Google Cloud

Partner Certification Academy

Professional Cloud Developer

v2309

Quiz questions*

Cloud Firestore in Native mode and Datastore mode

** These are for practice only and are not actual exam questions*

Question: What type of database is Cloud Firestore?

- A. SQL database with tables and rows.
- B. NoSQL, document-oriented database.
- C. Relational database with entities and relationships.
- D. Graph database with nodes and edges.

Question: What is one of the main benefits of using Firestore for application developers?

- A. It requires developers to manage server infrastructure.
- B. It focuses primarily on database management.
- C. It allows developers to focus primarily on their business logic and user experience.

- D. It lacks integration with the Firebase ecosystem.

Question: Which of the following describes Firestore in Native mode?

- A. Firestore in Native mode uses Datastore system behavior but accesses Firestore's storage layer.
- B. Firestore in Native mode is the next major version of Datastore and introduces features such as a new, strongly consistent storage layer, real-time updates, and mobile and web client libraries.
- C. Firestore in Native mode will accept Datastore API requests and deny Firestore API requests.
- D. Firestore in Native mode uses the Datastore viewer in the Google Cloud console.

Question: When is it recommended to use Firestore in Datastore mode?

- A. For new mobile and web apps.
- B. For projects that require real-time updates.
- C. For new server projects.
- D. For projects that need Firestore client libraries.

Question: Which mode of Firestore supports the ability to listen to a document or a set of documents for real-time updates?

- A. Firestore in Datastore mode
- B. Firestore in Native mode
- C. Both Firestore in Native mode and Datastore mode
- D. Neither Firestore in Native mode nor Datastore mode

Question: Which of the following features is disabled in Firestore when using Datastore mode?

- A. Strongly consistent queries across the entire database.
- B. Access to Firestore's new storage layer.
- C. Firestore real-time capabilities.
- D. Ability to scale to millions of writes per second.

Question: Which API requests are denied in Firestore when using Datastore mode?

- A. Firestore v1 API requests.
- B. Datastore v1 API requests.
- C. Both Firestore v1 and Datastore v1 API requests.
- D. Neither Firestore v1 nor Datastore v1 API requests.

Question: Which mode of Firestore organizes data into documents and collections as part of its data model?

- A. Firestore in Datastore mode
- B. Firestore in Native mode
- C. Both Firestore in Native mode and Datastore mode
- D. Neither Firestore in Native mode nor Datastore mode

Question: In Cloud Firestore in native mode, what is the unit of storage?
- A. Collection
- B. Key-value pair
- C. Document
- D. Table

Question: Which statement is true about collections in Cloud Firestore?
- A. Collections can directly contain raw fields with values.
- B. Collections can contain other collections.
- C. The names of documents within a collection are unique.
- D. You need to explicitly create and delete collections.

Question: How can you structure data hierarchically in Cloud Firestore?

- A. By referencing a collection in a collection.
- B. By referencing a document in a document.
- C. By using subcollections.
- D. By using multiple databases.

Question: Which feature of Cloud Firestore ensures that users can query their purchase immediately after submission, even in offline conditions?

- A. Realtime updates
- B. Expressive querying
- C. Offline support
- D. Designed to scale

Question: In Cloud Firestore, how can you efficiently sort purchases on a variety of fields?

- A. Using subcollections
- B. Using Realtime updates
- C. Using Expressive querying
- D. Using Offline support

Question: How does Cloud Firestore ensure that data on any connected device is updated?

- A. By using subcollections
- B. By using Realtime listeners
- C. By using Expressive querying
- D. By using Offline support

Question: In Cloud Firestore, what is a transaction?

- A. A set of read operations on one or more documents.
- B. A set of write operations on one or more documents.

- C. A set of read and write operations on one or more documents.
- D. A set of batched write operations on multiple documents.

Question: When using Cloud Firestore transactions, which of the following statements is true?

- A. Write operations must come before read operations.
- B. Read operations must come before write operations.
- C. Transactions can partially apply writes.
- D. Transactions can be used even when the client is offline.

Question: What happens if a concurrent edit affects a document that a transaction reads in Cloud Firestore?

- A. The transaction is immediately terminated.
- B. The transaction is paused until the edit is complete.
- C. Cloud Firestore runs the entire transaction again.
- D. The transaction continues with outdated data.

Question: In a Cloud Firestore transaction, when are all write operations executed?

- A. At the beginning of a successful transaction.
- B. In the middle of a successful transaction.
- C. At the end of a successful transaction.
- D. Write operations are executed immediately.

Correct Answer: C. At the end of a successful transaction.

Question: In Cloud Datastore, what is the guarantee provided by a transaction?

- A. Transactions are partially applied.
- B. Transactions are atomic.
- C. Transactions can be applied multiple times.
- D. Transactions are optional.

Question: What happens if a transaction in Cloud Datastore exceeds a resource limit?

- A. The transaction is automatically retried.
- B. The transaction is paused.
- C. The Datastore API returns a success message.
- D. The Datastore API returns an error.

Question: In Cloud Datastore, what is the default concurrency mode?

- A. Optimistic
- B. Pessimistic
- C. Optimistic With Entity Groups
- D. None of the above

Question: Which mode is recommended for new mobile and web apps?

- A. Firestore in Datastore mode.
- B. Firestore in Native mode.
- C. Both modes are equally recommended.
- D. Neither mode is recommended.

Question: Which of the following is NOT a recommended best practice for Firestore document IDs?

- A. Avoiding the use of forward slashes in document IDs.
- B. Using monotonically increasing document IDs such as Customer1, Customer2, Customer3.
- C. Avoiding the use of special characters in document IDs.
- D. Using Firestore's automatic document ID allocation.

Question: What is the potential impact of using too many indexes in Firestore?

- A. Decreased read latency.
- B. Reduced storage costs.
- C. Increased write latency.
- D. Improved query performance.

Question: You are building an application to host multiple blogs using Cloud Firestore. Each blog will have users posting comments. How should you optimally structure your collections in Firestore to store comments for each blog?

- A. Create a single collection named "Blogs" and store comments for each blog as an array field within each blog document.
- B. Create a single collection named "Comments" and use a reference field in each comment document to link it to a specific blog.
- C. Create a collection named "Blogs" and for each blog document, create a sub-collection named "Comments" to store comments related to that blog.
- D. Store both blogs and comments as individual documents in a single collection, using a map field to associate comments with blogs.

Question: Why should you avoid using offsets in Firestore queries?

- A. Offsets increase the cost of queries.
- B. Using an offset only avoids returning the skipped documents to your application, but these documents are still retrieved internally.
- C. Offsets improve the performance of queries.
- D. Offsets reduce the number of documents returned by a query.

Question: Which of the following can lead to hotspotting in Firestore?

- A. Creating new documents at a very high rate and allocating its own monotonically increasing IDs.
- B. Reading from a collection with a large number of documents.
- C. Using Firestore's automatic document ID allocation.

- D. Writing to the database at a moderate rate.

Question: Which characters should always be used for namespace names, kind names, property names, and custom key names in Firestore in Datastore mode?

- A. Non-UTF-8 characters.
- B. UTF-8 characters.
- C. Special characters.
- D. Forward slashes.

Question: What is the recommended approach for performing multiple read, write, or delete operations in Firestore in Datastore mode?

- A. Use single operations for each read, write, or delete.
- B. Use batch operations for reads, writes, and deletes.
- C. Use asynchronous calls for reads and synchronous calls for writes.
- D. Use synchronous calls for both reads and writes.

Question: What should you avoid when using numeric IDs for keys in Firestore in Datastore mode?

- A. Using positive numbers for the ID.
- B. Using the value 0 (zero) for the ID.
- C. Using a random number for the ID.
- D. Using a hash value for the ID.

Question: What is the purpose of using "sharded queries" in Firestore in Datastore mode?

- A. To reduce storage costs.
- B. To improve the performance of timestamp-based queries.
- C. To automatically restore deleted entities.

- D. To increase the write latency.

Question: Why should you avoid using too many indexes in Firestore?

- A. Indexes improve write latency.
- B. Indexes reduce storage costs.
- C. An excessive number of indexes can increase write latency.
- D. Indexes have no impact on storage costs.

Question: When adding a comment's key to an array property on a user profile in Firestore, which method should you avoid to ensure data integrity?

- A. Using arrayUnion() method to ensure uniqueness.
- B. Directly setting the array with a new value.
- C. Using arrayRemove() method to remove specific values.
- D. Querying the database to get the current array, modifying it, and then saving it back.

Question: In Firestore, if you want to listen to real-time updates for a specific document, which method would you use?

- A. get()
- B. addSnapshotListener()
- C. retrieve()
- D. listen()

Question: In Firestore, what ensures that a transaction runs on up-to-date and consistent data?

- A. Transactions always apply writes partially.
- B. Transactions run only once regardless of concurrent edits.

- C. Transactions run the entire transaction again in the case of a concurrent edit.
- D. Transactions allow write operations to come before read operations.

Question: Which of the following statements about Firestore transactions is true?

- A. Transactions can have write operations followed by read operations.
- B. Transactions can directly modify application state.
- C. Transactions will succeed even when the client is offline.
- D. All writes execute at the end of a successful transaction.

Question: What happens if a function calling a Firestore transaction reads a document that is affected by a concurrent edit?

- A. The transaction function runs only once.
- B. The transaction function might run more than once.
- C. The transaction function modifies the application state directly.
- D. The transaction function will fail and not retry.

Question: In Firestore, which of the following is true regarding the relationship between collections and documents?

- A. Collections can directly contain other collections.
- B. Documents can exist without being part of any collection.
- C. Collections can contain documents and other raw fields with values.
- D. The names of documents within a collection are unique.

Question: In Firestore, how would you create a reference to a specific comment within a specific article's comments subcollection?

- A. Use the path 'articles/articleID/comments/commentID'.
- B. Use the path 'articles.articleID.comments.commentID'.

- C. Directly reference the commentID since Firestore ensures global uniqueness.
- D. Use the path 'comments/commentID' and store the articleID within the comment document.

Answers to Quiz questions

Cloud Firestore

Question: What type of database is Cloud Firestore?

- A. SQL database with tables and rows.
- B. NoSQL, document-oriented database.
- C. Relational database with entities and relationships.
- D. Graph database with nodes and edges.

Correct Answer: B. NoSQL, document-oriented database.

Explanation: Cloud Firestore is a NoSQL, document-oriented database. Unlike a SQL database, it doesn't have tables or rows. Instead, data is stored in documents, which are organized into collections.
Resource: Cloud Firestore Data model | Firebase

Question: What is one of the main benefits of using Firestore for application developers?

- A. It requires developers to manage server infrastructure.
- B. It focuses primarily on database management.
- C. It allows developers to focus primarily on their business logic and user experience.
- D. It lacks integration with the Firebase ecosystem.

Correct Answer: C. It allows developers to focus primarily on their business logic and user experience.

Explanation: Firestore, along with the Firebase ecosystem, greatly simplifies common app development challenges, allowing developers to focus more on their business logic and user experience. Resource: Firestore documentation

Question: Which of the following describes Firestore in Native mode?

- A. Firestore in Native mode uses Datastore system behavior but accesses Firestore's storage layer.
- B. Firestore in Native mode is the next major version of Datastore and introduces features such as a new, strongly consistent storage layer, real-time updates, and mobile and web client libraries.
- C. Firestore in Native mode will accept Datastore API requests and deny Firestore API requests.
- D. Firestore in Native mode uses the Datastore viewer in the Google Cloud console.

Correct Answer: B. Firestore in Native mode is the next major version of Datastore and introduces features such as a new, strongly consistent storage layer, real-time updates, and mobile and web client libraries.

Explanation: Firestore in Native mode is designed as the successor to Datastore, combining the best features of Datastore and Firebase Realtime Database. It introduces a new data model, real-time updates, and client libraries for mobile and web, which are not available in Datastore mode.
Resource: Choosing between Native mode and Datastore mode | Firestore | Google Cloud

Question: When is it recommended to use Firestore in Datastore mode?

- A. For new mobile and web apps.
- B. For projects that require real-time updates.
- C. For new server projects.
- D. For projects that need Firestore client libraries.

Correct Answer: C. For new server projects.

Explanation: Firestore in Datastore mode is recommended for new server projects as it allows the use of established Datastore server architectures while removing some of the fundamental limitations of Datastore.
Resource: [Choosing between Native mode and Datastore mode | Firestore | Google Cloud]

Question: Which mode of Firestore supports the ability to listen to a document or a set of documents for real-time updates?

- A. Firestore in Datastore mode
- B. Firestore in Native mode
- C. Both Firestore in Native mode and Datastore mode
- D. Neither Firestore in Native mode nor Datastore mode

Correct Answer: B. Firestore in Native mode

Explanation: Firestore in Native mode supports the ability to listen to a document or a set of documents for real-time updates, allowing clients to be notified of any data changes and receive the newest set of data.
Resource: [Choosing between Native mode and Datastore mode | Firestore | Google Cloud]

Question: Which of the following features is disabled in Firestore when using Datastore mode?

- A. Strongly consistent queries across the entire database.
- B. Access to Firestore's new storage layer.
- C. Firestore real-time capabilities.
- D. Ability to scale to millions of writes per second.

Correct Answer: C. Firestore real-time capabilities.

Explanation: In Datastore mode, Firestore's real-time capabilities are not available. This means that real-time updates and notifications of data changes are not supported in this mode.
Resource: Choosing between Native mode and Datastore mode | Firestore | Google Cloud

Question: Which API requests are denied in Firestore when using Datastore mode?

- A. Firestore v1 API requests.
- B. Datastore v1 API requests.
- C. Both Firestore v1 and Datastore v1 API requests.
- D. Neither Firestore v1 nor Datastore v1 API requests.

Correct Answer: A. Firestore v1 API requests.

Explanation: In Datastore mode, the project will accept Datastore API requests but will deny Firestore API requests. This is because Datastore mode is designed to be compatible with Datastore and not with the new features of Firestore.
Resource: Choosing between Native mode and Datastore mode | Firestore | Google Cloud

Question: Which mode of Firestore organizes data into documents and collections as part of its data model?

- A. Firestore in Datastore mode
- B. Firestore in Native mode
- C. Both Firestore in Native mode and Datastore mode
- D. Neither Firestore in Native mode nor Datastore mode

Correct Answer: B. Firestore in Native mode

Explanation: Firestore in Native mode uses a data model that organizes data into documents and collections, providing a more flexible and intuitive way to structure data.
Resource: [Choosing between Native mode and Datastore mode | Firestore | Google Cloud]

Question: In Cloud Firestore in native mode, what is the unit of storage?

- A. Collection
- B. Key-value pair
- C. Document
- D. Table

Correct Answer: C. Document

Explanation: In Cloud Firestore, the unit of storage is the document. A document is a lightweight record that contains fields, which map to values.
Resource: [Cloud Firestore Data model | Firebase]

Question: Which statement is true about collections in Cloud Firestore?

- A. Collections can directly contain raw fields with values.
- B. Collections can contain other collections.
- C. The names of documents within a collection are unique.
- D. You need to explicitly create and delete collections.

Correct Answer: C. The names of documents within a collection are unique.

Explanation: The names of documents within a collection are unique. Collections contain documents and nothing else. They can't directly contain raw fields with values, and they can't contain other collections.
Resource: [Cloud Firestore Data model | Firebase]

Question: How can you structure data hierarchically in Cloud Firestore?

- A. By referencing a collection in a collection.
- B. By referencing a document in a document.
- C. By using subcollections.
- D. By using multiple databases.

Correct Answer: C. By using subcollections.

Explanation: Subcollections allow you to structure data hierarchically in Cloud Firestore. This makes data easier to access. You can create a subcollection associated with a specific document.
Resource: Cloud Firestore Data model | Firebase

Question: Which feature of Cloud Firestore ensures that users can query their purchase immediately after submission, even in offline conditions?

- A. Realtime updates
- B. Expressive querying
- C. Offline support
- D. Designed to scale

Correct Answer: C. Offline support

Explanation: Cloud Firestore caches data that your app is actively using, allowing the app to write, read, listen to, and query data even if the device is offline. When the device regains connectivity, Cloud Firestore synchronizes any local changes back to the database.
Resource: Cloud Firestore | Firebase

Question: In Cloud Firestore, how can you efficiently sort purchases on a variety of fields?

- A. Using subcollections
- B. Using Realtime updates
- C. Using Expressive querying
- D. Using Offline support

Correct Answer: C. Using Expressive querying

Explanation: In Cloud Firestore, you can use expressive querying to retrieve specific documents or all the documents in a collection that match your query parameters. Your queries can include multiple, chained filters and combine filtering and sorting.

Resource: [Cloud Firestore | Firebase]

Question: How does Cloud Firestore ensure that data on any connected device is updated?

- A. By using subcollections
- B. By using Realtime listeners
- C. By using Expressive querying
- D. By using Offline support

Correct Answer: B. By using Realtime listeners

Explanation: Like Realtime Database, Cloud Firestore uses data synchronization to update data on any connected device.
Resource: [Cloud Firestore | Firebase]

Question: In Cloud Firestore, what is a transaction?

- A. A set of read operations on one or more documents.
- B. A set of write operations on one or more documents.
- C. A set of read and write operations on one or more documents.
- D. A set of batched write operations on multiple documents.

Correct Answer: C. A set of read and write operations on one or more documents.

Explanation: A transaction in Cloud Firestore is a set of read and write operations on one or more documents. It ensures that either all of the operations succeed, or none of them are applied.
Resource: [Transactions and batched writes | Firestore | Firebase]

Question: When using Cloud Firestore transactions, which of the following statements is true?

- A. Write operations must come before read operations.
- B. Read operations must come before write operations.
- C. Transactions can partially apply writes.
- D. Transactions can be used even when the client is offline.

Correct Answer: B. Read operations must come before write operations.

Explanation: In Cloud Firestore transactions, read operations must be performed before write operations.
Resource: [Transactions and batched writes | Firestore | Firebase]()

Question: What happens if a concurrent edit affects a document that a transaction reads in Cloud Firestore?

- A. The transaction is immediately terminated.
- B. The transaction is paused until the edit is complete.
- C. Cloud Firestore runs the entire transaction again.
- D. The transaction continues with outdated data.

Correct Answer: C. Cloud Firestore runs the entire transaction again.

Explanation: If a concurrent edit affects a document that a transaction reads, Cloud Firestore retries the transaction to ensure it runs on up-to-date and consistent data.
Resource: [Transactions and batched writes | Firestore | Firebase]()

Question: In a Cloud Firestore transaction, when are all write operations executed?

- A. At the beginning of a successful transaction.
- B. In the middle of a successful transaction.
- C. At the end of a successful transaction.
- D. Write operations are executed immediately.

Correct Answer: C. At the end of a successful transaction.

Explanation: Transactions in Cloud Firestore never partially apply writes. All writes are executed at the end of a successful transaction.
Resource: [Transactions and batched writes | Firestore | Firebase]()

Question: In Cloud Datastore, what is the guarantee provided by a transaction?

- A. Transactions are partially applied.
- B. Transactions are atomic.
- C. Transactions can be applied multiple times.
- D. Transactions are optional.

Correct Answer: B. Transactions are atomic.

Explanation: In Cloud Datastore, a transaction is a set of operations on one or more entities. Each transaction is guaranteed to be atomic, meaning that transactions are never partially applied. Either all of the operations in the transaction are applied, or none of them are applied.
Resource: [Transactions | Cloud Datastore Documentation | Google Cloud]

Question: What happens if a transaction in Cloud Datastore exceeds a resource limit?

- A. The transaction is automatically retried.
- B. The transaction is paused.
- C. The Datastore API returns a success message.
- D. The Datastore API returns an error.

Correct Answer: D. The Datastore API returns an error.

Explanation: An operation may fail when too many concurrent modifications are attempted on the same entity, the transaction exceeds a resource limit, or the Datastore mode database encounters an internal error. In all these cases, the Datastore API returns an error.
Resource: [Transactions | Cloud Datastore Documentation | Google Cloud]

Question: In Cloud Datastore, what is the default concurrency mode?

- A. Optimistic
- B. Pessimistic
- C. Optimistic With Entity Groups
- D. None of the above

Correct Answer: B. Pessimistic

Explanation: Firestore in Datastore mode supports three concurrency modes: Pessimistic, Optimistic, and Optimistic With Entity Groups. The default concurrency mode is Pessimistic.
Resource: Transactions | Cloud Datastore Documentation | Google Cloud

Question: Which mode is recommended for new mobile and web apps?

- A. Firestore in Datastore mode.
- B. Firestore in Native mode.
- C. Both modes are equally recommended.
- D. Neither mode is recommended.

Correct Answer: B. Firestore in Native mode.

Explanation: Firestore in Native mode is recommended for new mobile and web apps because it offers mobile and web client libraries with real-time and offline features.
Resource: Choosing a database mode

Question: Which of the following is NOT a recommended best practice for Firestore document IDs?

- A. Avoiding the use of forward slashes in document IDs.
- B. Using monotonically increasing document IDs such as Customer1, Customer2, Customer3.
- C. Avoiding the use of special characters in document IDs.
- D. Using Firestore's automatic document ID allocation.

Correct Answer: B. Using monotonically increasing document IDs such as Customer1, Customer2, Customer3.

Explanation: Using monotonically increasing document IDs can lead to hotspots that impact latency. It's recommended to avoid such sequential IDs.
Resource: Firestore Best Practices

Question: What is the potential impact of using too many indexes in Firestore?

- A. Decreased read latency.
- B. Reduced storage costs.
- C. Increased write latency.
- D. Improved query performance.

Correct Answer: C. Increased write latency.

Explanation: An excessive number of indexes can increase write latency and also increase storage costs for index entries.
Resource: [Firestore Best Practices]()

Question: Why should you avoid using offsets in Firestore queries?

- A. Offsets increase the cost of queries.
- B. Using an offset only avoids returning the skipped documents to your application, but these documents are still retrieved internally.
- C. Offsets improve the performance of queries.
- D. Offsets reduce the number of documents returned by a query.

Correct Answer: B. Using an offset only avoids returning the skipped documents to your application, but these documents are still retrieved internally.

Explanation: Using an offset only avoids returning the skipped documents, but they are still retrieved internally. This affects the latency of the query, and you are billed for the read operations required to retrieve them.
Resource: [Firestore Best Practices]()

Question: Which of the following can lead to hotspotting in Firestore?

- A. Creating new documents at a very high rate and allocating its own monotonically increasing IDs.
- B. Reading from a collection with a large number of documents.

- C. Using Firestore's automatic document ID allocation.
- D. Writing to the database at a moderate rate.

Correct Answer: A. Creating new documents at a very high rate and allocating its own monotonically increasing IDs.

Explanation: Hotspotting can occur if you create new documents at a very high rate and allocate your own monotonically increasing IDs. Firestore allocates document IDs using a scatter algorithm, so using automatic document IDs should not lead to hotspotting.
Resource: Firestore Best Practices

Question: Which characters should always be used for namespace names, kind names, property names, and custom key names in Firestore in Datastore mode?

- A. Non-UTF-8 characters.
- B. UTF-8 characters.
- C. Special characters.
- D. Forward slashes.

Correct Answer: B. UTF-8 characters.

Explanation: Always use UTF-8 characters for namespace names, kind names, property names, and custom key names. Non-UTF-8 characters used in these names can interfere with Datastore mode functionality.
Resource: Firestore in Datastore mode Best Practices

Question: What is the recommended approach for performing multiple read, write, or delete operations in Firestore in Datastore mode?

- A. Use single operations for each read, write, or delete.
- B. Use batch operations for reads, writes, and deletes.
- C. Use asynchronous calls for reads and synchronous calls for writes.
- D. Use synchronous calls for both reads and writes.

Correct Answer: B. Use batch operations for reads, writes, and deletes.

Explanation: Batch operations are more efficient because they perform multiple operations with the same overhead as a single operation.
Resource: Firestore in Datastore mode Best Practices

Question: What should you avoid when using numeric IDs for keys in Firestore in Datastore mode?

- A. Using positive numbers for the ID.
- B. Using the value 0 (zero) for the ID.
- C. Using a random number for the ID.
- D. Using a hash value for the ID.

Correct Answer: B. Using the value 0 (zero) for the ID.

Explanation: Do not use the value 0 (zero) for the ID. If you do, you will get an automatically allocated ID.
Resource: Firestore in Datastore mode Best Practices

Question: What is the purpose of using "sharded queries" in Firestore in Datastore mode?

- A. To reduce storage costs.
- B. To improve the performance of timestamp-based queries.
- C. To automatically restore deleted entities.
- D. To increase the write latency.

Correct Answer: B. To improve the performance of timestamp-based queries.

Explanation: "Sharded queries" prepend a fixed-length string to the expiration timestamp. The index is sorted on the full string, so entities with the same timestamp will be located throughout the key range of the index. This improves the performance of timestamp-based queries.
Resource: Firestore in Datastore mode Best Practices

Question: Why should you avoid using too many indexes in Firestore?

- A. Indexes improve write latency.
- B. Indexes reduce storage costs.
- C. An excessive number of indexes can increase write latency.
- D. Indexes have no impact on storage costs.

Correct Answer: C. An excessive number of indexes can increase write latency.

Explanation: Having too many indexes can lead to increased write latency and also increase storage costs for index entries.
Resource: Firestore Best Practices

Question: You are building an application to host multiple blogs using Cloud Firestore. Each blog will have users posting comments. How should you optimally structure your collections in Firestore to store comments for each blog?

- A. Create a single collection named "Blogs" and store comments for each blog as an array field within each blog document.
- B. Create a single collection named "Comments" and use a reference field in each comment document to link it to a specific blog.
- C. Create a collection named "Blogs" and for each blog document, create a sub-collection named "Comments" to store comments related to that blog.
- D. Store both blogs and comments as individual documents in a single collection, using a map field to associate comments with blogs.

Correct Answer: C. Create a collection named "Blogs" and for each blog document, create a sub-collection named "Comments" to store comments related to that blog.

Explanation: In Firestore, Using sub-collections for comments under each blog document allows for efficient querying and scalability. It keeps the data structure

organized and ensures that each blog and its comments are closely associated in a hierarchical manner.
Resource: Firestore Data Model

Question: When adding a comment's key to an array property on a user profile in Firestore, which method should you avoid to ensure data integrity?

- A. Using arrayUnion() method to ensure uniqueness.
- B. Directly setting the array with a new value.
- C. Using arrayRemove() method to remove specific values.
- D. Querying the database to get the current array, modifying it, and then saving it back.

Correct Answer: D. Querying the database to get the current array, modifying it, and then saving it back.

Explanation: Directly querying and then modifying the array can lead to race conditions where data might be overwritten by concurrent operations. Firestore provides arrayUnion() and arrayRemove() methods to safely add or remove items from an array without needing to read the current value.
Resource: Firestore Arrays

Question: In Firestore, if you want to listen to real-time updates for a specific document, which method would you use?

- A. get()
- B. addSnapshotListener()
- C. retrieve()
- D. listen()

Correct Answer: B. addSnapshotListener()

Explanation: The addSnapshotListener() method in Firestore allows you to listen to a document and receive real-time updates whenever the document changes.

This is useful for building reactive applications where you want to reflect changes in the database immediately in the UI.
Resource: Get Realtime Updates with Firestore

Question: In Firestore, what ensures that a transaction runs on up-to-date and consistent data?

- A. Transactions always apply writes partially.
- B. Transactions run only once regardless of concurrent edits.
- C. Transactions run the entire transaction again in the case of a concurrent edit.
- D. Transactions allow write operations to come before read operations.

Correct Answer: C. Transactions run the entire transaction again in the case of a concurrent edit.

Explanation: In Firestore, if a transaction reads documents and another client modifies any of those documents, Firestore retries the transaction. This ensures that the transaction runs on up-to-date and consistent data.
Resource: Transactions and batched writes | Firestore | Google Cloud

Question: Which of the following statements about Firestore transactions is true?

- A. Transactions can have write operations followed by read operations.
- B. Transactions can directly modify application state.
- C. Transactions will succeed even when the client is offline.
- D. All writes execute at the end of a successful transaction.

Correct Answer: D. All writes execute at the end of a successful transaction.

Explanation: In Firestore, transactions never partially apply writes. This means that all write operations within a transaction are executed at the end, ensuring atomicity.
Resource: Transactions and batched writes | Firestore | Google Cloud

Question: What happens if a function calling a Firestore transaction reads a document that is affected by a concurrent edit?

- A. The transaction function runs only once.
- B. The transaction function might run more than once.
- C. The transaction function modifies the application state directly.
- D. The transaction function will fail and not retry.

Correct Answer: B. The transaction function might run more than once.

Explanation: If a function calling a Firestore transaction reads a document that is affected by a concurrent
edit, the transaction function might run more than once to ensure that the transaction runs on up-to-date and consistent data.
Resource: Transactions and batched writes | Firestore | Google Cloud

Question: In Firestore, which of the following is true regarding the relationship between collections and documents?

- A. Collections can directly contain other collections.
- B. Documents can exist without being part of any collection.
- C. Collections can contain documents and other raw fields with values.
- D. The names of documents within a collection are unique.

Correct Answer: D. The names of documents within a collection are unique.

Explanation: In Firestore, the names of documents within a collection are unique. You can provide your own keys, such as user IDs, or you can let Firestore create random IDs for you automatically.
Resource: Data model | Firestore | Google Cloud

Question: In Firestore, how would you create a reference to a specific comment within a specific article's comments subcollection?

- A. Use the path 'articles/articleID/comments/commentID'.
- B. Use the path 'articles.articleID.comments.commentID'.
- C. Directly reference the commentID since Firestore ensures global uniqueness.
- D. Use the path 'comments/commentID' and store the articleID within the comment document.

Correct Answer: A. Use the path 'articles/articleID/comments/commentID'.

Explanation: In Firestore, you can create references by specifying the path to a document or collection as a string, with path components separated by a forward slash (/). To reference a specific comment within an article's comments subcollection, you would use the path 'articles/articleID/comments/commentID'.
Resource: Data model | Firestore | Google Cloud