1)

{'$mul': { 'balance': 1.2 } -- увеличить на 20%  
{'$mul': { 'balance': .8 } -- уменьшить на 20%

2) '$set': {is\_blocked: false}  
  
  
'$unset': {is\_blocked: null}

3) db.users.find(  
 {  
 "country" :'China',  
 $and':   
 [  
 {is\_blocked': true },  
 {'balance': 0}  
 ]  
 }

4) db.users.find(  
 {  
 "country" :'China',  
 {is\_blocked': true },  
 $or': [  
   
 {'balance': 0},  
 {'balance': {'$exists': false}}  
 ]  
 }   
)

5) удалить все credit + premiumdb.calls.updateMany( {}, { '$pullAll': { 'topic': ['credit', 'premium'] } })

6) db.calls.updateMany(  
 {},  
 {  
 '$pull': {  
 'topic': 'credit'  
 }  
 }  
)

7) db.calls.updateMany(  
 {},  
 {  
 '$pullAll': {  
 'topic': ['credit', 'premium']  
 }  
 }  
)

8) db.users.find(  
 {  
 "country" :'China',  
 'is\_blocked': true ,  
 $or': [  
   
 {'balance': 0},  
 {'balance': {'$exists': false}}  
 ]  
 }   
)

9) db.users.find(  
 {  
 "country" :'China',  
 'is\_blocked': true ,  
 '$or': [  
   
 {'balance': 0},  
 {'balance': {'$exists': false}}  
 ]  
 }   
)

10) Задача  
 - найти всех пользователей из Германии, которые  
 (1) заблокированы  
 или  
 (2) имеют нулевой баланс

db.users.find(  
 {  
 'country': "Germany",  
 '$or': [  
 {'balance': 0},  
 {'balance': {'$exists': false}},  
 {'is\_blocked': true}  
 ]  
 }  
);

11) Задача (2)  
 - для всех разговоров с продолжительностью от 1 мин или темой 'premium'  
 (1) удалить тему 'news'  
 (2) добавить тему 'important'

11.1) db.calls.updateMany(  
 {  
 $or: [  
 {'duration\_secs': {$gte: 60}},  
 {'topic': 'premium'},  
 ]  
 }, // filter  
 {  
 $addToset: {'topic': 'important'},  
 $pull: {'topic': 'news'}  
 } // action  
);

11.2) db.calls.updateMany(  
 {  
 $or: [  
 {'duration\_secs': {$gte: 60}},  
 {'topic': 'premium'},  
 ]  
 }, // filter  
 {  
 $addToSet: {'topic': 'important'},  
 $pull: {'topic': 'news'}  
 } // action  
);

Аггрегация:

1) db.calls.aggregate([])

2) db.calls.aggregate([  
 {  
 '$match': {  
 'direction': 'out'  
 }  
 }  
])

3) db.calls.aggregate([  
 {  
 '$match': {  
 'direction': 'out'  
 }  
 },  
 {  
 '$sort': {  
 'duration\_secs': -1  
 }  
 }  
])

4) db.calls.aggregate([  
 { // этап 1 - фильтрация  
 '$match': {  
 'direction': 'out'  
 }  
 },  
 { // этап 2 - сортировка  
 '$sort': {  
 'duration\_secs': -1 // по убыванию  
 }  
 },  
 { // этап 3 - проекция  
 '$project': {  
 'duration\_secs': 1,  
 '\_id': 0  
 }  
 }  
])

5) Задача  
 - вывести пользователей не из Германии (но из Европы, с балансом больше нуля)  
 - сортировка по балансу (по убыванию)  
 - поля на выходе  
 -- имя  
 -- баланс  
 -- страна

'$match': {  
 'country': { '$ne': 'Germany' },  
 'is\_europe': true,  
 'balance': {'$gt': 0}  
 }  
 },  
   
 { // этап 2 - сортировка  
 '$sort': {  
 'balance': -1 // по убыванию   
 }  
 },  
 { // этап 3 - проекция  
 '$project': {  
 'fullname': 1,  
 'balance':1,  
 'country':1,  
 '\_id': 0  
 }  
 }

.

6)

Задача  
 - вывести пользователя с самым боьлшим балансом  
   
   
 db.users.aggregate([  
 {  
 '$sort': {  
 'balance': -1 // по убыванию   
 }  
 },  
 {  
 '$limit': 1  
 }  
])

page = 1 // номер страницы  
limit = 5 // ко-во на стран  
skip = limit \* (page-1)  
  
===CASE 1 ==  
page = 1  
limit = 5  
skip = 5 \* 0 = 0  
===CASE 2===  
page = 2  
limit = 5  
skip = 5 \* (2-1) == 5

7)

ЗАДАЧА  
 - вывести одного произвольного пользователя, у которого  
 (1) отрицательный баланс  
 и  
 (2) статус "не премиум"

7.1) db.users.aggregate([  
 {  
 '$match': {  
 'balance': { '$lt': 0 },  
 'is\_premium': { '$ne': true }  
 }  
 },  
 {  
 '$sample': {  
 'size': 1  
 }  
 }  
])

7.2) db.users.aggregate([  
 {  
 '$match': {  
 'balance': { '$lt': 0 },  
 'is\_premium': { '$ne': true }  
 }  
 },  
 {  
 '$count': 'users\_to\_call'  
 }  
])

7.3) db.calls.aggregate([  
 {  
 '$lookup': {  
 'from': 'users',  
 'localField': 'user\_id',  
 'foreignField': 'id',  
 'as': 'user'  
 }  
 }  
])

8) ЗАДАЧА  
 - вывести транзакции, отправленные из China

8.1) db.transactions.aggregate([  
 {  
 '$lookup': {  
 'from': 'users',  
 'localField': 'sender\_id',  
 'foreignField': 'id',  
 'as': 'sender'  
 }  
 },  
 {  
 '$match': {  
 'sender.country': 'China'  
 }  
 }   
])

8.1) db.transactions.aggregate([  
 {  
 '$lookup': {  
 'from': 'users',  
 'localField': 'sender\_id',  
 'foreignField': 'id',  
 'as': 'sender'  
 }  
 },  
 {  
 '$lookup': {  
 'from': 'users',  
 'localField': 'recipient\_id',  
 'foreignField': 'id',  
 'as': 'recipient'  
 }  
 },  
 {  
 '$match': {  
 '$or': [  
 {'sender.country': 'China'},  
 {'recipient.country': 'China'},  
 ]  
 }  
 }   
])

9) db.goods.aggregate([  
 {  
 '$project': {  
 '\_id': 0,  
 'id': 1,  
 'price\_eur': '$price',  
 'price\_usd': {  
 '$multiply': ['$price', 1.05]  
 }  
 }  
 }  
])

10) db.goods.aggregate([  
 {  
 '$addFields': {  
 'price\_usd': {  
 '$multiply': ['$price', 1.05]  
 }  
 }  
 }  
])

**GenTech / Mar 7, 2023**

**Что такое первичный ключ?**

* уникальный идентификатор сущности (уникальное поле)
* в качестве значения может использовать UUID
* неизменяемый

**Распространенная архитектура хранения данных**

ОДНА СУЩНОСТЬ (ОБЪЕКТ) == ОДНА ЗАПИСЬ

**Коллекция в MongoDB**

* аналог таблицы
* группирует однотипные сущности

**Как работает СУБД?**

* работает по схеме ЗАПРОС-ОТВЕТ (арх. КЛИЕНТ-СЕРВЕР)
* BE --> DBMS --> DB

**Типовые подсистемы (части) проекта**

* DB
* FE
* BE
* Deployment

deleteMany(filter)

deleteOne(filter)

countDocuments(filter)

updateMany(filter, action)

**Базовые структуры хранения в MongoDB**

* скаляр
* ассоц. массив
* массив
* множество (уникальные значения)

**Как работает метод (функция)**

* IPO
* вход --> | метод | --> выход

**Базовые операторы модификации**

$inc увеличение

$mul умножение

$push добавляет значение **в массив**

$addToSet добавление значение **в множество**

$pull удаляет один элемент из массива/множество

$pullAll удаляет несколько значений

$each (для $addToSet и $push) добавить список значений в массив или множество

Рефакторинг кода - перевыпуск (перепроизводство)

**MongoDB Aggregation Framework**

* реализуется через метод aggregate()
  + аргументы
    - pipeline (массив)
* инструмент для анализа, обработки данных
* позволяет получить вычисленные данные

**Основные стадии (этапы) обработки (stages) - операторы**

$match фильтрация

$sort сортировка

-1 по убыванию

1 по возрастанию

$project проекция (полей)

$limit ограничение ко-ва документов

$skip пропустить документы

$sample получить произвольные документы

$count посчитать ко-во документов

$lookup объединение коллекций