Snowflake SPS-C01

Snowflake SnowPro Specialty - Snowpark Certification Questions & Answers

Get Instant Access to Vital Exam Acing Materials | Study Guide | Sample Questions | Practice Test

SPS-C01

Snowflake Certified SnowPro Specialty - Snowpark

55 Questions Exam – Duration of 85 minutes

Table of Contents:

Discover More about the SPS-C01 Certification

Are you interested in passing the Snowflake SPS-C01 exam? First discover, who benefits from the SPS-C01 certification. The SPS-C01 is suitable for a candidate if he wants to learn about Specialty. Passing the SPS-C01 exam earns you the Snowflake Certified SnowPro Specialty - Snowpark title.

While preparing for the SPS-C01 exam, many candidates struggle to get the necessary materials. But do not worry; your struggling days are over. The SPS-C01 PDF contains some of the most valuable preparation tips and the details and instant access to useful SPS-C01 study materials **just at one click**.

Snowflake SPS-C01 SnowPro Specialty - Snowpark Certification Details:

Exam Name	Snowflake SnowPro Specialty - Snowpark
Exam Code	SPS-C01
Exam Price	$225 USD
Duration	85 minutes
Number of Questions	55
Passing Score	750 + Scaled Scoring from 0 - 1000
Recommended Training / Books	**Snowpark DataFrame Programming Training Course** **SnowPro Specialty: Snowpark Exam Study Guide** **Level Up Snowpark Essentials Track**
Schedule Exam	**PEARSON VUE**
Sample Questions	**Snowflake SPS-C01 Sample Questions**
Recommended Practice	**Snowflake Certified SnowPro Specialty - Snowpark Practice Test**

SPS-C01 Syllabus:

Section	Objectives
Snowpark Concepts - 15%	
Outline Snowpark architecture	- Lazy evaluation - Use of key objects • Snowpark DataFrames • User-Defined Functions (UDFs) • User-Defined Table Functions (UDTFs) • Stored procedures • File operations - Types of libraries (DataFrames, Machine Learning) • Anaconda repository (Python packages directly into Snowflake) • Other third-party libraries (not managed by Anaconda repository) - Client-side and server-side capabilities
Set-up Snowpark	- Installation • Versioning • Python environment - Development environments • Third-party tools • Snowflake Notebooks • Jupyter Notebooks • Microsoft Visual Studio Code (VS Code)
Snowpark API for Python - 30%	
Create and manage user sessions	- Account identifiers - Parameters for the CONNECT function

Section	**Objectives**
	- Authentication methods • Construct a dictionary • Key pair authentication • Snowflake CLI or .env parameters - Session creation - SessionBuilder - Session methods - Session attributes - Asyncjob
Use Snowpark with unstructured data	- Read files with SnowflakeFile object - Use UDFs and UDTFs to process files - Use stored procedures to process files
Create Snowpark DataFrames	- Multiple methods to create Snowpark DataFrames • From Snowflake tables/views • From Python objects (list, dictionary) • From SQL statements • From files (JSON, CSV, Parquet, XML) • From pandas DataFrames - Schemas (apply to DataFrames) - Data types (for example, IntegerType, StringType, DateType)
Operationalize UDFs and UDTFs in Snowpark	- Create UDFs from files (locally, on a stage) - Use Python modules (packaged Python code) with UDFs - Write Python function to create UDFs and UDTFs - Register UDFs and UDTFs (for example, session.utf(...), functions.utf(...)) - Secure UDFs and UDTFs • Use SQL to alter UDFs and UDTFs created with

Section	Objectives
	Snowpark • Grant access to UDFs and UDTFs to share code • Understanding how to grant object permissions so other Snowflake users can see and use the UDFs and UDTFs - Data types (type hints vs. registration API) • Provide the data types as parameters when creating a UDF or UDTF to return as Python hints/specify them as part of the registration - Compare scalar and vectorized operations
Operationalize Snowpark stored procedures	- Create stored procedures from files (locally, on stage) - Write Python functions to power stored procedures - Use Python modules (packaged code, Anaconda) with stored procedures - Register stored procedures - Make dependencies available to code - Secure stored procedures • Use SQL to alter stored procedures created with Snowpark • Caller versus owner rights - Use Snowpark Python stored procedures to run workloads - Data types (type hints vs. registration API) • Provide the data types as parameters when creating a stored procedure to return as Python hints/specify them as part of the registration - Create Directed Acyclic Graphs (tasks) executing stored procedures Python API - Bring Python modules (packaged code) to be used with

Section	Objectives
	UDFs • Stored procedures to enable reuse of code
Snowpark for Data Transformations - 35%	
Apply operations for filtering and transforming data	- Use scalar functions and operators - Sort and limit results - Input/output (parameters) - Snowpark DataFrames - Columns - Data type casting - Rows and data extraction from a Rows object
Clean and enrich data using Snowpark for Python	- Perform joins - Handle missing values - Sample data
Perform aggregate and set-based operations on DataFrames	- Functions - Window - Grouping - Table functions - UDFs
Transform semi-structured data in DataFrames	- Traverse semi-structured data - Explicitly cast values in semi-structured data - Flatten an array of objects into rows - Load semi-structured data into DataFrames
Persist the results of Snowpark DataFrames	- Create views from DataFrames - Save DataFrame results as Snowflake tables - Save DataFrame results as files in a stage
Perform DML operations using Snowpark DataFrames	- Delete data - Update data - Insert data - Merge data
Snowpark Performance Optimization - 20%	
Configure Snowpark-	- Use cases for Snowpark-optimized virtual warehouses

Section	Objectives
optimized warehouses	- Modify Snowpark-optimized virtual warehouse properties - Billing for Snowpark-optimized virtual warehouses - When to scale up/down virtual warehouses
Enhance performance in Snowpark applications	- Materialize results (caching) • Caching DataFrames (using .cache_result()) and understanding why this is useful • Create a temporary table - Vectorization • Understanding the difference between vectorized and scalar UDFs • Vectorized UDFs for batching • Snowpark DataFrames versus pandas on Snowflake - Synchronous versus asynchronous calls • Block parameter
Troubleshoot common errors in Snowpark	- Event tables - Snowpark Python local testing framework - Writing tests (pyTest) - Query history (SQL equivalency to help identify bottlenecks)

Broaden Your Knowledge with Snowflake SPS-C01 Sample Questions:

Question: 1

How can you use a third-party Python library inside a Snowpark UDF?

a) Manually install the package on Snowflake servers
b) Use the Anaconda repository to import the package
c) Run the package installation using pip inside Snowflake
d) Only built-in Python libraries are supported in UDFs

Answer: b

Question: 2

Why are temporary tables useful for Snowpark applications?

a) They improve performance by reducing data re-processing
b) They automatically cache query results indefinitely
c) They can be queried across multiple sessions
d) They require fewer Snowflake credits than standard tables

Answer: a

Question: 3

What is the main difference between scaling up and scaling out a Snowpark warehouse?

a) Scaling up increases warehouse size, while scaling out adds clusters
b) Scaling up adds more nodes, while scaling out increases concurrency
c) Scaling up decreases warehouse credits, while scaling out increases them
d) Scaling up is automatic, while scaling out must be configured manually

Answer: a

Question: 4

A Snowpark Specialist needs to define a Python function to be used as a stored procedure. What should they consider?

a) The function must always return a Snowpark DataFrame.
b) The first parameter for the function must be a Session class object.
c) The @sproc decorator must always be used before the function definition.
d) The pandas DataFrame or pandas Series object can be used as parameters for the function.

Answer: b

Question: 5

A Snowpark Specialist wants to create a Python User-Defined Function (UDF) and operationalize it in Snowflake. The function will not use the IMPORTS clause. What can the Specialist do with this Python UDF in Snowflake?

a) Share the Python UDF directly.
b) Share a view that calls the Python UDF.
c) Access the session object within the Python UDF.
d) Grant the USAGE privilege on the Python UDF to a role.

Answer: d

Question: 6

A Snowpark Specialist developed an application that uses Snowpark for Python to interact with Snowflake tables. Users are reporting constant Multi-Factor Authentication (MFA) alerts. What is the MOST secure method of reducing the MFA requests?

a) Create a NETWORK POLICY for the affected users.
b) Set the account parameter ALLOW_CLIENT_MFA_CACHING to TRUE.
c) Allow users to add a passcode as part of their Snowpark session creation.
d) Disable MFA temporarily for affected users using the parameter DISABLE_MFA.

Answer: b

Question: 7

How can a Snowpark Specialist summarize the sales quantity by product, given a DataFrame containing product sales quantities in columns named product_id and quantity?

a) df.sum("quantity").group_by("product_id")
b) df.summarize("quantity").over("product_id")
c) df.group_by("product_id").agg(sum("quantity"))
d) df.agg("quantity", type="sum").group_by("product_id")

Answer: c

Question: 8

Which method retrieves the first few rows of a Snowpark DataFrame?

a) df.first()
b) df.take()
c) df.show()
d) df.fetch()

Answer: c

Question: 9

Which workload would benefit the MOST from using a Snowpark-optimized virtual warehouse?

a) Machine learning training
b) Machine learning inference
c) Registering a model into the Snowflake Model Registry
d) Creating a compute pool in Snowpark Container Services

Answer: a

Question: 10

What should be done if a Snowpark session fails to connect?

a) Check Snowflake account credentials and network settings
b) Manually increase the session timeout
c) Increase warehouse size to improve connectivity
d) Disable authentication methods

Answer: a

Avail the Study Guide to Pass Snowflake SPS-C01 SnowPro Specialty - Snowpark Exam:

- Find out about the SPS-C01 syllabus topics. Visiting the official site offers an idea about the exam structure and other important study resources. Going through the syllabus topics help to plan the exam in an organized manner.
- Once you are done exploring the **SPS-C01 syllabus**, it is time to plan for studying and covering the syllabus topics from the core. Chalk out the best plan for yourself to cover each part of the syllabus in a hassle-free manner.
- A study schedule helps you to stay calm throughout your exam preparation. It should contain your materials and thoughts like study hours, number of topics for daily studying mentioned on it. The best bet to clear the exam is to follow your schedule rigorously.
- The candidate should not miss out on the scope to learn from the SPS-C01 training. Joining the Snowflake provided training for SPS-C01 exam helps a candidate to strengthen his practical knowledge base from the certification.
- Learning about the probable questions and gaining knowledge regarding the exam structure helps a lot. Go through the **SPS-C01 sample questions** and boost your knowledge
- Make yourself a pro through online practicing the syllabus topics. SPS-C01 practice tests would guide you on your strengths and weaknesses regarding the syllabus topics. Through rigorous practicing, you can improve the weaker sections too. Learn well about time management during exam and become confident gradually with practice tests.

Career Benefits:

- Passing the SPS-C01 exam, helps a candidate to prosper highly in his career. Having the certification on the resume adds to the candidate's benefit and helps to get the best opportunities.