[Q12-Q34 C100DEV Certification - The Ultimate Guide [Updated 2023

C100DEV Certification - The Ultimate Guide [Updated 2023]
C100DEV Practice Exam and Study Guides - Verified By BraindumpsIT

MongoDB C100DEV Exam is a computer-based test that consists of multiple-choice questions. C100DEV exam is timed, and candidates are given a maximum of 90 minutes to complete it. C100DEV exam is available in English and can be taken at any of the Pearson VUE testing centers worldwide. Candidates who pass the exam will earn the MongoDB Certified Developer Associate Certification and a digital badge that can be displayed on their resume or LinkedIn profile.

MongoDB C100DEV certification is an industry-recognized certification that validates the skills and knowledge of developers who work with MongoDB. It is designed for developers who want to demonstrate their proficiency in building applications using MongoDB. MongoDB Certified Developer Associate Exam certification exam covers a wide range of topics, including data modeling, indexing, aggregation, and application development with MongoDB.

Q12. Select all true statements regarding to Aggregation Framework.
* The Aggregation Framework will automatically reorder stages in a certain conditions (for optimization).

* The query in a $match stage can not be entirely covered by an index.
https://docs.mongodb.com/manual/aggregation/

Q13. Which of the following commands will add a collection that is stored in JSON file to a MongoDB cluster?
* mongodump
* mongoimport
* mongostore
* mongoexport
https://docs.mongodb.com/database-tools/mongoimport/

Q14. Select all true statements regarding to replica sets in MongoDB.
* We can have up to 50 replica set members, but only 7 of those will be voting members.
* Replica set members have a fixed role assigned.
* Replica sets use failover to provide high availability to client applications.
Replica set members have a fixed role assigned. -> False The availability of the system will be ensured by failing over the role of primary to an available secondary node through an election. https://docs.mongodb.com/manual/replication/

Q15. What is MongoDB Charts?
* MongoDB product that helps you visualize data stored in an Atlas cluster.
* A feature that displays data about the performance of your Atlas cluster.
* An application that allows you to embed on your website visualizations that are created in other applications.
https://docs.mongodb.com/charts/

Q16. Which of the following commands can you use to exports data in BSON format from a MongoDB cluster?
* mongodump
* mongoexport
* mongostore
* mongoimport
https://docs.mongodb.com/database-tools/mongodump/

Q17. Suppose you are connected to mongod instance that is already running on port 27000 as admin user. You have to dump reviews collection from the restaurants database to BSON file. Which command should you use?
* mongodump &#8211;db restaurants &#8211;collection reviews
* mongodump &#8211;db reviews &#8211;collection restaurants
* mongoexport &#8211;db restaurants &#8211;collection reviews
https://docs.mongodb.com/database-tools/mongodump/

Q18. What can you deduce from the explain() method?

For example:

db.collection.find().explain()
* All stages that the query must go through with details about the time it takes, number of documents processed and returned to the next stage in the pipeline.
* All available indexes for this collection.
* The index used by the chosen plan.
* Whether the sort stage was performed by index or performed in memory.
https://docs.mongodb.com/manual/reference/method/cursor.explain/

Q19. What is a document in MongoDB?

* It&#8217;s an ordered collection of keys (fields) with assigned values. The keys in each document are strings of characters. Documents in MongoDB cannot contain duplicate keys.
* It&#8217;s a text file containing the configuration of the MongoDB instance.
* It&#8217;s a table that contains information about all collections in the database.

Q20. Suppose you have an accounts collection with the following document structure: { _id: ObjectId(&#8220;61af47c6e29861661d063714&#8221;), account_id: 1010, type: &#8216;investment&#8217;, limit: 1000000 }, { _id: ObjectId(&#8220;61af47c6e29861661d063715&#8221;), account_id: 4336, type: &#8216;derivatives&#8217;, limit: 100000 }, { _id: ObjectId(&#8220;61af47c6e29861661d063716&#8221;), account_id: 4336, type: &#8216;commodity&#8217;, limit: 1000 } Which of the following documents would be modified by this update? db.accounts.updateMany({ &#8220;type&#8221;: &#8220;investment&#8221; }, { &#8220;$set&#8221;: { &#8220;limit&#8221;: 2000000 } })
* { _id: ObjectId(&#8220;61af47c6e29861661d063715&#8221;), account_id: 4336, type: &#8216;derivatives&#8217;, limit: 100000 }
* { _id: ObjectId(&#8220;61b1bde1ceb6f770f56b0cd9&#8221;), account_id: 4915, type: &#8216;investment&#8217;, limit: 1000000 }
* { _id: ObjectId(&#8220;61af47c6e29861661d067825&#8221;), account_id: 7355, type: &#8216;commodity&#8217;, limit: 500000 }
* { _id: ObjectId(&#8220;61af47c6e29861661d063714&#8221;), account_id: 1010, type: &#8216;investment&#8217;, limit: 1000000 }
https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/

Q21. How many indexes will the following command create?

db.products.createIndex({ product_name: 1, product_category: -1 })
* 0
* 3
* 1
* 2
https://docs.mongodb.com/manual/reference/method/db.collection.createIndex/

Q22. In your database a movies collection is given where each document has the following structure: { _id: ObjectId(&#8220;573a1391f29313caabcd9264&#8221;), genres: [&#8216;Romance&#8217;, &#8216;Drama&#8217;], title: &#8216;The Divorcee&#8217;, languages: [&#8216;English&#8217;, &#8216;French&#8217;], year: 1930, imdb: { rating: 6.9, votes: 1740, id: 20827 }, countries: [&#8216;USA&#8217;] } Which of the following queries will return all movies that are in the Fantasy and Drama (both) genre?
* db.movies.find({ genres: { $elemMatch: [&#8216;Fantasy&#8217;, &#8216;Drama&#8217;] } })
* db.movies.find({ genres: { $in: [&#8216;Fantasy&#8217;, &#8216;Drama&#8217;] } })
* db.movies.find({ genres: { $size: [&#8216;Fantasy&#8217;, &#8216;Drama&#8217;] } })
* db.movies.find({ genres: { $all: [&#8216;Fantasy&#8217;, &#8216;Drama&#8217;] } })
https://docs.mongodb.com/manual/reference/operator/query/all/

Q23. We have a movies collection with the following document structure: { _id: ObjectId(&#8220;573a1390f29313caabcd6223&#8221;), genres: [&#8216;Comedy&#8217;, &#8216;Drama&#8217;, &#8216;Family&#8217;], title: &#8216;The Poor Little Rich Girl&#8217;, released: ISODate(&#8220;1917-03-05T00:00:00.000Z&#8221;), year: 1917, imdb: { rating: 6.9, votes: 884, id: 8443 } }, { _id: ObjectId(&#8220;573a13e3f29313caabdc08a4&#8243;), genres: [&#8216;Horror&#8217;, &#8216;Thriller&#8217;], title: &#8216;Mary Loss of Soul&#8217;, year: 2014, imdb: { rating: &#8221;, votes: &#8221;, id: 2904798 } } We need to use Aggregation Framework to calculate the following aggregates: -> average imdb rating -> minimum imdb rating -> maximum imdb rating Expected output: [{ _id: null, avg_rating: 6.6934040649367255, min_rating: 1.6, max_rating: 9.6 }] Please note that some documents have &#8220;&#8221; (empty string) for the field &#8220;imdb.rating&#8221;. Exclude these documents before

aggregation. Which pipeline should you use?
* [{ $match: { &#8220;imdb.rating&#8221;: { $ne: &#8220;&#8221; } } }, { $group: { _id: null, avg_rating: { $avg: &#8220;imdb.rating&#8221; }, min_rating: { $min: &#8220;imdb.rating&#8221; }, max_rating: { $max: &#8220;imdb.rating&#8221; } } }]
* [{ $match: { &#8220;imdb.rating&#8221;: { $ne: &#8220;&#8221; } } }, { $group: { _id: null, avg_rating: { $avg: &#8220;$imdb.rating&#8221; }, min_rating: { $min: &#8220;$imdb.rating&#8221; }, max_rating: { $max: &#8220;$imdb.rating&#8221; } } }]
* [{ $group: { _id: null, avg_rating: { $avg: &#8220;$imdb.rating&#8221; }, min_rating: { $min: &#8220;$imdb.rating&#8221; }, max_rating: { $max: &#8220;$imdb.rating&#8221; } } }]
* [{ $match: { &#8220;imdb.rating&#8221;: { $ne: &#8220;&#8221; } } }, { $group: { _id: null, avg_rating: { avg: &#8220;$imdb.rating&#8221; }, min_rating: { min: &#8220;$imdb.rating&#8221; }, max_rating: { max: &#8220;$imdb.rating&#8221; } } }]
https://docs.mongodb.com/manual/aggregation/

Q24. Select true statements about shard keys.
* Shard keys are used to organize documents into chunks and shards, and mongos can use this to route queries.
* Shard keys can include the _id field, but they don&#8217;t have to.
* All shard keys must be supported by an index with the same document fields.
https://docs.mongodb.com/manual/core/sharding-shard-key/

Q25. Select all true statements about reconfiguring a replica set with rs.reconfig().
* When we reconfigure a replica set with rs.reconfig(), we don&#8217;t need to restart any of the individual nodes.
* When we reconfigure a replica set with rs.reconfig(), we need to update all of the nodes&#8217; configuration files.
https://docs.mongodb.com/manual/reference/method/rs.reconfig/

Q26. Select all true statements regarding to unique indexes in MongoDB.
* In MongoDB we cannot enforce a unique constraint on compound indexes.
* A unique index ensures that the indexed fields don&#8217;t store duplicate values.
* By default, MongoDB creates a unique index on the _id field.
* When creating a unique index, we need to pass an additional unique option set to true. For example:
* db.products.createIndex({ manufacturer_id: 1 }, { unique: true })
https://docs.mongodb.com/manual/core/index-unique/

Q27. We have a movies collection with the following document structure: { _id: ObjectId(&#8220;573a1390f29313caabcd6223&#8221;), genres: [&#8216;Comedy&#8217;, &#8216;Drama&#8217;, &#8216;Family&#8217;], title: &#8216;The Poor Little Rich Girl&#8217;, released: ISODate(&#8220;1917-03-05T00:00:00.000Z&#8221;), year: 1917, imdb: { rating: 6.9, votes: 884, id: 8443 } } We need to extract all movies from this collection where genres includes both &#8216;Crime&#8217; and &#8216;Mystery&#8217;. Which query should we use?
* db.movies.find({ genres: { $any: [&#8220;Crime&#8221;, &#8220;Mystery&#8221;] } })
* db.movies.find({ genres: { $all: [&#8220;Crime&#8221;, &#8220;Mystery&#8221;] } })
* db.movies.find({ genres: { $nin: [&#8220;Crime&#8221;, &#8220;Mystery&#8221;] } })
* db.movies.find({ genres: { $in: [&#8220;Crime&#8221;, &#8220;Mystery&#8221;] } })
https://docs.mongodb.com/manual/reference/operator/query/all/

Q28. You have a configuration file for mongod instance called mongod.conf in your working directory. How can you launch mongod instance with this configuration file?
* mongod &#8211;conf mongod.conf
* mongod &#8211;use mongod.conf
* mongod &#8211;config mongod.conf

* mongod -f mongod.conf
https://docs.mongodb.com/manual/reference/configuration-options/#use-the-configuration-file

Q29. How to create a new user with the root role named root_user with password root123?
* use admin db.createUser({ user: &#8220;root_user&#8221;, pwd: &#8220;root123&#8221;, roles: [&#8220;dbAdmin&#8221;] })
* use admin db.createUser({ user: &#8220;root_user&#8221;, pwd: &#8220;root123&#8221;, roles: [&#8220;root&#8221;] })
* use admin db.createUser({ user: &#8220;root_user&#8221;, pwd: &#8220;root123&#8221;, roles: [&#8220;clusterAdmin&#8221;] })
* use admin db.createUser({ user: &#8220;root_user&#8221;, pwd: &#8220;root123&#8221;, roles: [&#8220;dbAdminAnyDatabase&#8221;] })
https://docs.mongodb.com/manual/reference/method/db.createUser/

Q30. Suppose you have a sales collection with the following document structure: { _id: ObjectId(&#8220;5bd761dcae323e45a93ccfe8&#8221;), saleDate: ISODate(&#8220;2015-03-23T21:06:49.506Z&#8221;), items: [{ name: &#8216;printer paper&#8217;, tags: [&#8216;office&#8217;, &#8216;stationary&#8217;], price: Decimal128(&#8220;40.01&#8221;), quantity: 2 } { name: &#8216;pens&#8217;, tags: [&#8216;writing&#8217;, &#8216;office&#8217;, &#8216;school&#8217;, &#8216;stationary&#8217;], price: Decimal128(&#8220;56.12&#8221;), quantity: 5 }, { name: &#8216;notepad&#8217;, tags: [&#8216;office&#8217;, &#8216;writing&#8217;, &#8216;school&#8217;], price: Decimal128(&#8220;18.47&#8221;), quantity: 2 }], storeLocation: &#8216;Denver&#8217;, couponUsed: true, purchaseMethod: &#8216;Online&#8217; } Which of the following queries will return all document sales with &#8216;printer paper&#8217; sold?
* db.sales.find({ items: { $match: { name: &#8216;printer paper&#8217; } } })
* db.sales.find({ items: { $elemMatch: { name: &#8216;printer paper&#8217; } } })
* db.sales.find({ items: { elemMatch: { name: &#8216;printer paper&#8217; } } })
Explanation: https://docs.mongodb.com/manual/reference/operator/query/elemMatch/

Q31. Select all true statements regarding to schema validation in MongoDB.
* Validation rules are on a per-collection basis.
* MongoDB doesn&#8217;t provide the capability to perform schema validation during updates and insertions.
* We need to use db.createCollection() with the validator option to specify validation rules for a new collection.
https://docs.mongodb.com/manual/core/schema-validation/

Q32. Select true statements about index performance.
* For the fastest processing, we should make sure our indexes fit entirely in RAM.
* Indexes cannot decrease insert throughput.
* Indexes don&#8217;t have to be completely stored in RAM, but permanent disk access to retrieve index information will have a performance impact.
https://www.mongodb.com/blog/post/performance-best-practices-indexing

Q33. How can we represent a one-to-one relationship in MongoDB?
* We can link to a single document in another collection.
* We can embed fields as a child document in the document.
* We can embed fields in the document.
https://docs.mongodb.com/manual/tutorial/model-embedded-one-to-one-relationships-between-documen ts/

Q34. Does MongoDB support query operations that perform a text search of string content?
* Yes
* No
https://docs.mongodb.com/manual/text-search/

Ultimate Guide to the C100DEV - Latest Edition Available Now: https://www.braindumpsit.com/C100DEV_real-exam.html]