데이터베이스응용 조별과제

산업데이터사이언스학부 201804248
정진우
산업데이터사이언스학부 201804238
이희구
산업데이터사이언스학부 202004224
이서영

■ 선정 데이터: **은행 고객 데이터**

본 팀은 데이터베이스를 bank 로 선정하였다. Collection 은 bankname 으로 설정하였다.

■ **Collection 에 들어가는 Doc 의 구조**


```
_id: ObjectId('6272Oebda0dc9853dbcb7f7c')
RowNumber: 1
CustomerId: 15634602
Surname: "Hargrave"
CreditScore: 619
Geography: "France"
Gender: "Female"
Age: 42
Tenure: 2
Balance: 0
NumOfProducts: 1
HasCrCard: 1
IsActiveMember: 1
EstimatedSalary: 101348.88
Exited: 1
```


bank 데이터베이스 Collection 에 들어가는 Doc 의 구조는 RowNumber, CustomerId, Surname, CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary, Exited 로 구성되어있다.

* Document 는 key-value 값의 집합으로 이루어져 있다. 아래 예시를 통해 나타낼 수 있다.

RowNumber	레코드(행) 번호에 해당
CustomerId	임의의 값을 포함하는 고객 아이디
Surname	고객의 성

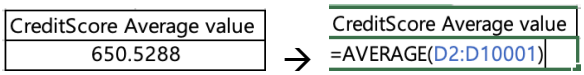
CreditScore	신용 점수
Geography	고객의 위치를 나타내는 지역
Gender	고객의 성별
Age	고객의 나이
Tenure	고객이 은행의 고객이었던 기간
Balance	계좌 잔고
NumOfProducts	고객이 은행을 통해 구매한 제품 수
HasCrCard	고객이 신용카드를 가지고 있는지 여부
IsActiveMember	활성 고객의 유무
EstimatedSalary	고객의 예상 급여
Exited	고객이 은행을 떠났는지 여부

■ Find/Update/Insert/Delete 을 이용한 쿼리 작성(각 1 가지, 총 4 개)

본 팀은 스페인, 독일, 프랑스의 각 나라별 신용 등급과 나이에 따른 남녀 소득차이와 고소득, 저소득 층을 구분해내어 고객별 맞춤 추천 상품을 제공하는 것을 목적으로 데이터를 분석했다.

▸ Find() 함수

먼저 find() 함수를 사용하여 국적이 스페인 남성이며, 신용등급이 650 점 이상인 사람 5 명을 추출하였다.
이 신용 등급은 average() 함수를 이용하여 평균이 650 임을 확인하고 진행하였다.



Find() 함수를 사용한 이유?
14 개의 columns 에서 우리가 찾고자 하는 정보는 지역, 성별, 신용등급 정보를 종합적으로 찾기 위해서 find() 함수를 사용했다.
그리고 $gte 를 이용해 신용등급 평균이 650 이상인 데이터를 추출해냈고, sort() 함수를 사용해서 나이를 내림차순으로 정렬했다.

<스페인 남자>

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.find({Geography:"Spain",Gend
er:"Male",CreditScore:{$gte:650}},{_id:0,CreditScore:1,Geography:1,Age:1,Gender:1,Estimat
edSalary:1}).sort({Age:-1}).pretty().limit(5)
{
        "CreditScore" : 809,
        "Geography" : "Spain",
        "Gender" : "Male",
        "Age" : 80,
        "EstimatedSalary" : 34164.05
}
{
        "CreditScore" : 659,
        "Geography" : "Spain",
        "Gender" : "Male",
        "Age" : 78,
        "EstimatedSalary" : 49978.67
}
{
        "CreditScore" : 767,
        "Geography" : "Spain",
        "Gender" : "Male",
        "Age" : 77,
        "EstimatedSalary" : 190146.83
}
{
        "CreditScore" : 667,
        "Geography" : "Spain",
        "Gender" : "Male",
        "Age" : 77,
        "EstimatedSalary" : 34702.92
}
{
        "CreditScore" : 657,
        "Geography" : "Spain",
        "Gender" : "Male",
        "Age" : 75,
        "EstimatedSalary" : 91673.6
}
```

<스페인 여자>

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.find({Geography:"Spain",Gend
er:"Female",CreditScore:{$gte:650}},{_id:0,CreditScore:1,Geography:1,Age:1,Gender:1,Estim
atedSalary:1}).sort({Age:-1}).pretty().limit(5)
{
        "CreditScore" : 712,
        "Geography" : "Spain",
        "Gender" : "Female",
        "Age" : 79,
        "EstimatedSalary" : 174118.93
}
{
        "CreditScore" : 655,
        "Geography" : "Spain",
        "Gender" : "Female",
        "Age" : 78,
        "EstimatedSalary" : 188435.38
}
{
        "CreditScore" : 652,
        "Geography" : "Spain",
        "Gender" : "Female",
        "Age" : 75,
        "EstimatedSalary" : 114675.75
}
{
        "CreditScore" : 710,
        "Geography" : "Spain",
        "Gender" : "Female",
        "Age" : 75,
        "EstimatedSalary" : 9376.89
}
{
        "CreditScore" : 759,
        "Geography" : "Spain",
        "Gender" : "Female",
        "Age" : 74,
        "EstimatedSalary" : 48244.64
}
```

<독일 여자>

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.find({Geography:"Germany",Ge
nder:"Female",CreditScore:{$gte:650}},{_id:0,CreditScore:1,Geography:1,Age:1,Gender:1,Est
imatedSalary:1}).sort({Age:-1}).pretty().limit(5)
{
        "CreditScore" : 775,
        "Geography" : "Germany",
        "Gender" : "Female",
        "Age" : 77,
        "EstimatedSalary" : 37836.64
}
{
        "CreditScore" : 673,
        "Geography" : "Germany",
        "Gender" : "Female",
        "Age" : 77,
        "EstimatedSalary" : 59595.66
}
{
        "CreditScore" : 719,
        "Geography" : "Germany",
        "Gender" : "Female",
        "Age" : 76,
        "EstimatedSalary" : 176244.87
}
{
        "CreditScore" : 775,
        "Geography" : "Germany",
        "Gender" : "Female",
        "Age" : 74,
        "EstimatedSalary" : 134869.93
}
{
        "CreditScore" : 697,
        "Geography" : "Germany",
        "Gender" : "Female",
        "Age" : 74,
        "EstimatedSalary" : 16445.79
}
```

<독일 남자>

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.find({Geography:"France",Gen
der:"Male",CreditScore:{$gte:650}},{_id:0,CreditScore:1,Geography:1,Age:1,Gender:1,Estima
tedSalary:1}).sort({Age:-1}).pretty().limit(5)
{
        "CreditScore" : 753,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 92,
        "EstimatedSalary" : 195563.99
}
{
        "CreditScore" : 705,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 92,
        "EstimatedSalary" : 34436.83
}
{
        "CreditScore" : 850,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 81,
        "EstimatedSalary" : 44827.47
}
{
        "CreditScore" : 732,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 79,
        "EstimatedSalary" : 104222.8
}
{
        "CreditScore" : 774,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 76,
        "EstimatedSalary" : 143133.18
}
```

<프랑스 남자>

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.find({Geography:"France",Gen
der:"Male",CreditScore:{$gte:650}},{_id:0,CreditScore:1,Geography:1,Age:1,Gender:1,Estima
tedSalary:1}).sort({Age:-1}).pretty().limit(5)
{
        "CreditScore" : 753,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 92,
        "EstimatedSalary" : 195563.99
}
{
        "CreditScore" : 705,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 92,
        "EstimatedSalary" : 34436.83
}
{
        "CreditScore" : 850,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 81,
        "EstimatedSalary" : 44827.47
}
{
        "CreditScore" : 732,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 79,
        "EstimatedSalary" : 104222.8
}
{
        "CreditScore" : 774,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 76,
        "EstimatedSalary" : 143133.18
}
```

<프랑스 여자>

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.find({Geography:"France",Gen
der:"Female",CreditScore:{$gte:650}},{_id:0,CreditScore:1,Geography:1,Age:1,Gender:1,Esti
matedSalary:1}).sort({Age:-1}).pretty().limit(5)
{
        "CreditScore" : 787,
        "Geography" : "France",
        "Gender" : "Female",
        "Age" : 85,
        "EstimatedSalary" : 116537.96
}
{
        "CreditScore" : 700,
        "Geography" : "France",
        "Gender" : "Female",
        "Age" : 82,
        "EstimatedSalary" : 182055.36
}
{
        "CreditScore" : 711,
        "Geography" : "France",
        "Gender" : "Female",
        "Age" : 81,
        "EstimatedSalary" : 72276.24
}
{
        "CreditScore" : 850,
        "Geography" : "France",
        "Gender" : "Female",
        "Age" : 81,
        "EstimatedSalary" : 59568.24
}
{
        "CreditScore" : 652,
        "Geography" : "France",
        "Gender" : "Female",
        "Age" : 80,
        "EstimatedSalary" : 188603.07
}
```

▸ Update() 함수

그 후 update() 함수를 이용하여 예상 급여(EstimatedSalary) 에 스페인, 독일, 프랑스의 화폐 단위인 유로(EUR) 를 추가해주기 위해 rename() 함수를 사용하여 EstimatedSalary-> EstimatedSalary(EUR)으로 이름을 재설정해주었다.

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.updateMany({},{$rename:{"EstimatedSalary":"EstimatedSalary(EUR)"}})
{ "acknowledged" : true, "matchedCount" : 10000, "modifiedCount" : 9999 }
```

<독일 유로 표시>


```
{
        "_id" : ObjectId("62720ebda0dc9853dbcb7f83"),
        "RowNumber" : 8,
        "CustomerId" : 15656148,
        "Surname" : "Obinna",
        "CreditScore" : 376,
        "Geography" : "Germany",
        "Gender" : "Female",
        "Age" : 29,
        "Tenure" : 4,
        "Balance" : 115046.74,
        "NumOfProducts" : 4,
        "HasCrCard" : 1,
        "IsActiveMember" : 0,
        "Exited" : 1,
        "EstimatedSalary(EUR)" : 119346.88
}
```


<스페인 유로 표시>


```
{
        "_id" : ObjectId("62720ebda0dc9853dbcb7f7d"),
        "RowNumber" : 2,
        "CustomerId" : 15647311,
        "Surname" : "Hill",
        "CreditScore" : 608,
        "Geography" : "Spain",
        "Gender" : "Female",
        "Age" : 41,
        "Tenure" : 1,
        "Balance" : 83807.86,
        "NumOfProducts" : 1,
        "HasCrCard" : 0,
        "IsActiveMember" : 1,
        "Exited" : 0,
        "EstimatedSalary(EUR)" : 112542.58
}
```


<프랑스 유로 표시>


```
{
        "_id" : ObjectId("62720ebda0dc9853dbcb7f7c"),
        "RowNumber" : 1,
        "CustomerId" : 15634602,
        "Surname" : "Hargrave",
        "CreditScore" : 619,
        "Geography" : "France",
        "Gender" : "Female",
        "Age" : 42,
        "Tenure" : 2,
        "Balance" : 0,
        "NumOfProducts" : 1,
        "HasCrCard" : 1,
        "IsActiveMember" : 1,
        "Exited" : 1,
        "EstimatedSalary(EUR)" : 101348.88
}
```

▸ update + insert () 함수

Upsert() 함수를 이용해 원래 없던 은행명 정보를 $set 을 사용해서 Bank_name: "BNP_bank" 를 추가하였다.

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.updateMany({},{$set:{Bank_name:"BNP_bank"}},{upsert:true})
{ "acknowledged" : true, "matchedCount" : 10000, "modifiedCount" : 10000 }
```


```
{
        "_id" : ObjectId("6273bc39bd576237373928b9"),
        "RowNumber" : 17,
        "CustomerId" : 15737452,
        "Surname" : "Romeo",
        "CreditScore" : 653,
        "Geography" : "Germany",
        "Gender" : "Male",
        "Age" : 58,
        "Tenure" : 1,
        "Balance" : 132602.88,
        "NumOfProducts" : 1,
        "HasCrCard" : 1,
        "IsActiveMember" : 0,
        "EstimatedSalary" : 5097.67,
        "Exited" : 1,
        "Bank_name" : "BNP_bank"
}
{
        "_id" : ObjectId("6273bc39bd576237373928ba"),
        "RowNumber" : 18,
        "CustomerId" : 15788218,
        "Surname" : "Henderson",
        "CreditScore" : 549,
        "Geography" : "Spain",
        "Gender" : "Female",
        "Age" : 24,
        "Tenure" : 9,
        "Balance" : 0,
        "NumOfProducts" : 2,
        "HasCrCard" : 1,
        "IsActiveMember" : 1,
        "EstimatedSalary" : 14406.41,
        "Exited" : 0,
        "Bank_name" : "BNP_bank"
```


▸ Delete() 함수

계좌 잔고 (Balance) 하위 10% 인 저소득층에게 필요한 금융상품을 소개 시켜주기 위한 저소득층 데이터만 뽑아 금융판매원에게 넘겨주는 목적을 가지고 데이터를 선별해 삭제해주었다.

우선 저소득층을 확인하기 위해 Delete() 함수를 이용해 계좌 잔고(Balance) 가 0 인 사람들을 제거해주었고, 3617 개의 데이터가 삭제된 것을 확인할 수 있다.

```
MongoDB Enterprise atlas-amenlz-shard-0:PRIMARY> db.bank1.deleteMany({Balance:0})
{ "acknowledged" : true, "deletedCount" : 3617 }
```

그 후 계좌 잔고 하위 10% 의 평균을 구해보니 42851 유로가 나와 $gte 를 사용해서 42851 유로 이상의 데이터는 다 삭제를 해주어 6343 개의 데이터가 삭제되어 저소득층 데이터를 완성시켰다.

Balance Bottom 10%
42805.18495

```
MongoDB Enterprise atlas-amenlz-shard-0:PRIMARY> db.bank1.deleteMany({Balance:{$gte:42851}})
{ "acknowledged" : true, "deletedCount" : 6343 }
```

<저소득층 결과값>

```
MongoDB Enterprise atlas-amenlz-shard-0:PRIMARY> db.bank1.find({Balance:{$lte:42851}},{_id:0,Balance:1,CustomerId:1}).sort({Balance:-1})
.pretty()
{ "CustomerId" : 15665008, "Balance" : 42712.87 }
{ "CustomerId" : 15569807, "Balance" : 42157.08 }
{ "CustomerId" : 15740345, "Balance" : 41473.33 }
{ "CustomerId" : 15734044, "Balance" : 41299.03 }
{ "CustomerId" : 15612350, "Balance" : 40915.55 }
{ "CustomerId" : 15804771, "Balance" : 40685.92 }
{ "CustomerId" : 15576124, "Balance" : 40488.76 }
{ "CustomerId" : 15624428, "Balance" : 40224.7 }
{ "CustomerId" : 15672875, "Balance" : 40172.91 }
{ "CustomerId" : 15682834, "Balance" : 40169.88 }
{ "CustomerId" : 15684042, "Balance" : 40105.51 }
{ "CustomerId" : 15754929, "Balance" : 39539.39 }
{ "CustomerId" : 15611186, "Balance" : 39344.83 }
{ "CustomerId" : 15763662, "Balance" : 39043.29 }
{ "CustomerId" : 15602811, "Balance" : 38848.19 }
{ "CustomerId" : 15679394, "Balance" : 38617.2 }
{ "CustomerId" : 15752622, "Balance" : 38550.06 }
{ "CustomerId" : 15773283, "Balance" : 38340.02 }
{ "CustomerId" : 15743976, "Balance" : 37702.79 }
{ "CustomerId" : 15592979, "Balance" : 37266.67 }
Type "it" for more
```

그리고 계좌 잔고(Balance) 상위 10% 인 고소득층에게 필요한 금융상품을 소개 시켜주기 위한 고소득층 데이터만 뽑아 금융판매원에게 넘겨주는 목적을 가지고 데이터를 선별해 삭제해주었다.

새로운 데이터를 찾기 위해 우선 삭제된 데이터를 복구시킨 뒤 원래 데이터를 몽고 compass 에 재 연결하여 다시 진행하였다.

그 후 계좌 잔고 상위 10% 의 평균을 구해보니 197584 유로가 나와 $lte 를 사용해서 197584 유로 이하의 데이터는 다 삭제를 해주었고 6343 개의 데이터가 삭제되어 고소득층 데이터를 완성시켰다.

Balance Top 10%
197584.8438

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.deleteMany({Balance:{$lte:197584}})
{ "acknowledged" : true, "deletedCount" : 6343 }
```

그렇게 find() 함수를 사용해 _id 는 삭제하고, CustomerId(고객 아이디)와 Balance(계좌 잔고)를 내림차순으로 데이터를 출력을 해보았다.

<고소득층 결과값>

```
MongoDB Enterprise atlas-1boriu-shard-0:PRIMARY> db.bankname.find({},{_id:0, CustomerId:1, Balance:1}).sort({Balance:-1}
).limit(40)
{ "CustomerId" : 15757408, "Balance" : 250898.09 }
{ "CustomerId" : 15715622, "Balance" : 238387.56 }
{ "CustomerId" : 15714241, "Balance" : 222267.63 }
{ "CustomerId" : 15571958, "Balance" : 221532.8 }
{ "CustomerId" : 15586674, "Balance" : 216109.88 }
{ "CustomerId" : 15599131, "Balance" : 214346.96 }
{ "CustomerId" : 15594408, "Balance" : 213146.2 }
{ "CustomerId" : 15769818, "Balance" : 212778.2 }
{ "CustomerId" : 15620268, "Balance" : 212696.32 }
{ "CustomerId" : 15780212, "Balance" : 212692.97 }
{ "CustomerId" : 15690589, "Balance" : 212314.03 }
{ "CustomerId" : 15671256, "Balance" : 211774.31 }
{ "CustomerId" : 15736420, "Balance" : 210433.08 }
{ "CustomerId" : 15721658, "Balance" : 209767.31 }
{ "CustomerId" : 15578671, "Balance" : 209490.21 }
{ "CustomerId" : 15709920, "Balance" : 208165.53 }
{ "CustomerId" : 15769412, "Balance" : 207034.96 }
{ "CustomerId" : 15795298, "Balance" : 206868.78 }
{ "CustomerId" : 15627971, "Balance" : 206663.75 }
{ "CustomerId" : 15784180, "Balance" : 206329.65 }
```

■ Aggregation 을 이용한 분석 프로세스 설명 (2 가지)

40~50 대가 고객 예상 급여가 가장 많다는 통계를 통해(상위 1% 부자의 평균연령이 48.8 세였던 점) 각 나라별 40~50 대의 예상 급여를 aggregate() 함수를 사용하여 추출해냈다.

-> 40~50 대가 예상 급여가 가장 많다고 생각한 이유는 퇴사 직전이 20~30 대에 비해 고연봉 직급을 가지고 있다고 예상했기 때문이다.

▸ Aggregate() 함수

$gte 와 $lte 를 사용해서 40~59 사이의 연령대의 정보를 가져왔다. 그 다음 $sort 를 사용해 예상급여(EstimatedSalary(EUR))를 내림차순 정렬해주었다.

<프랑스>

```
MongoDB Enterprise atlas-amenlz-shard-0:PRIMARY> db.bank1.aggregate([{$match:{Age:{$gte:40,$lte:59},Geography:"France"}},{$sort:{"EstimatedSalary(EUR)":-1}
},{$project:{Geography:1,_id:0,"EstimatedSalary(EUR)":1,Age:1,CustomerId:1}}])
{ "CustomerId" : 15741719, "Geography" : "France", "Age" : 40, "EstimatedSalary(EUR)" : 199862.75 }
{ "CustomerId" : 15782758, "Geography" : "France", "Age" : 40, "EstimatedSalary(EUR)" : 199674.83 }
{ "CustomerId" : 15814040, "Geography" : "France", "Age" : 45, "EstimatedSalary(EUR)" : 199657.46 }
{ "CustomerId" : 15672798, "Geography" : "France", "Age" : 45, "EstimatedSalary(EUR)" : 199392.14 }
{ "CustomerId" : 15661903, "Geography" : "France", "Age" : 43, "EstimatedSalary(EUR)" : 199378.58 }
{ "CustomerId" : 15792102, "Geography" : "France", "Age" : 42, "EstimatedSalary(EUR)" : 199316.19 }
{ "CustomerId" : 15746326, "Geography" : "France", "Age" : 43, "EstimatedSalary(EUR)" : 198926.36 }
{ "CustomerId" : 15754577, "Geography" : "France", "Age" : 51, "EstimatedSalary(EUR)" : 198810.65 }
{ "CustomerId" : 15749381, "Geography" : "France", "Age" : 41, "EstimatedSalary(EUR)" : 198224.38 }
{ "CustomerId" : 15715541, "Geography" : "France", "Age" : 42, "EstimatedSalary(EUR)" : 198193.75 }
{ "CustomerId" : 15579616, "Geography" : "France", "Age" : 42, "EstimatedSalary(EUR)" : 198134.9 }
{ "CustomerId" : 15576745, "Geography" : "France", "Age" : 48, "EstimatedSalary(EUR)" : 197885.72 }
{ "CustomerId" : 15811415, "Geography" : "France", "Age" : 44, "EstimatedSalary(EUR)" : 197572.41 }
{ "CustomerId" : 15652289, "Geography" : "France", "Age" : 47, "EstimatedSalary(EUR)" : 197528.62 }
{ "CustomerId" : 15812888, "Geography" : "France", "Age" : 41, "EstimatedSalary(EUR)" : 197490.39 }
{ "CustomerId" : 15660263, "Geography" : "France", "Age" : 40, "EstimatedSalary(EUR)" : 197372.13 }
{ "CustomerId" : 15588918, "Geography" : "France", "Age" : 42, "EstimatedSalary(EUR)" : 197202.48 }
{ "CustomerId" : 15777784, "Geography" : "France", "Age" : 44, "EstimatedSalary(EUR)" : 197193.49 }
{ "CustomerId" : 15761487, "Geography" : "France", "Age" : 55, "EstimatedSalary(EUR)" : 196794.11 }
{ "CustomerId" : 15605293, "Geography" : "France", "Age" : 43, "EstimatedSalary(EUR)" : 196645.87 }
Type "it" for more
```

<독일>

```
MongoDB Enterprise atlas-amenlz-shard-0:PRIMARY> db.bank1.aggregate([{$match:{Age:{$gte:40,$lte:59},Geography:"Germany"}},{$sort:{"EstimatedSalary(EUR)":-1
},{$project:{Geography:1,_id:0,"EstimatedSalary(EUR)":1,Age:1,CustomerId:1}}])
{ "CustomerId" : 15634359, "Geography" : "Germany", "Age" : 41, "EstimatedSalary(EUR)" : 199970.74 }
{ "CustomerId" : 15772601, "Geography" : "Germany", "Age" : 41, "EstimatedSalary(EUR)" : 199761.29 }
{ "CustomerId" : 15698474, "Geography" : "Germany", "Age" : 54, "EstimatedSalary(EUR)" : 199661.5 }
{ "CustomerId" : 15743040, "Geography" : "Germany", "Age" : 41, "EstimatedSalary(EUR)" : 199645.45 }
{ "CustomerId" : 15707362, "Geography" : "Germany", "Age" : 43, "EstimatedSalary(EUR)" : 199273.98 }
{ "CustomerId" : 15585100, "Geography" : "Germany", "Age" : 40, "EstimatedSalary(EUR)" : 198814.24 }
{ "CustomerId" : 15662751, "Geography" : "Germany", "Age" : 40, "EstimatedSalary(EUR)" : 198798.44 }
{ "CustomerId" : 15583076, "Geography" : "Germany", "Age" : 41, "EstimatedSalary(EUR)" : 198766.61 }
{ "CustomerId" : 15616213, "Geography" : "Germany", "Age" : 51, "EstimatedSalary(EUR)" : 198715.27 }
{ "CustomerId" : 15750466, "Geography" : "Germany", "Age" : 42, "EstimatedSalary(EUR)" : 198182.73 }
{ "CustomerId" : 15760085, "Geography" : "Germany", "Age" : 48, "EstimatedSalary(EUR)" : 198129.96 }
{ "CustomerId" : 15650432, "Geography" : "Germany", "Age" : 41, "EstimatedSalary(EUR)" : 198072.16 }
{ "CustomerId" : 15713621, "Geography" : "Germany", "Age" : 41, "EstimatedSalary(EUR)" : 198064.52 }
{ "CustomerId" : 15616550, "Geography" : "Germany", "Age" : 44, "EstimatedSalary(EUR)" : 198059.16 }
{ "CustomerId" : 15652048, "Geography" : "Germany", "Age" : 44, "EstimatedSalary(EUR)" : 197812.16 }
{ "CustomerId" : 15750811, "Geography" : "Germany", "Age" : 44, "EstimatedSalary(EUR)" : 197643.24 }
{ "CustomerId" : 15666548, "Geography" : "Germany", "Age" : 56, "EstimatedSalary(EUR)" : 197634.11 }
{ "CustomerId" : 15724466, "Geography" : "Germany", "Age" : 41, "EstimatedSalary(EUR)" : 197548.63 }
{ "CustomerId" : 15743617, "Geography" : "Germany", "Age" : 47, "EstimatedSalary(EUR)" : 197529.23 }
{ "CustomerId" : 15770995, "Geography" : "Germany", "Age" : 47, "EstimatedSalary(EUR)" : 197444.69 }
```

<스페인>

```
MongoDB Enterprise atlas-amenlz-shard-0:PRIMARY> db.bank1.aggregate([{$match:{Age:{$gte:40,$lte:59},Geography:"Spain"}},{$sort:{"EstimatedSalary(EUR)":-1}}
,{$project:{Geography:1,_id:0,"EstimatedSalary(EUR)":1,Age:1,CustomerId:1}}])
{ "CustomerId" : 15662021, "Geography" : "Spain", "Age" : 42, "EstimatedSalary(EUR)" : 199992.48 }
{ "CustomerId" : 15763065, "Geography" : "Spain", "Age" : 40, "EstimatedSalary(EUR)" : 199753.97 }
{ "CustomerId" : 15585961, "Geography" : "Spain", "Age" : 43, "EstimatedSalary(EUR)" : 199505.53 }
{ "CustomerId" : 15755262, "Geography" : "Spain", "Age" : 41, "EstimatedSalary(EUR)" : 199304.74 }
{ "CustomerId" : 15624641, "Geography" : "Spain", "Age" : 43, "EstimatedSalary(EUR)" : 199290.68 }
{ "CustomerId" : 15765415, "Geography" : "Spain", "Age" : 45, "EstimatedSalary(EUR)" : 199256.98 }
{ "CustomerId" : 15800736, "Geography" : "Spain", "Age" : 42, "EstimatedSalary(EUR)" : 199242.65 }
{ "CustomerId" : 15648951, "Geography" : "Spain", "Age" : 41, "EstimatedSalary(EUR)" : 199108.88 }
{ "CustomerId" : 15671591, "Geography" : "Spain", "Age" : 52, "EstimatedSalary(EUR)" : 198874.52 }
{ "CustomerId" : 15790809, "Geography" : "Spain", "Age" : 40, "EstimatedSalary(EUR)" : 198694.2 }
{ "CustomerId" : 15780989, "Geography" : "Spain", "Age" : 43, "EstimatedSalary(EUR)" : 198402.37 }
{ "CustomerId" : 15621768, "Geography" : "Spain", "Age" : 45, "EstimatedSalary(EUR)" : 198398.68 }
{ "CustomerId" : 15690440, "Geography" : "Spain", "Age" : 47, "EstimatedSalary(EUR)" : 197961.93 }
{ "CustomerId" : 15681506, "Geography" : "Spain", "Age" : 43, "EstimatedSalary(EUR)" : 197916.43 }
{ "CustomerId" : 15801265, "Geography" : "Spain", "Age" : 45, "EstimatedSalary(EUR)" : 197804 }
{ "CustomerId" : 15596455, "Geography" : "Spain", "Age" : 45, "EstimatedSalary(EUR)" : 197789.83 }
{ "CustomerId" : 15814998, "Geography" : "Spain", "Age" : 42, "EstimatedSalary(EUR)" : 197602.29 }
{ "CustomerId" : 15668629, "Geography" : "Spain", "Age" : 44, "EstimatedSalary(EUR)" : 196582.19 }
{ "CustomerId" : 15588537, "Geography" : "Spain", "Age" : 41, "EstimatedSalary(EUR)" : 196499.96 }
{ "CustomerId" : 15693690, "Geography" : "Spain", "Age" : 52, "EstimatedSalary(EUR)" : 196257.67 }
Type "it" for more
```

▸ Aggregate() - group() 함수

Group() 함수를 사용해 지역과 신용등급의 평균을 그룹화 해주고 $avg 를 사용해서 avg_CreditScore 에 평균정보를 저장한 뒤 내림차순으로 정렬해주었다.

```
MongoDB Enterprise atlas-amenlz-shard-0:PRIMARY> db.bank1.aggregate([{$group:{_id:{"Geography":"$Geography"},avg_CreditScore:{$avg:"$CreditScore"}}},{$sor
t:{avg_CreditScore:-1}}])
{ "_id" : { "Geography" : "Germany" }, "avg_CreditScore" : 651.4535671582304 }
{ "_id" : { "Geography" : "Spain" }, "avg_CreditScore" : 651.3338716188938 }
{ "_id" : { "Geography" : "France" }, "avg_CreditScore" : 649.6683286796969 }
MongoDB Enterprise atlas-amenlz-shard-0:PRIMARY>
```

■RDBMS 와 비교했을 때, 대상 Data 를 MongoDB 로 구현할 때 얻는 장단점

위의 은행 데이터를 RDBMS 로 구현할 때 데이터베이스가 많고, 파일이 많아서 우리가 원하는 데이터를 얻기 위해서는 시간이 많이 소모되고 또한 처리속도가 느려 응답성이 떨어진다. 그렇기에 MongoDB 를 사용하게 되면 이러한 문제점들이 해소가 된다. 위의 은행 데이터를 관리하면서 중복 데이터 및 수정해야 할 정보들이 생기는데 RDBMS 는 중복된 데이터를 한번만 저장할 수 있는 무결성 데이터인 반면에 MongoDB 는 DOC 마다 _id 값이 다르기 때문에 여러번 저장할 수 있고 CRUD(Create, Read, Update, Delete)에 유용하다.

하지만 MongoDB 는 데이터 중복이 발생할 수 있어서 만약에 중복된 데이터가 변경이 될 경우 수정을 할 시에는 데이터의 수정을 모든 컬렉션에서 수행해야 한다는 단점이 있다. 또한 스키마가 존재하지 않기 때문에 명확한 데이터 구조를 보장하지 않으며, 데이터 구조를 결정하기 어려울 수 있다.

본 팀은 은행데이터가 DOC 의 _id 값 마다 다른 데이터를 갖고 있기 때문에 여러 번 저장하고, 우리가 원하는 데이터를 얻기 위해 여러 가지 함수를 이용하여 은행의 성별과 국적별 신용등급과 같은 데이터들을 찾음과 더불어 필드의 이름을 수정하고 싶었다. 그리고 현재 필드에 없는 데이터를 삽입 시키거나 우리가 원하는 데이터만을 추출해 내기 위해 필요 없는 데이터를 삭제시켰다. 그렇기 때문에 CRUD 에 효율적이고 빠르게 작동하는 MongoDB 를 사용하는 것이 적합하다고 판단이 되어 최종적으로 MongoDB 를 이용해 분석을 진행하였다.