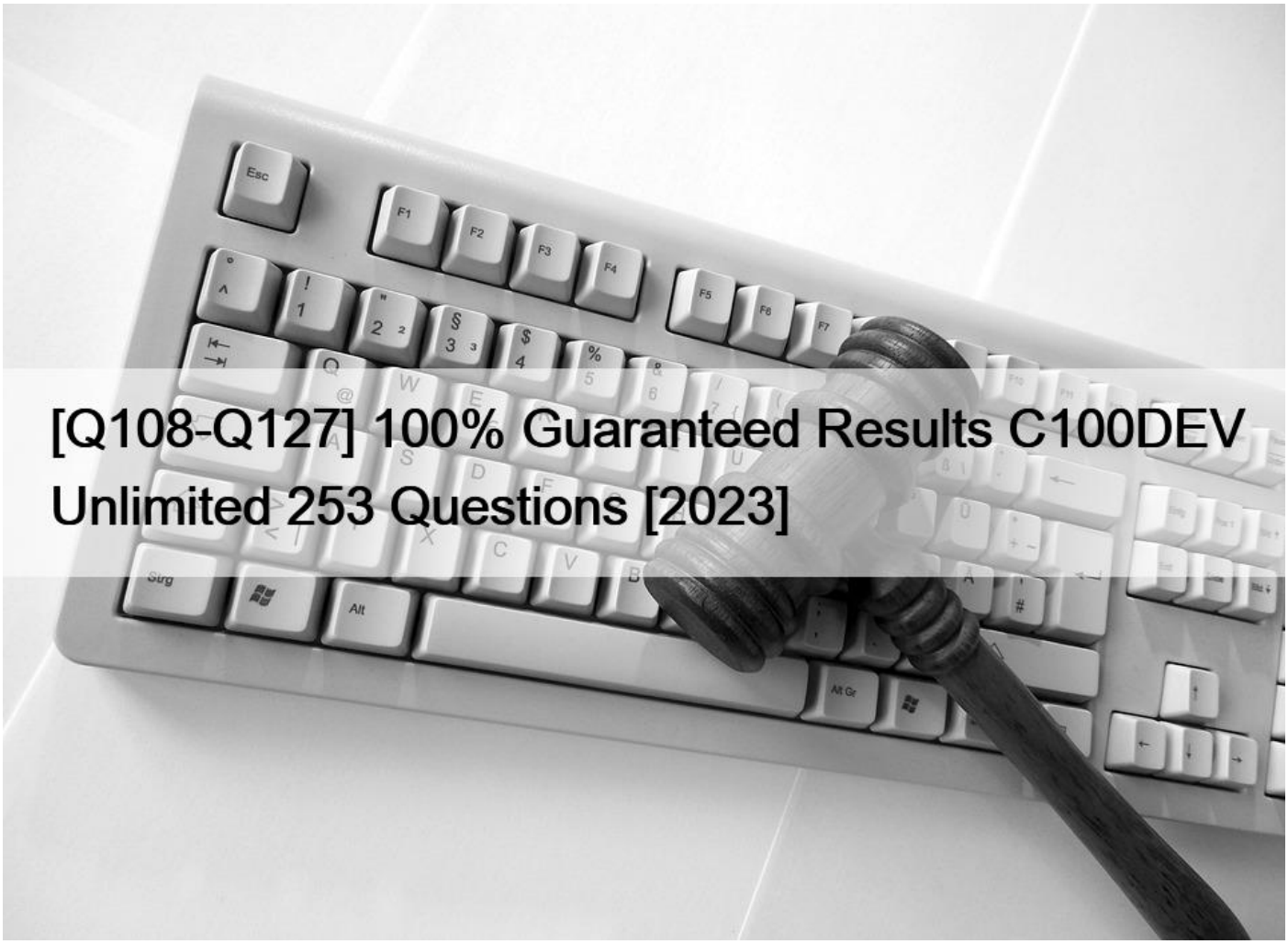
[Q108-Q127 100% Guaranteed Results C100DEV Unlimited 253 Questions [2023



100% Guaranteed Results C100DEV Unlimited 253 Questions [2023]
C100DEV Dumps PDF - Want To Pass C100DEV Fast

How to be successful in the Developer Associate by getting MongoDB C100DEV Certified

If you want to get started with MongoDB, then you should consider getting the MongoDB Certified Developer Associate certification. As a MongoDB Certified Developer Associate, you will be able to use key data modeling skills and concepts, as well as get an understanding of how to use MongoDB effectively in your applications. Our complete package free demo can help improve your preparation for the certified professionals exam pass guarantee. If you fail the exam, we will refund your full purchase price.

The best way to be successful in the Developer Associate by getting MongoDB C100DEV Certified is by studying hard and taking practice exams.

Read these tips on how you can do that:

Study hard: The first thing you need to do is to study hard. You can't just memorize things and expect them to come out right in the exam. You need to understand concepts and practice them until they become second nature. And don't forget about practice exams! They are very useful when it comes to preparing for exams like these because they allow you to test yourself before going into the actual exam room.

Take practice exams: While studying is important and all, practice exams are even more important if you want to pass your exam on the first try! MongoDB **C100DEV Dumps** help give you a sense of what kinds of questions will appear on your exam so that when you go into the actual room, there won't be any surprises or unexpected questions popping up!

MongoDB C100DEV is a certification exam designed for developers who have a thorough understanding of MongoDB and its applications. The exam tests their knowledge of MongoDB's fundamentals, data modeling, aggregation, and application development using MongoDB. This certification is ideal for developers who are looking to demonstrate their MongoDB skills and knowledge to potential employers or clients.

The MongoDB C100DEV (MongoDB Certified Developer Associate) Exam is a certification program designed to test a developer's proficiency in MongoDB technology. This exam is intended for developers who are interested in gaining a deep understanding of MongoDB and its related technologies. The MongoDB C100DEV Exam is an ideal certification for those who want to demonstrate their knowledge and skills to potential employers or clients.

NO.108 There is a gamers collection in your database with the following document structure:

{ _id: 1, level: 15, is_active: true }, { _id: 2, level: 14, is_active: true }, { _id: 3, level: 7, is_active: false }

How do you update the value of the level field to 20 for a player with an id = 2?

Expected result:

{ _id: 1, level: 15, is_active: true }, { _id: 2, level: 20, is_active: true }, { _id: 3, level: 7, is_active: false }
* db.gamers.update({ _id: 2 }, { level: 20 })
* db.gamers.update({ _id: 2 }, { $inc: { level: 20 } })
* db.gamers.update({ _id: 2 }, { $set: { level: 20 } })
https://docs.mongodb.com/manual/reference/operator/update/set/

NO.109 Select true statements about capped collections.
* Capped collections automatically remove the oldest documents in the collection to make room for new documents.
* If the collection is full, the oldest document in the collection is removed when a new document is added to the collection.
* Capped collections are fixed-size collections.
* Capped collections guarantee preservation of the insertion order.
https://docs.mongodb.com/manual/core/capped-collections/

NO.110 Suppose you have a movies collection with the following document structure: { _id: ObjectId(“573a1390f29313caabcd60e4”), title: ‘The Immigrant’, fullplot: “Charlie is on his way to the USA. He wins in a card game, puts the money in Edna’s bag (she and her sick mother have been robbed of everything). When he retrieves a little for himself he is accused of being a thief. Edna clears his name. Later, broke, Charlie finds a coin and goes into a restaurant.” } You want to sort result set by a relevance score computed by MongoDB in text search query and extract only three documents with the highest score. Which query do you need to use?
* db.movies.find({ $text: { $search: ‘spaceship’ } }, { score: { $meta: ‘textScore’ } },).sort({ score: { $meta: ‘textScore’ } })
* db.movies.find({ $text: { $search: ‘spaceship’ } }, { score: { $meta: ‘textScore’ } },).sort({ score: { $meta: ‘textScore’ } }).limit(3)
* db.movies.find({}, { score: { $meta: ‘textScore’ } },).sort({ score: { $meta: ‘textScore’ } }).limit(3)

https://docs.mongodb.com/manual/text-search/ https://docs.mongodb.com/manual/core/index-text/

NO.111 Suppose you have a books collection with title field. Which of the following queries will return all books with a title ending in ‘ian’?
* db.movies.find({ title: { $regex: /.*ian$/ } })
* db.movies.find({ title: { $text: /.*ian$/ } })
* db.movies.find({ title: { $regex: /.*ian/ } })
* db.movies.find({ title: { $regex: /ian/ } })
https://docs.mongodb.com/manual/reference/operator/query/regex/

NO.112 What is the built-in database called local in MongoDB for?
* The local database plays an important role in the authentication and authorization process. Certain actions performed by administrators also require access to this database.
* The local database is used to store information about shards in shared MongoDB cluster.
* The local database stores data describing the MongoDB server. For replica sets, it also stores information about the replication process.
https://docs.mongodb.com/manual/reference/local-database/

NO.113 We have a movies collection with the following document structure: { _id: ObjectId(“573a1390f29313caabcd6223”), genres: [‘Comedy’, ‘Drama’, ‘Family’], title: ‘The Poor Little Rich Girl’, released: ISODate(“1917-03-05T00:00:00.000Z”), year: 1917, imdb: { rating: 6.9, votes: 884, id: 8443 } } We need to use Aggregation Framework to fetch all movies from this collection where ‘Drama’ is not in genres list and the minimum ‘imdb.votes’ is at least 100. Additionally, in the projection stage, we want to leave only the following fields: -> title -> genres -> imdb.votes We also want to sort the result set by decreasing imdb votes and limit the number of documents retuned to 5. Example output: [{ imdb: { votes: 1294646 }, genres: [‘Action’, ‘Mystery’, ‘Sci-Fi’], title: ‘Inception’ }, { imdb: { votes: 1109724 }, genres: [‘Adventure’, ‘Fantasy’], title: ‘The Lord of the Rings: The Fellowship of the Ring’ }, { imdb: { votes: 1081144 }, genres: [‘Adventure’, ‘Fantasy’], title: ‘The Lord of the Rings: The Return of the King’ }, { imdb: { votes: 1080566 }, genres: [‘Action’, ‘Sci-Fi’], title: ‘The Matrix’ }, { imdb: { votes: 1004805 }, genres: [‘Action’, ‘Thriller’], title: ‘The Dark Knight Rises’ }] Which pipeline should you use?
* [{ $match: { genres: { $nin: [‘Drama’] }, ‘imdb.votes’: { $gte: 100 } } }, { $project: { _id: 0, title: 1, genres: 1, ‘imdb.votes’: 1 } }, { $sort: { ‘imdb.votes’: -1 } }, { $limit: 5 }]
* [{ $match: { genres: { $in: [‘Drama’] }, ‘imdb.votes’: { $gte: 100 } } }, { $project: { _id: 0, title: 1, genres: 1, ‘imdb.votes’: 1 } }, { $sort: { ‘imdb.votes’: -1 } }, { $limit: 5 }]
* [{ $match: { genres: { $nin: [‘Drama’] }, ‘imdb.votes’: { $gte: 100 } } }, { $project: { _id: 0, title: 1, genres: 1, ‘imdb.votes’: 1 } }, { $sort: { ‘imdb.votes’: -1 } }]
https://docs.mongodb.com/manual/aggregation/

NO.114 Why is MongoDB using BSON instead of JSON to store data? Select all that apply.
* BSON format is not human readable (BSON is machine-readable only).
* BSON supports more data types than JSON.
* BSON contains metadata to describe a document/object.
* BSON simply means ‘Binary JSON’.
https://www.mongodb.com/json-and-bson

NO.115 You have the following index in a movies collection: { “title”: 1, “imdb.rating”: -1, “imdb.votes”: -1, “type”: 1 } Can the following query use the given index for both filtering and sorting?

db.movies.find({ “title”: { “$lt”: “M” } }).sort({ “imdb.votes”: -1 })
* Yes
* No
No, this query does not use equality in the index prefix. When you use an index to filter and sort, the query must contain equality conditions on all prefix keys that precede the sort keys. Also, it skipped the next key in the “imdb.rating” prefix in the sort predicate. https://docs.mongodb.com/manual/indexes/

NO.116 You have the following index in a products collection: { product_category: 1, product_name: 1, origin_country: -1 } Which of the following queries can use this index? Check all that apply.
* db.products.find({ origin_country: ‘Bangladesh’, product_name: ‘AIR MAX 90’, product_category: ‘shoes’ })
* db.products.find({ product_name: ‘AIR MAX 90’, product_category: ‘shoes’, origin_country: ‘Bangladesh’ })
* db.products.find({ product_category: ‘shoes’, product_name: ‘AIR MAX 90’, origin_country: ‘Bangladesh’ })
* db.products.find({ product_name: ‘AIR MAX 90’, origin_country: ‘Bangladesh’ })
https://docs.mongodb.com/manual/indexes/

NO.117 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/

NO.118 We have the following schema for a movies collection: { _id: ObjectId, title: String, genres: Array, languages: Array, year: 32-bit integer } And the following index on the movies collection: { title: 1 } Which of the following queries will use the given index to perform the sorting stage?
* db.movies.find({ genres: “Drama” }).sort({ year: 1 })
* db.movies.find({}).sort({ title: -1 })
* db.movies.find({}).sort({ title: 1 })
* db.movies.find({ genres: “Drama” }).sort({ title: 1 })
db.movies.find({ genres: “Drama” }).sort({ title: 1 }) Yes, in this case the index will be used to retrieve the sorted documents and then it will filter the movies matching the genre. https://docs.mongodb.com/manual/indexes/

NO.119 What is data modeling in MongoDB world?
* A way to organize fields in a document to support your application performance and querying capabilities.
* A way to build your application based on how your data is stored.
* A way to show your database with graphs.
* A way to decide whether to store your data in the cloud or locally.
https://docs.mongodb.com/manual/core/data-modeling-introduction/

NO.120 Select all default configurations for mongod.
* By default, mongod doesn’t enforce authentication.
* By default, mongod is only bound to localhost (or 127.0.0.1).
* The default port for mongod is 27017.
* The default data directory for mongod is /data/db
https://docs.mongodb.com/manual/reference/configuration-options/

NO.121 Select all true statements regarding to transactions in MongoDB.
* Transactions are associated with a session.
* We cannot read/write to collections in the config, admin, or local databases.
* The collections used in a transaction can be in different databases.
https://docs.mongodb.com/manual/core/transactions/

NO.122 In your database there is a collection named trips with the following document structure: { ‘_id’: ObjectId(“572bb8222b288919b68abf6d”), ‘trip_duration’: 858, ‘start_station id’: 532, ‘end_station_id’: 401, ‘bike_id’: 17057, ‘start_station_location’: { type: ‘Point’, coordinates: [-73.960876, 40.710451] }, ‘end_station_location’: { type: ‘Point’, coordinates: [-73.98997825, 40.72019576] }, ‘start_time’: ISODate(“2016-01-01T00:09:31.000Z”), ‘stop_time’: ISODate(“2016-01-01T00:23:49.000Z”) } How can you extract all trips from this collection ended at stations that are to the west of the -73.5 longitude coordinate?
* db.trips.find({ ‘coordinates’: { $lt: -73.5 } })
* db.trips.find({ ‘end_station_location.coordinates.0’: { $gt: -73.5 } })
* db.trips.find({ ‘end_station_location.coordinates.0’: { $lt: -73.5 } })
* db.trips.find({ ‘end_station_location.coordinates.1’: { $lt: -73.5 } })
https://docs.mongodb.com/manual/reference/operator/query/lt/

NO.123 Given a movies collection where each document has the following structure: { _id: ObjectId(“573a1390f29313caabcd60e4”), genres: [‘Short’, ‘Comedy’, ‘Drama’], title: ‘The Immigrant’, year: 1917, imdb: { rating: 7.8, votes: 4680, id: 8133 }, countries: [‘USA’] } Which of the following queries will find all Comedy movies that were made in 2000? (select 2)
* db.movies.find({ year: 2000 }, { genres: “Comedy” })
* db.movies.find({ $or: [{ year: 2000 }, { genres: “Comedy” }] })
* db.movies.find({ year: 2000, genres: “Comedy” })
* db.movies.find({ $and: [{ year: 2000 }, { genres: “Comedy” }] })
https://docs.mongodb.com/manual/reference/method/db.collection.find/
https://docs.mongodb.com/manual/reference/operator/query/and/

NO.124 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/

NO.125 How to shutdown the server? (Mongo shell)
* use admin
* db.shutdown()
* use admin
* db.shutdownServer()
* use admin
* db.disconnect()

https://docs.mongodb.com/manual/reference/method/db.shutdownServer/

NO.126 Select all true statements about data modeling in MongoDB.
* MongoDB can easily handle both unstructured and structured data sets.
* Unlike SQL databases, where the table schema must be specified and declared before inserting the data, MongoDB collections don’t, by default, require documents to have the same schema.
* The field set and data type for each field can differ across documents within a collection.
* In MongoDB we can enforce document validation rules for a collection during update and insert operations.
https://docs.mongodb.com/manual/core/data-modeling-introduction/

NO.127 Suppose you have a companies collection in your database. Only the following documents are stored in this collection:

{ _id: ObjectId(“52cdef7c4bab8bd675297da4”), name: ‘Powerset’, category_code: ‘search’, founded_year: 2006 }, { _id: ObjectId(“52cdef7c4bab8bd675297da5”), name: ‘Technorati’, category_code: ‘advertising’, founded_year: 2002 }, { _id: ObjectId(“52cdef7c4bab8bd675297da7”), name: ‘AddThis’, category_code: ‘advertising’, founded_year: 2004 }, { _id: ObjectId(“52cdef7c4bab8bd675297da8”), name: ‘OpenX’, category_code: ‘advertising’, founded_year: 2008 }, { _id: ObjectId(“52cdef7c4bab8bd675297daa”), name: ‘Sparter’, category_code: ‘games_video’, founded_year: 2007 }, { _id: ObjectId(“52cdef7c4bab8bd675297dac”), name: ‘Veoh’, category_code: ‘games_video’, founded_year: 2004 }, { _id: ObjectId(“52cdef7c4bab8bd675297dae”), name: ‘Thoof’, category_code: ‘web’, founded_year: 2006 }

How many documents will be deleted when executing the following query?

db.companies.deleteMany({ “category_code”: “advertising” })
* 7
* 0
* 3
* 4
https://docs.mongodb.com/manual/reference/method/db.collection.deleteMany/