1. Установить сервер MondoDb и консольный клиент MongoDB Compass.
2. Создать базу данных по своему варианту.
3. Создать коллекции по варианту. Добавить данные в коллекции. Изменить и обновить несколько элементов коллекций.
4. Произвести выборку элементов, используя условные операции, операторы работы с массивами, операторы $exists, $type, $regex.

**Orders:**

db.Orders.find({status: {$exists: true, $ne: "Cancelled"}, totalWeight: {$gt: 200}, deliveryType: {$regex: /^Express/} });

db.Orders.find({

status: { $exists: true, $ne: "Cancelled" },

totalWeight: { $gt: 200, $lt: 1000 },

deliveryType: { $regex: /^Express/ },

clientId: { $in: [1, 2, 3] }

});

db.Orders.find({

clientId: { $in: [1, 2, 3] }

});

db.Orders.find({

clientId: { $in: [1, 2, 3, 4] },

status: { $ne: "Cancelled" }

});

**Services:**

db.Services.find({

baseRate: { $gte: 70 }

});

db.Service.find({

description: { $regex: /service/i }

});

**ServiceType:**

db.ServiceType.find({

$or: [

{ serviceTypeId: { $exists: true } },

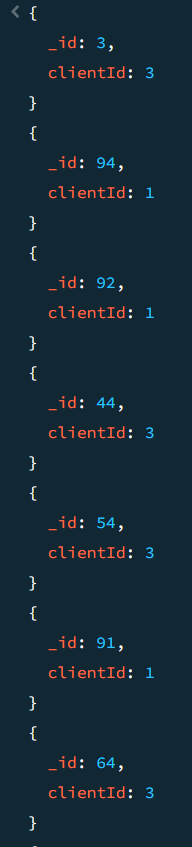
{ description: { $type: "string" } }

]

});

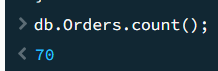
1. Произвести выборку элементов в своей базе данных, используя проекции.

db.Orders.find({ status: "Delivered" }, { orderId: 1, clientId: 1 });

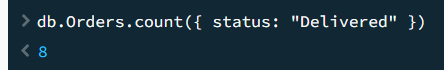


1. С помощью функции count() получить число всех элементов любой коллекции, а также число элементов с заданным фильтром.

db.Orders.count();



db.Orders.count({ status: "Delivered" })



1. Продемонстрировать использование методов limit() и skip().

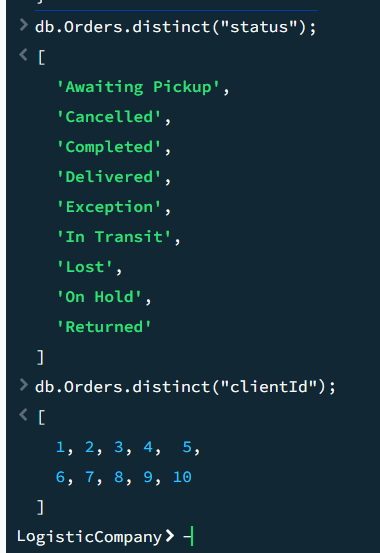
db.Orders.find().limit(5);

db.Orders.find().skip(10).limit(5);

1. Создать запросы с функцией distinct().

db.Orders.distinct("clientId");

db.Orders.distinct("status");



1. Используя метод aggregate(), продемонстрировать работу агрегатных операций.

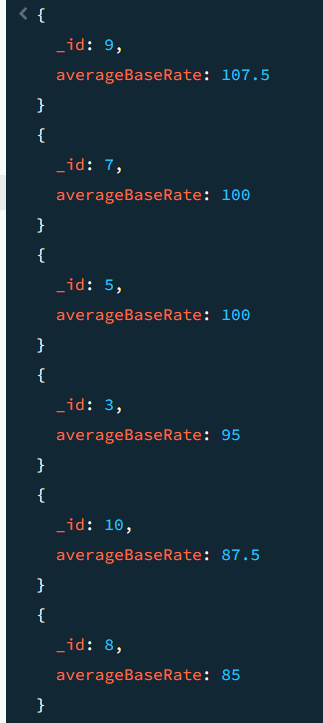
**Services:**

db.Services.aggregate([

{ $group: { \_id: "$serviceTypeId", averageBaseRate: { $avg: "$baseRate" } } },

{ $sort: { averageBaseRate: -1 } }

]);



**ServiceType:**

db.ServiceType.aggregate([

{ $lookup: {

from: "Services",

localField: "\_id",

foreignField: "serviceTypeId",

as: "services"

}},

{ $project: {

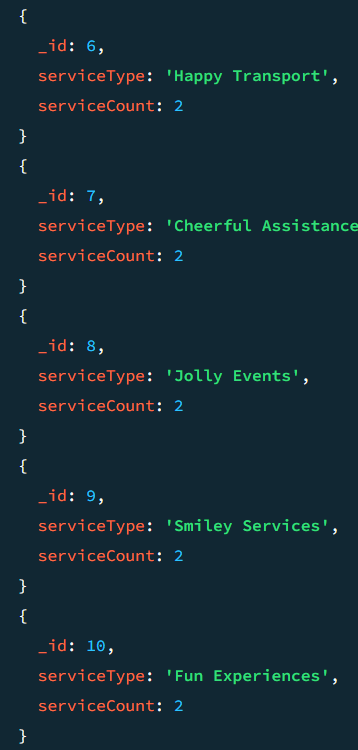
serviceType: "$name",

serviceCount: { $size: "$services" }

}},

{ $sort: { serviceCount: -1 } }

]);



**Orders:**

db.Orders.aggregate([

{ $group: { \_id: "$deliveryType", averageWeight: { $avg: "$totalWeight" } } },

{ $sort: { averageWeight: -1 } }

]);



1. Продемонстрировать использование пустого и непустого фильтра в секции match, группировку по нескольким ключам.

Пустой фильтр:

db.Orders.aggregate([

{ $match: {} },

{ $group: {

\_id: { clientId: "$clientId", status: "$status" },

totalOrders: { $sum: 1 },

totalWeight: { $sum: "$totalWeight" }

}},

{ $sort: { totalOrders: -1 } }

]);

Непустой фильтр:

db.Orders.aggregate([

{ $match: { status: "active" } },

{ $group: {

\_id: { clientId: "$clientId", status: "$status" },

totalOrders: { $sum: 1 },

totalWeight: { $sum: "$totalWeight" }

}},

{ $sort: { totalOrders: -1 } }

]);

Группировка по нескольким ключам:

db.Orders.aggregate([

{

$group: {

\_id: {

clientId: "$clientId",

status: "$status"

},

totalOrders: { $sum: 1 },

totalWeight: { $sum: "$totalWeight" }

}

},

{

$sort: { totalOrders: -1 }

}

]);