course
3. Review and study applicable white papers and documentation
4. Get hands-on practical experience with relevant business requirements using Snowflake
5. Attend Snowflake Webinars
6. Attend Snowflake Virtual Hands-on Labs for more hands-on practical experience
7. Schedule your exam
8. Take your exam!

Additional Snowflake Asset to check out for Data Engineering:

Cloud Data Engineering for Dummies

SNOWPRO ADVANCED: DATA ENGINEER DOMAINS & OBJECTIVES

1.0 Domain: Data Movement

1.1 Given a data set, load data into Snowflake.
- Outline considerations for data loading
- Define data loading features and potential impact

1.2 Ingest data of various formats through the mechanics of Snowflake.
- Required data formats
- Outline Stages

1.3 Troubleshoot data ingestion.

1.4 Design, build and troubleshoot continuous data pipelines.
- Design a data pipeline that forces uniqueness but is not unique.
- Stages
- Tasks
- Streams
- Snowpipe
- Auto ingest as compared to Rest API

1.5 Analyze and differentiate types of data pipelines.
- Understand Snowpark architecture (client vs server)
- Create and deploy UDFs and Stored Procedures using Snowpark
- Design and use the Snowflake SLQ API

1.6 Install, configure, and use connectors to connect to Snowflake.

1.7 Design and build data sharing solutions.
- Implement a data share
- Create a secure view
- Implement row level filtering

1.8 Outline when to use External Tables and define how they work.
- Partitioning external tables
- Materialized views
- Partitioned data unloading

Data Movement Study Resources:

Lab Guides
Accelerating Data Engineering with Snowflake & dbt (lab guide)
Auto-Ingest Twitter Data into Snowflake (lab guide)

Automating Data Pipelines to Drive Marketing Analytics with Snowflake & Fivetran (lab guide)

Reading Assets
Support for Calling External functions via Google Cloud API Gateway Now in Public Preview (blog)
Snowflake and Spark, Part 2: Pushing Spark Query (Blog)
Fetching Query Results From Snowflake (Blog)
Moving from On-Premises ETL to Cloud-Driven ELT (White paper)

Snowflake Documentation
COPY INTO (Documentation)
Loading Data into Snowflake (Documentation)
DESCRIBE STAGE (Documentation)
Data Loading Tutorials(Documentation)
CREATE FILE FORMAT (Documentation)
Continuous Data Pipelines (Documentation)
VALIDATE_PIPE_LOAD (Documentation)
COPY_HISTORY (Documentation)
Databases, Tables & Views (Documentation)
CREATE STREAM (Documentation)
CREATE TASK (Documentation)
Connectors & Drivers (Documentation)
Sharing Data Securely in Snowflake (Documentation)
CREATE EXTERNAL TABLE (Documentation)

2.0 Domain: Performance Optimization

2.1 Troubleshoot underperforming queries.
- Identify underperforming queries
- Outline telemetry around the operation
- Increase efficiency
- Identify the root cause

2.2 Given a scenario, configure a solution for the best performance.
- Scale out as compared to scale in
- Clustering as compared to increasing warehouse size
- Query complexity
- Micro partitions and the impact of clustering
- Materialized views
- Search optimization

2.3 Outline and use caching features.

2.4 Monitor continuous data pipelines.

- Snowpipe
- Stages
- Tasks
- Streams

Performance Optimization Study Resources:

Lab Guides
Resource Optimization: Performance (lab guide)
Resource Optimization: Usage Monitoring (lab guide)
Building a Data Application (lab guide)

Reading Assets
Performance Impact from Local and Remote Disk Spilling (Blog)
Snowflake: Visualizing Warehouse Performance (Blog)
Caching in Snowflake Data Warehouse (Blog)

Snowflake Documentation
Queries (Documentation)
System Functions (Documentation)
Account Usage (Documentation)
QUERY_HISTORY, QUERY_HISTORY_BY_* (Documentation)
Analyzing Queries Using Query Profile (Documentation)
Databases, Tables & Views (Documentation)
Virtual Warehouses (Documentation)
COPY_HISTORY (Documentation)
LOAD_HISTORY View (Documentation)
TASK_HISTORY (Documentation)
COPY_HISTORY View (Documentation)
SHOW STREAMS (Documentation)
PIPE_USAGE_HISTORY View (Documentation)

3.0 Domain: Storage & Data Protection

3.1 Implement data recovery features in Snowflake.
- Time Travel
- Fail-safe

3.2 Outline the impact of Streams on Time Travel.

3.3 Use System Functions to analyze Micro-partitions.
- Clustering depth
- Cluster keys

3.4 Use Time Travel and Cloning to create new development environments.

- Backup databases
- Test changes before deployment
- Rollback

Storage & Data Protection Study Resources:

Lab Guides
Getting Started with Time Travel (lab guide)

Snowflake Documentation
Snowflake Time Travel & Fail-safe (Documentation)
Databases, Tables & Views (Documentation)
Parameter Hierarchy and Types (Documentation)
Database Replication and Failover/Failback (Documentation)
Continuous Data Pipelines (Documentation)
SYSTEM$CLUSTERING_INFORMATION (Documentation)
SYSTEM$CLUSTERING_DEPTH (Documentation)

4.0 Domain: Security

4.1 Outline Snowflake security principles.
- Authentication methods (Single Sign On (SSO), Key Authentication, Username/Password, Multi-factor Authentication (MFA))
- Role Based Access Control (RBAC)
- Column Level Security and how data masking works with RBAC to secure sensitive data

4.2 Outline the system defined roles and when they should be applied.
- The purpose of each of the System Defined Roles including best practices usage in each case
- The primary differences between SECURITYADMIN and USERADMIN roles
- The difference between the purpose and usage of the USERADMIN/ SECURITYADMIN roles and SYSADMIN

4.3 Manage Data Governance.
- Explain the options available to support column level security including Dynamic Data Masking and External Tokenization
- Explain the options available to support row level security using Snowflake Row Access Policies
- Use DDL required to manage Dynamic Data Masking and Row Access Policies
- Use methods and best practices for creating and applying masking policies on data
- Use methods and best practices for Object Tagging

Security Study Resources:

Reading Assets
Snowflake RBAC Security Prefers Role Inheritance to Role Composition (Blog)

Snowflake Documentation
Managing Security in Snowflake (Documentation)
Managing Your User Preferences (Documentation)
Managing Governance in Snowflake (Documentation)
Stored Procedures (Documentation)
GRANT <privileges>…TO ROLE (Documentation)
CREATE MATERIALIZED VIEW (Documentation)

5.0 Domain: Data Transformation

5.1 Define User-Defined Functions (UDFs) and outline how to use them.
- Secure UDFs
- SQL UDFs
- JavaScript UDFs
- Returning table value as compared to scalar value

5.2 Define and create External Functions.
- Secure External Functions

5.3 Design, Build, and Leverage Stored Procedures.
- Transaction management

5.4 Handle and transform semi-structured data.
- Traverse and transform semi-structured data to structured data
- Transform structured to semi-structured data

5.5 Use Snowpark for data transformation.
- Query and filter data using the Snowpark library
- Perform data transformations using Snowpark (ie., aggregations)
- Join Snowpark dataframes

Data Transformation Study Resources:

Reading Assets
Snowflake For Data Engineering – Easily Ingest, Transform and Deliver Data for Up-To-The Moment Insight (white paper)

Bringing Extensibility to Data Pipelines: What's New with Snowflake External Functions (blog)
Generating a JSON Dataset Using Relational Data in Snowflake (blog)
Best Practices for Managing Unstructured Data (White paper)

Snowflake Documentation
UDFs (User-Defined Functions) (Documentation)
External Functions (Documentation)
CREATE EXTERNAL FUNCTION (Documentation)
CREATE API INTEGRATION (Documentation)
CREATE EXTERNAL FUNCTION (Documentation)
Transactions (Documentation)
Stored Procedures (Documentation)
TRY_PARSE_JSON (Documentation)
Queries (Documentation)
Semi-Structured Data (Documentation)
Databases, Tables & Views (Documentation)
Snowpark (Documentation)

Ready to register for an exam? Navigate here to get started.

SNOWPRO™ ADVANCED: DATA ENGINEER SAMPLE QUESTIONS

1. Running the below clustering information analysis function:

```
SELECT SYSTEM$CLUSTERING_INFORMATION(‘table1 , ‘(col1, col2)’)
```

on `TABLE1`, that is not clustered, will return which of the following?

a. An error: this function works only on clustered tables.

b. Clustering information on all tables: this function clusters all tables by default.

c. Clustering information: the information will be presented as if the table was clustered by `col1,col2`.

d. An error: this function does not accept lists of columns as a second parameter.

2. A Data Engineer has inherited a database and is monitoring a table with the below query every 30 days:

   ```
   SELECT SYSTEM$CLUSTERING_INFORMATION( 'orders', '(o_orderdate)');
   ```

   The Engineer gets the first two results (e.g., Day 0 and Day 30).

   ```
   -- DAY 0 -------
   {
     "cluster_by_keys" : "LINEAR(o_orderdate)",
     "total_partition_count" : 3218,
     "total_constant_partition_count" : 0,
     "average_overlaps" : 20.4133,
     "average_depth" : 11.4326,
     "partition_depth_histogram" : {
       "00000" : 0,
       "00001" : 0,
       "00002" : 0,
       "00003" : 0,
       "00004" : 0,
       "00005" : 0,
       "00006" : 0,
       "00007" : 0,
       "00008" : 0,
       "00009" : 0,
       "00010" : 993,
       "00011" : 841,
       "00012" : 748,
       "00013" : 413,
       "00014" : 121,
       "00015" : 74,
       "00016" : 16,
       "00032" : 12
     }
   }

   -- DAY 30 -------
   {
     "cluster_by_keys" : "LINEAR(o_orderdate)",
     "total_partition_count" : 3240,
     "total_constant_partition_count" : 0,
     "average_overlaps" : 64.1185,
     "average_depth" : 33.4704,
     "partition_depth_histogram" : {
       "00000" : 0,
       "00001" : 0,
       "00002" : 0,
       "00003" : 0,
   ```

```
      "00004" : 0,
      "00005" : 0,
      "00006" : 0,
      "00007" : 0,
      "00008" : 0,
      "00009" : 0,
      "00010" : 0,
      "00011" : 0,
      "00012" : 0,
      "00013" : 0,
      "00014" : 0,
      "00015" : 0,
      "00016" : 0,
      "00032" : 993,
      "00064" : 2247
    }
  }
```

How should the Engineer interpret these results?

  a. The table is well organized for queries that range over column `o_orderdate`. Over time, this organization is degrading.
  b. The table was initially well organized for queries that range over column `o_orderdate`. Over time this organization has improved further.
  c. The table was initially not organized for queries that range over column `o_orderdate`. Over time, this organization has changed.
  d. The table was initially poorly organized for queries that range over column `o_orderdate`. Over time, this organization has improved.

3. A Data Engineer is preparing to load staged data from an external stage using a task object.

   Which of the following practices will provide the MOST efficient load performance?

  a. Store the files on the external stage to ensure caching is maintained
  b. PUT all files in a single directory
  c. Limit file names to under 30 characters
  d. Organize files into logical paths that reflect a scheduling pattern

4. A Data Engineer is working on a project that requires data to be moved directly from an internal stage to an external stage.

   Which of the following is the QUICKEST way to accomplish this task?

   a. `COPY INTO @myExtStage` from (`SELECT $1, $2, ... @myInternalStage);`
   b. Copy the data from the internal stage to a table and then unload the data to an external stage
   c. `COPY INTO @myExtStage from @myInternalStage;`
   d. Write a custom script to move the data

5. The `S1` schema contains two permanent tables that were created as shown below:

```
CREATE TABLE table_a (c1 INT)
DATA_RETENTION_TIME_IN_DAYS = 10;

CREATE TABLE table_b (c1 INT);
```

   What will be the impact of running the following command?

```
ALTER SCHEMA S1 SET DATA_RETENTION_TIME_IN_DAYS = 20;
```

a. The retention time on `table_a` does not change; `table_b` is set to 20 days.
a. An error will be generated; a data retention time on a schema cannot be set.
b. The retention time on both tables will be set to 20 days.
c. The retention time will not change on either table.

Keys: 1) B

2) A

3) D

4) A

5) A