db.sales.drop();

true

db.sales.insertMany([

{ "\_id" : 1, "item" : "Americanos", "price" : 5, "size": "Short", "quantity" : 22, "date" : ISODate("2022-01-15T08:00:00Z") },

{ "\_id" : 2, "item" : "Cappuccino", "price" : 6, "size": "Short","quantity" : 12, "date" : ISODate("2022-01-16T09:00:00Z") },

{ "\_id" : 3, "item" : "Lattes", "price" : 15, "size": "Grande","quantity" : 25, "date" : ISODate("2022-01-16T09:05:00Z") },

{ "\_id" : 4, "item" : "Mochas", "price" : 25,"size": "Tall", "quantity" : 11, "date" : ISODate("2022-02-17T08:00:00Z") },

{ "\_id" : 5, "item" : "Americanos", "price" : 10, "size": "Grande","quantity" : 12, "date" : ISODate("2022-02-18T21:06:00Z") },

{ "\_id" : 6, "item" : "Cappuccino", "price" : 7, "size": "Tall","quantity" : 20, "date" : ISODate("2022-02-20T10:07:00Z") },

{ "\_id" : 7, "item" : "Lattes", "price" : 25,"size": "Tall", "quantity" : 30, "date" : ISODate("2022-02-21T10:08:00Z") },

{ "\_id" : 8, "item" : "Americanos", "price" : 10, "size": "Grande","quantity" : 21, "date" : ISODate("2022-02-22T14:09:00Z") },

{ "\_id" : 9, "item" : "Cappuccino", "price" : 10, "size": "Grande","quantity" : 17, "date" : ISODate("2022-02-23T14:09:00Z") },

{ "\_id" : 10, "item" : "Americanos", "price" : 8, "size": "Tall","quantity" : 15, "date" : ISODate("2022-02-25T14:09:00Z")}

]);

{

acknowledged: true,

insertedIds: {

'0': 1,

'1': 2,

'2': 3,

'3': 4,

'4': 5,

'5': 6,

'6': 7,

'7': 8,

'8': 9,

'9': 10

}

}

db.sales.find({$max:quantity});

ReferenceError: quantity is not defined

db.sales.aggregate([

{ $group: { \_id: null, maxQty: { $max: "$quantity" } } }

]);

{

\_id: null,

maxQty: 30

}

db.sales.aggregate([

{ $group: { \_id: "$item", maxQty: { $max: "$quantity" } } }

]);

{

\_id: 'Cappuccino',

maxQty: 20

}

{

\_id: 'Mochas',

maxQty: 11

}

{

\_id: 'Americanos',

maxQty: 22

}

{

\_id: 'Lattes',

maxQty: 30

}

db.sales2.aggregate([{$group:{\_id:"$item",maxrev:{$max:{$multiply:["$price","$quantity"]}}}}]);

db.sales.aggregate([{$group:{\_id:"$item",maxrev:{$max:{$multiply:["$price","$quantity"]}}}}]);

{

\_id: 'Americanos',

maxrev: 210

}

{

\_id: 'Lattes',

maxrev: 750

}

{

\_id: 'Mochas',

maxrev: 275

}

{

\_id: 'Cappuccino',

maxrev: 170

}

db.sales.aggregate([

{

$addFields: {

totalSale: { $multiply: ["$price", "$quantity"] }

}

},

{

$sort: { item: 1, totalSale: -1 }

},

{

$group: {

\_id: "$item",

salesList: { $push: "$totalSale" }

}

},

{

$project: {

secondMaxSale: { $arrayElemAt: ["$salesList", 1] }

}

}

]);

{

\_id: 'Americanos',

secondMaxSale: 120

}

{

\_id: 'Lattes',

secondMaxSale: 375

}

{

\_id: 'Cappuccino',

secondMaxSale: 140

}

{

\_id: 'Mochas'

}

db.sales.getIndexes()

[ { v: 2, key: { \_id: 1 }, name: '\_id\_' } ]

db.sales.createIndex({item:1})

item\_1

db.sales.getIndexes();

[

{ v: 2, key: { \_id: 1 }, name: '\_id\_' },

{ v: 2, key: { item: 1 }, name: 'item\_1' }

]

db.sales.dropIndex({item:1});

{ nIndexesWas: 2, ok: 1 }

db.sales.getIndexes();

[ { v: 2, key: { \_id: 1 }, name: '\_id\_' } ]

db.sales.createIndex({item:1})

item\_1

db.sales.getIndexes();

[

{ v: 2, key: { \_id: 1 }, name: '\_id\_' },

{ v: 2, key: { item: 1 }, name: 'item\_1' }

]

db.sales.dropIndex({item:1});

{ nIndexesWas: 2, ok: 1 }

db.sales.getIndexes();

[ { v: 2, key: { \_id: 1 }, name: '\_id\_' } ]

db.users.insertMany([

{ email: "john@test.com", name: "john"},

{ email: "jane@test.com", name: "jane"},

]);

{

acknowledged: true,

insertedIds: {

'0': ObjectId('6841a17c312a7e8cd543075c'),

'1': ObjectId('6841a17c312a7e8cd543075d')

}

}

db.users.createIndex({email:1},{unique:true});

email\_1

db.users.getIndexes();

[

{ v: 2, key: { \_id: 1 }, name: '\_id\_' },

{ v: 2, key: { email: 1 }, name: 'email\_1', unique: true }

]

db.locations.insertOne({

address: "Downtown San Jose, CA, USA",

lat: 37.335480,

long: -121.893028

});

{

acknowledged: true,

insertedId: ObjectId('6841a40d312a7e8cd543075e')

}

db.locations.insertOne({

address: "Downtown San Jose, CA, USA",

lat: 37.335480,

long: -121.893028

});

{

acknowledged: true,

insertedId: ObjectId('6841a416312a7e8cd543075f')

}

db.locations.insertOne({

address: "Central San Jose, CA, USA",

lat: 37.335480,

long: -121.893028

});

{

acknowledged: true,

insertedId: ObjectId('6841a4a3312a7e8cd5430760')

}

db.locations.createIndex(

{

lat: 1,

long: 1

},

{

unique: true

}

);

MongoServerError[DuplicateKey]: Index build failed: 7dd5553e-9866-4f98-88f4-7a9f1010419f: Collection test.locations ( ca673722-5bf4-4af9-8366-bc7101726829 ) :: caused by :: E11000 duplicate key error collection: test.locations index: lat\_1\_long\_1 dup key: { lat: 37.33548, long: -121.893028 }

db.movie.insertMany([

{ year: 2020, title: "The Silent Forest", award: "Best International Film" },

{ year: 2021, title: "Midnight Sky", award: "Best Visual Effects" },

{ year: 2022, title: "Nomadland", award: "Best Picture" },

{ year: 2020, title: "Soul", award: "Best Animated Feature" }

]);

{

acknowledged: true,

insertedIds: {

'0': ObjectId('6841a58e312a7e8cd5430761'),

'1': ObjectId('6841a58e312a7e8cd5430762'),

'2': ObjectId('6841a58e312a7e8cd5430763'),

'3': ObjectId('6841a58e312a7e8cd5430764')

}

}

db.movie.createIndex(

{ year: 1, title: 1 }

);

year\_1\_title\_1

package connection;

import com.mongodb.MongoClient;

import com.mongodb.MongoCredential;

import com.mongodb.client.MongoDatabase;

public class MongoDB {

public static void main(String[] args) {

try {

MongoClient db

= new MongoClient("localhost", 27017);

MongoCredential credential;

credential

= MongoCredential

.createCredential(

"GFGUser", "mongoDb",

"password".toCharArray());

System.out.println(

"Successfully Connected"

+ " to the database");

MongoDatabase database

= db.getDatabase("mongoDb");

System.out.println("Credentials are: "

+ credential);

}

catch (Exception e) {

System.out.println(

"Connection establishment failed");

System.out.println(e);

}

}

}