ARA-C01 Valid Exam Discount | New ARA-C01 Braindumps

DOWNLOAD the newest DumpsKing ARA-C01 PDF dumps from Cloud Storage for free: https://drive.google.com/open?id=1NL-D3ONCQdRKk8GS73v7mr-AFpYwPqge

Our specialists check whether the contents of ARA-C01 real exam are updated every day. If there are newer versions, they will be sent to users in time to ensure that users can enjoy the latest resources in the first time. In such a way, our ARA-C01 Guide materials can have such a fast update rate that is taking into account the needs of users. And we will always send our customers with the latest and accurate ARA-C01 exam questions.

Snowflake ARA-C01 (SnowPro Advanced Architect Certification) Certification Exam is a highly reputable certification that is recognized globally by businesses and organizations that use Snowflake. SnowPro Advanced Architect Certification certification exam is designed to test the skills and knowledge of individuals who want to become advanced architects in data warehousing and data analytics.

>> ARA-C01 Valid Exam Discount <<

New ARA-C01 Braindumps & ARA-C01 New Exam Bootcamp

In order to meet the demand of most of the IT employees, DumpsKing's IT experts team use their experience and knowledge to study the past few years Snowflake certification ARA-C01 exam questions. Finally, DumpsKing's latest Snowflake ARA-C01 simulation test, exercise questions and answers have come out. Our Snowflake ARA-C01 simulation test questions have 95% similarity answers with real exam questions and answers, which can help you 100% pass the exam. If you do not pass the exam, DumpsKing will full refund to you. You can also free online download the part of DumpsKing's Snowflake Certification ARA-C01 Exam practice questions and answers as a try. After your understanding of our reliability, I believe you will quickly add DumpsKing's products to your cart. DumpsKing will achieve your dream.

Snowflake SnowPro Advanced Architect Certification Sample Questions (Q39-Q44):

NEW QUESTION # 39

An Architect with the ORGADMIN role wants to change a Snowflake account from an Enterprise edition to a Business Critical edition.
How should this be accomplished?

- A. Run an ALTER ACCOUNT command and create a tag of EDITION and set the tag to Business Critical.
- B. Failover to a new account in the same region and specify the new account's edition upon creation.
- C. Contact Snowflake Support and request that the account's edition be changed.
- D. Use the account's ACCOUNTADMIN role to change the edition.

Answer: C

Explanation:
To change the edition of a Snowflake account, an organization administrator (ORGADMIN) cannot directly alter the account settings through SQL commands or the Snowflake interface. The proper procedure is to contact Snowflake Support to request an edition change for the account. This ensures that the change is managed correctly and aligns with Snowflake's operational protocols.
References: This process is outlined in the Snowflake documentation, which specifies that changes to an account's edition should be facilitated through Snowflake Support1.

NEW QUESTION # 40
A Snowflake Architect is designing a multiple-account design strategy.
This strategy will be MOST cost-effective with which scenarios? (Select TWO).

- A. The company needs to share data between two databases, where one must support Payment Card Industry Data Security Standard (PCI DSS) compliance but the other one does not.
- B. The company needs to support different role-based access control features for the development, test, and production environments.
- C. The company security policy mandates the use of different Active Directory instances for the development, test, and production environments.
- D. The company wants to clone a production database that resides on AWS to a development database that resides on Azure.
- E. The company must use a specific network policy for certain users to allow and block given IP addresses.

Answer: A,B

Explanation:
A multiple-account design strategy is a way of organizing Snowflake accounts into logical groups based on different criteria, such as cloud provider, region, environment, or business unit. A multiple-account design strategy can help achieve various goals, such as cost optimization, performance isolation, security compliance, and data sharing1. In this question, the scenarios that would be most cost-effective with a multiple-account design strategy are:
* The company wants to clone a production database that resides on AWS to a development database that resides on Azure. This scenario would benefit from a multiple-account design strategy because it would allow the company to leverage the cross-cloud replication feature of Snowflake, which enables replicating databases across different cloud platforms and regions. This feature can help reduce the data transfer costs and latency, as well as provide high availability and disaster recovery2.
* The company security policy mandates the use of different Active Directory instances for the development, test, and production environments. This scenario would benefit from a multiple-account design strategy because it would allow the company to use different federated authentication methods for each environment, and integrate them with different Active Directory instances. This can help improve the security and governance of the access to the Snowflake accounts, as well as simplify the user management and provisioning3.
The other scenarios would not be most cost-effective with a multiple-account design strategy, because:
* The company needs to share data between two databases, where one must support Payment Card Industry Data Security Standard (PCI DSS) compliance but the other one does not. This scenario can be handled within a single Snowflake account, by using secure views and secure UDFs to mask or filter the sensitive data, and applying the appropriate roles and privileges to the users who access the data. This can help achieve the PCI DSS compliance without incurring the additional costs of managing multiple accounts4.
* The company needs to support different role-based access control features for the development, test, and production environments. This scenario can also be handled within a single Snowflake account, by using the native role-based access control (RBAC) features of Snowflake, such as roles, grants, and privileges, to define different access levels and permissions for each environment. This can help ensure the security and integrity of the data and the objects, as well as the separation of duties and responsibilities among the users.
* The company must use a specific network policy for certain users to allow and block given IP addresses.

This scenario can also be handled within a single Snowflake account, by using the network policy
* feature of Snowflake, which enables creating and applying network policies to restrict the IP addresses that can access the Snowflake account. This can help prevent unauthorized access and protect the data from malicious attacks.
References:
* Designing Your Snowflake Topology
* Cross-Cloud Replication
* Configuring Federated Authentication and SSO
* Using Secure Views and Secure UDFs to Comply with PCI DSS
* [Understanding Access Control in Snowflake]
* [Network Policies]

NEW QUESTION # 41
A company has a source system that provides JSON records for various loT operations. The JSON Is loading directly into a persistent table with a variant field. The data Is quickly growing to 100s of millions of records and performance to becoming an issue. There is a generic access pattern that Is used to filter on the create_date key within the variant field.
What can be done to improve performance?

- A. Alter the target table to include additional fields pulled from the JSON records. This would include a create_date field with a datatype of varchar. When this field is used in the filter, partition pruning will occur.
- B. Incorporate the use of multiple tables partitioned by date ranges. When a user or process needs to query a particular date range, ensure the appropriate base table Is used.
- C. Alter the target table to Include additional fields pulled from the JSON records. This would Include a create_date field with a datatype of time stamp. When this field Is used in the filter, partition pruning will occur.
- D. Validate the size of the warehouse being used. If the record count is approaching 100s of millions, size XL will be the minimum size required to process this amount of data.

Answer: C

Explanation:
* The correct answer is A because it improves the performance of queries by reducing the amount of data scanned and processed. By adding a create_date field with a timestamp data type, Snowflake can automatically cluster the table based on this field and prune the micro-partitions that do not match the filter condition. This avoids the need to parse the JSON data and access the variant field for every record.
* Option B is incorrect because it does not improve the performance of queries. By adding a create_date field with a varchar data type, Snowflake cannot automatically cluster the table based on this field and prune the micro-partitions that do not match the filter condition. This still requires parsing the JSON data and accessing the variant field for every record.
* Option C is incorrect because it does not address the root cause of the performance issue. By validating the size of the warehouse being used, Snowflake can adjust the compute resources to match the data volume and parallelize the query execution. However, this does not reduce the amount of data scanned and processed, which is the main bottleneck for queries on JSON data.
* Option D is incorrect because it adds unnecessary complexity and overhead to the data loading and querying process. By incorporating the use of multiple tables partitioned by date ranges, Snowflake can reduce the amount of data scanned and processed for queries that specify a date range. However, this requires creating and maintaining multiple tables, loading data into the appropriate table based on the date, and joining the tables for queries that span multiple date ranges. References:
* Snowflake Documentation: Loading Data Using Snowpipe: This document explains how to use Snowpipe to continuously load data from external sources into Snowflake tables. It also describes the syntax and usage of the COPY INTO command, which supports various options and parameters to control the loading behavior, such as ON_ERROR, PURGE, and SKIP_FILE.
* Snowflake Documentation: Date and Time Data Types and Functions: This document explains the different data types and functions for working with date and time values in Snowflake. It also describes how to set and change the session timezone and the system timezone.
* Snowflake Documentation: Querying Metadata: This document explains how to query the metadata of the objects and operations in Snowflake using various functions, views, and tables. It also describes how to access the copy history information using the COPY_HISTORY function or the COPY_HISTORY view.
* Snowflake Documentation: Loading JSON Data: This document explains how to load JSON data into Snowflake tables using various methods, such as the COPY INTO command, the INSERT command, or the PUT command. It also describes how to access and query JSON data using the dot notation, the FLATTEN function, or the LATERAL join.
* Snowflake Documentation: Optimizing Storage for Performance: This document explains how to
* optimize the storage of data in Snowflake tables to improve the performance of queries. It also describes the concepts and benefits of automatic clustering, search optimization service, and materialized views.

NEW QUESTION # 42
You are a snowflake architect in an organization. The business team came to to deploy an use case which requires you to load some data which they can visualize through tableau. Everyday new data comes in and the old data is no longer required.
What type of table you will use in this case to optimize cost

- A. TEMPORARY
- B. TRANSIENT
- C. PERMANENT

Answer: B

NEW QUESTION # 43
A retail company has over 3000 stores all using the same Point of Sale (POS) system. The company wants to deliver near real-time sales results to category managers. The stores operate in a variety of time zones and exhibit a dynamic range of transactions each minute, with some stores having higher sales volumes than others.
Sales results are provided in a uniform fashion using data engineered fields that will be calculated in a complex data pipeline. Calculations include exceptions, aggregations, and scoring using external functions interfaced to scoring algorithms. The source data for aggregations has over 100M rows.
Every minute, the POS sends all sales transactions files to a cloud storage location with a naming convention that includes store numbers and timestamps to identify the set of transactions contained in the files. The files are typically less than 10MB in size.
How can the near real-time results be provided to the category managers? (Select TWO).

- A. An external scheduler should examine the contents of the cloud storage location and issue SnowSQL commands to process the data at a frequency that matches the real-time analytics needs.
- B. A stream should be created to accumulate the near real-time data and a task should be created that runs at a frequency that matches the real-time analytics needs.
- C. A Snowpipe should be created and configured with AUTO_INGEST = true. A stream should be created to process INSERTS into a single target table using the stream metadata to inform the store number and timestamps.
- D. The copy into command with a task scheduled to run every second should be used to achieve the near- real time requirement.
- E. All files should be concatenated before ingestion into Snowflake to avoid micro-ingestion.

Answer: B,C

Explanation:
To provide near real-time sales results to category managers, the Architect can use the following steps:
* Create an external stage that references the cloud storage location where the POS sends the sales transactions files. The external stage should use the file format and encryption settings that match the source files2
* Create a Snowpipe that loads the files from the external stage into a target table in Snowflake. The Snowpipe should be configured with AUTO_INGEST = true, which means that it will automatically detect and ingest new files as they arrive in the external stage. The Snowpipe should also use a copy option to purge the files from the external stage after loading, to avoid duplicate ingestion3
* Create a stream on the target table that captures the INSERTS made by the Snowpipe. The stream should include the metadata columns that provide information about the file name, path, size, and last modified time. The stream should also have a retention period that matches the real-time analytics needs4
* Create a task that runs a query on the stream to process the near real-time data. The query should use the stream metadata to extract the store number and timestamps from the file name and path, and perform the calculations for exceptions, aggregations, and scoring using external functions. The query should also output the results to another table or view that can be accessed by the category managers.
The task should be scheduled to run at a frequency that matches the real-time analytics needs, such as every minute or every 5 minutes.
The other options are not optimal or feasible for providing near real-time results:
* All files should be concatenated before ingestion into Snowflake to avoid micro-ingestion. This option is not recommended because it would introduce additional latency and complexity in the data pipeline.
Concatenating files would require an external process or service that monitors the cloud storage location and performs the file merging operation. This would delay the ingestion of new files into Snowflake and increase the risk of data loss or corruption. Moreover, concatenating files would not avoid micro-ingestion, as Snowpipe would still ingest each concatenated file as a separate load.
* An external scheduler should examine the contents of the cloud storage location and issue SnowSQL commands to process the data at a frequency that matches the real-time analytics needs. This option is not necessary because Snowpipe can automatically ingest new files from the external stage without requiring an external trigger or scheduler. Using an external scheduler would add

more overhead and dependency to the data pipeline, and it would not guarantee near real-time ingestion, as it would depend on the polling interval and the availability of the external scheduler.
* The copy into command with a task scheduled to run every second should be used to achieve the near- real time requirement. This option is not feasible because tasks cannot be scheduled to run every second in Snowflake. The minimum interval for tasks is one minute, and even that is not guaranteed, as tasks are subject to scheduling delays and concurrency limits. Moreover, using the copy into command with a task would not leverage the benefits of Snowpipe, such as automatic file detection, load balancing, and micro-partition optimization. References:
* 1: SnowPro Advanced: Architect | Study Guide
* 2: Snowflake Documentation | Creating Stages
* 3: Snowflake Documentation | Loading Data Using Snowpipe
* 4: Snowflake Documentation | Using Streams and Tasks for ELT
* : Snowflake Documentation | Creating Tasks
* : Snowflake Documentation | Best Practices for Loading Data
* : Snowflake Documentation | Using the Snowpipe REST API
* : Snowflake Documentation | Scheduling Tasks
* : SnowPro Advanced: Architect | Study Guide
* : Creating Stages
* : Loading Data Using Snowpipe
* : Using Streams and Tasks for ELT
* : [Creating Tasks]
* : [Best Practices for Loading Data]
* : [Using the Snowpipe REST API]
* : [Scheduling Tasks]

NEW QUESTION # 44
.....

www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.4shared.com, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, Disposable vapes

P.S. Free & New ARA-C01 dumps are available on Google Drive shared by DumpsKing: https://drive.google.com/open?id=1NL-D3ONCQdRKk8GS73v7mr-AFpYwPqge