CSE 414 DATABASE

SPRING 2024

HOMEWORK 1

MERT GÜRŞİMŞİR

1901042646

DESIGN

For the building a NoSQL database system for banking operations, I have used MongoDB database technology.

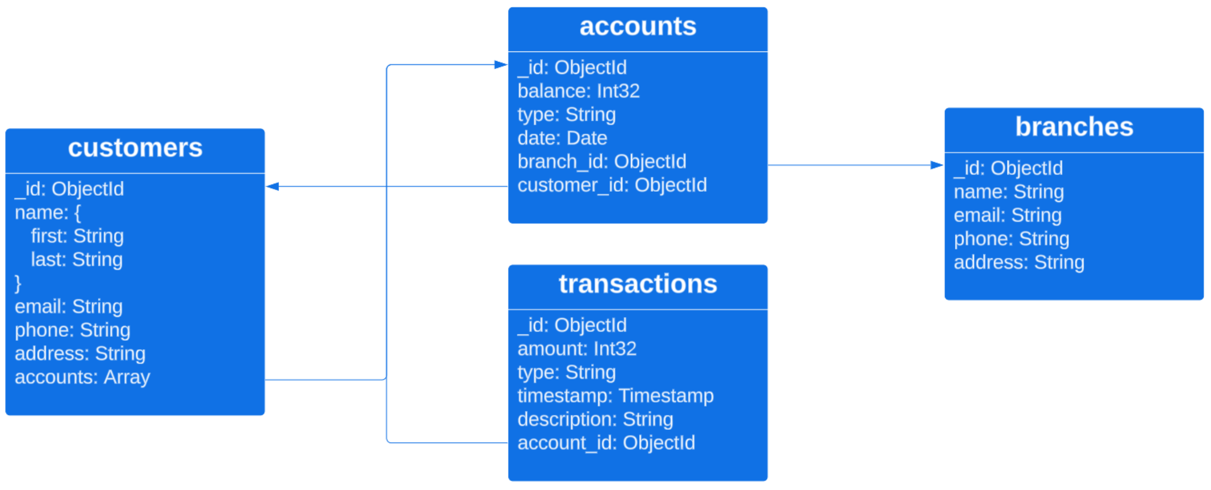
For the banking operations I have 4 collections:

- customers
- accounts
- transactions
- branches

I am keeping accounts' id values that customers has as an array in the customers collection. Also I am keeping associated customer's id value in the accounts collection to easily access customer, to know who owns the account. Each customer can have several accounts but each account has only 1 user. I am also keeping branches' id values in the accounts collection to know which branch each account belongs to. Different accounts can have same branch but an account can have only 1 branch. I am keeping accounts' id values in the transactions to know that transaction belongs to which account. Accounts can have more than 1 transactions but transactions can have only 1 account.

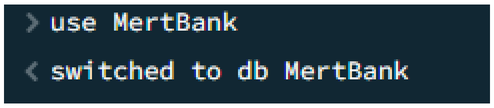
Advantage of keeping the id values is it reduces the data duplication. We can access the to other collections by using the id values.

In order to visualize the paragraph, I am representing my schema design below:



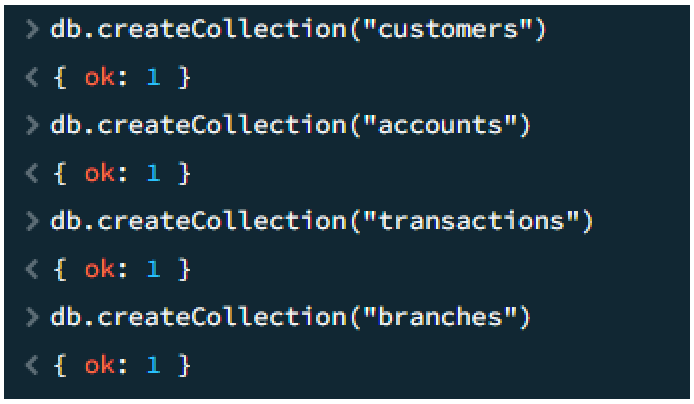
These schemes provide flexibility and scalability. For example, if a new requirement arises in the future, we can easily add new fields or change the existing structure.

Creating the database:



```
> use MertBank
< switched to db MertBank
```

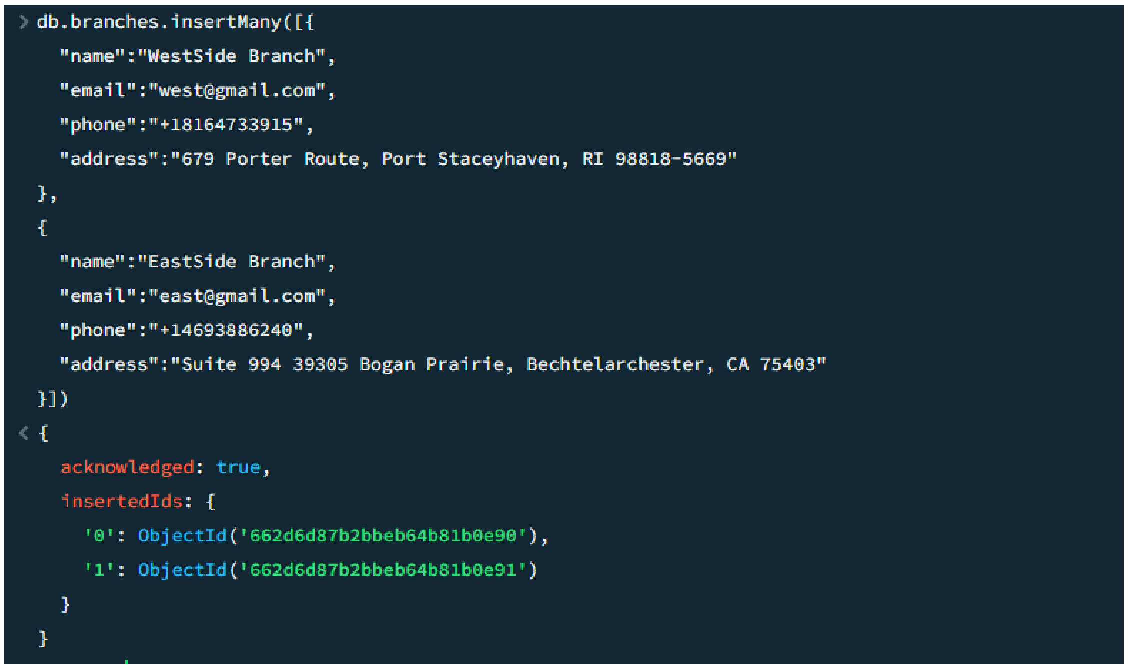
Creating collections:



```
> db.createCollection("customers")
< { ok: 1 }
> db.createCollection("accounts")
< { ok: 1 }
> db.createCollection("transactions")
< { ok: 1 }
> db.createCollection("branches")
< { ok: 1 }
```

BRANCHES

Inserting Branches:

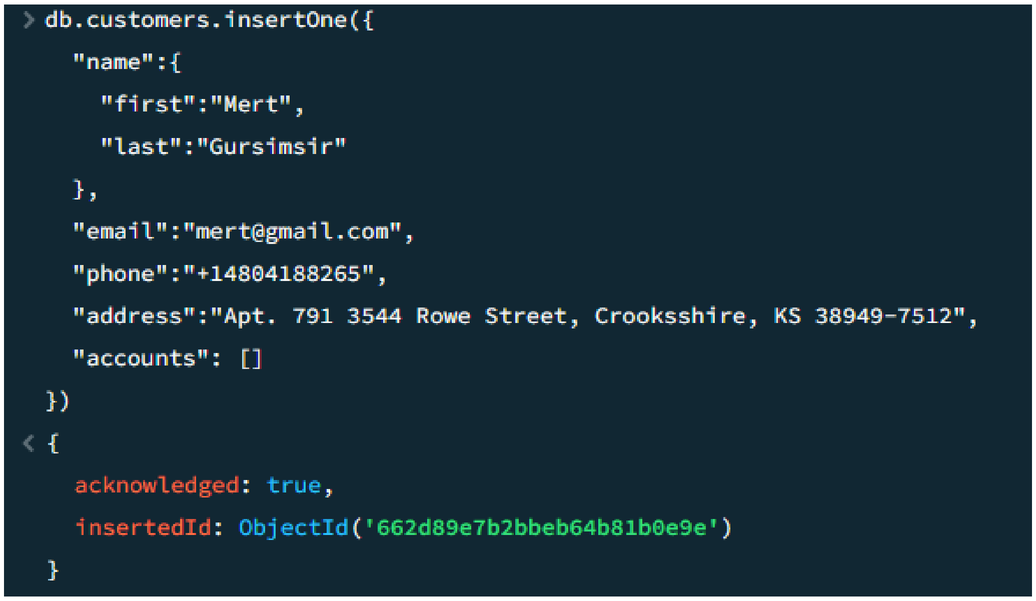


```
> db.branches.insertMany([{
    "name":"WestSide Branch",
    "email":"west@gmail.com",
    "phone":"+18164733915",
    "address":"679 Porter Route, Port Staceyhaven, RI 98818-5669"
  },
  {
    "name":"EastSide Branch",
    "email":"east@gmail.com",
    "phone":"+14693886240",
    "address":"Suite 994 39305 Bogan Prairie, Bechtelarchester, CA 75403"
  }])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('662d6d87b2bbeb64b81b0e90'),
      '1': ObjectId('662d6d87b2bbeb64b81b0e91')
    }
  }
```

I have used insertMany to insert 2 branches at once. I have given fields and values for 2 branches.

CUSTOMERS

Inserting a customer:



```
> db.customers.insertOne({
    "name":{
      "first":"Mert",
      "last":"Gursimsir"
    },
    "email":"mert@gmail.com",
    "phone":"+14804188265",
    "address":"Apt. 791 3544 Rowe Street, Crooksshire, KS 38949-7512",
    "accounts": []
  })
< {
    acknowledged: true,
    insertedId: ObjectId('662d89e7b2bbeb64b81b0e9e')
  }
```

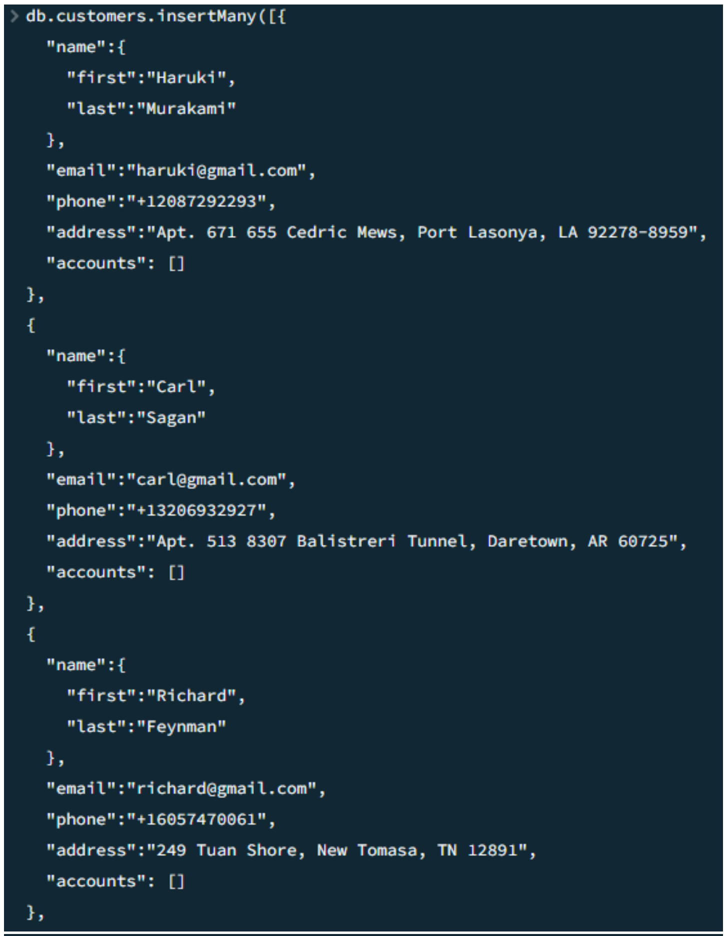
Here, I am inserting 1 customer with specified fields. I have not given any id value for the customer because it is automatically assigned.

name is an object that keeps first and last values that specify the first name and last name.

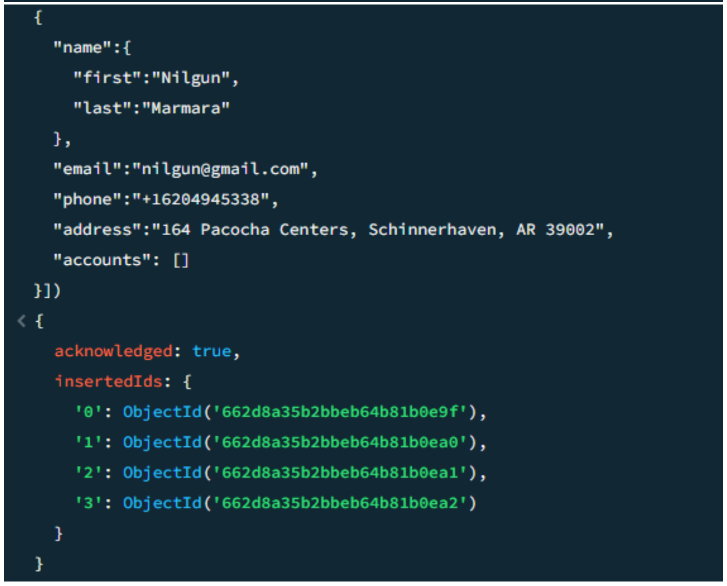
email, phone, address fields are all string.

accounts field has an array to keep accounts id values that user has. Initially the array is empty because user doesn't have any account at the beginning.

Inserting many customers:



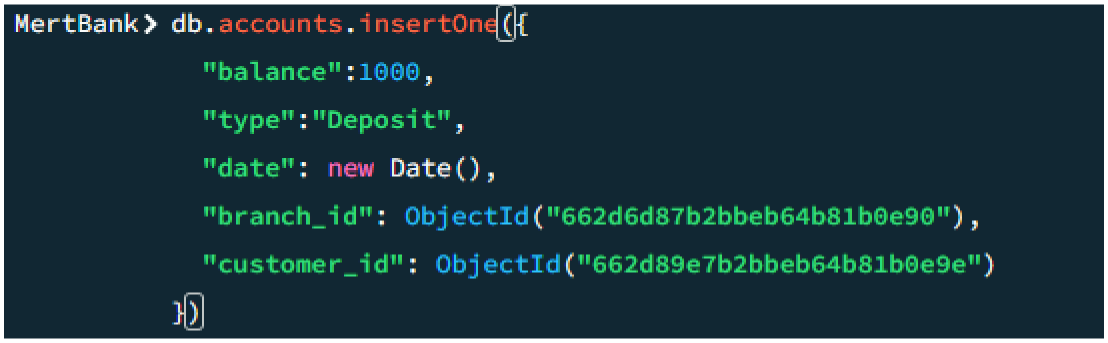
```
> db.customers.insertMany([{
    "name":{
      "first":"Haruki",
      "last":"Murakami"
    },
    "email":"haruki@gmail.com",
    "phone":"+12087292293",
    "address":"Apt. 671 655 Cedric Mews, Port Lasonya, LA 92278-8959",
    "accounts": []
  },
  {
    "name":{
      "first":"Carl",
      "last":"Sagan"
    },
    "email":"carl@gmail.com",
    "phone":"+13206932927",
    "address":"Apt. 513 8307 Balistreri Tunnel, Daretown, AR 60725",
    "accounts": []
  },
  {
    "name":{
      "first":"Richard",
      "last":"Feynman"
    },
    "email":"richard@gmail.com",
    "phone":"+16057470061",
    "address":"249 Tuan Shore, New Tomasa, TN 12891",
    "accounts": []
  },
```



```
  {
    "name":{
      "first":"Nilgun",
      "last":"Marmara"
    },
    "email":"nilgun@gmail.com",
    "phone":"+16204945338",
    "address":"164 Pacocha Centers, Schinnerhaven, AR 39002",
    "accounts": []
  }])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('662d8a35b2bbeb64b81b0e9f'),
      '1': ObjectId('662d8a35b2bbeb64b81b0ea0'),
      '2': ObjectId('662d8a35b2bbeb64b81b0ea1'),
      '3': ObjectId('662d8a35b2bbeb64b81b0ea2')
    }
  }
```

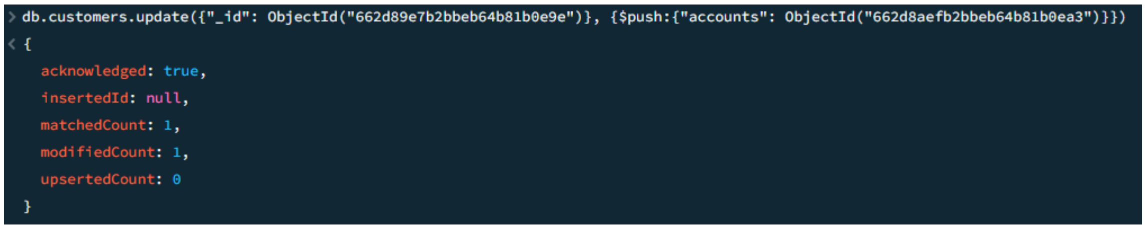
ACCOUNTS

Inserting a new account for user Mert (for WestSide branch):



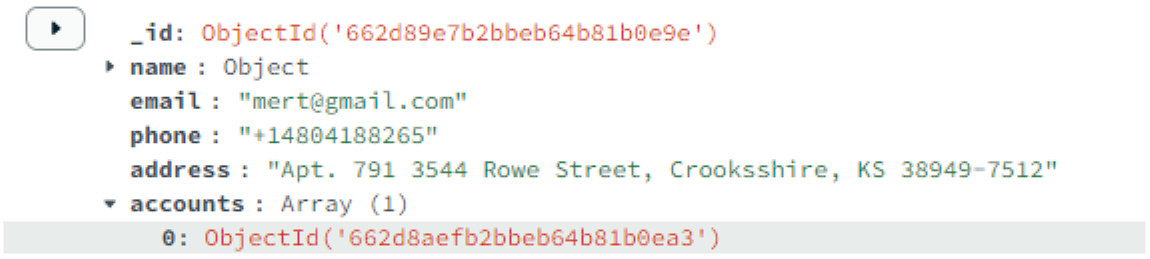
```
MertBank> db.accounts.insertOne({
            "balance":1000,
            "type":"Deposit",
            "date": new Date(),
            "branch_id": ObjectId("662d6d87b2bbeb64b81b0e90"),
            "customer_id": ObjectId("662d89e7b2bbeb64b81b0e9e")
          })
```

Now since a new account is added to customer with ID "662d89e7b2bbeb64b81b0e9e", we need to add this account to that customer's accounts field.



```
> db.customers.update({"_id": ObjectId("662d89e7b2bbeb64b81b0e9e")}, {$push:{"accounts": ObjectId("662d8aefb2bbeb64b81b0ea3")}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

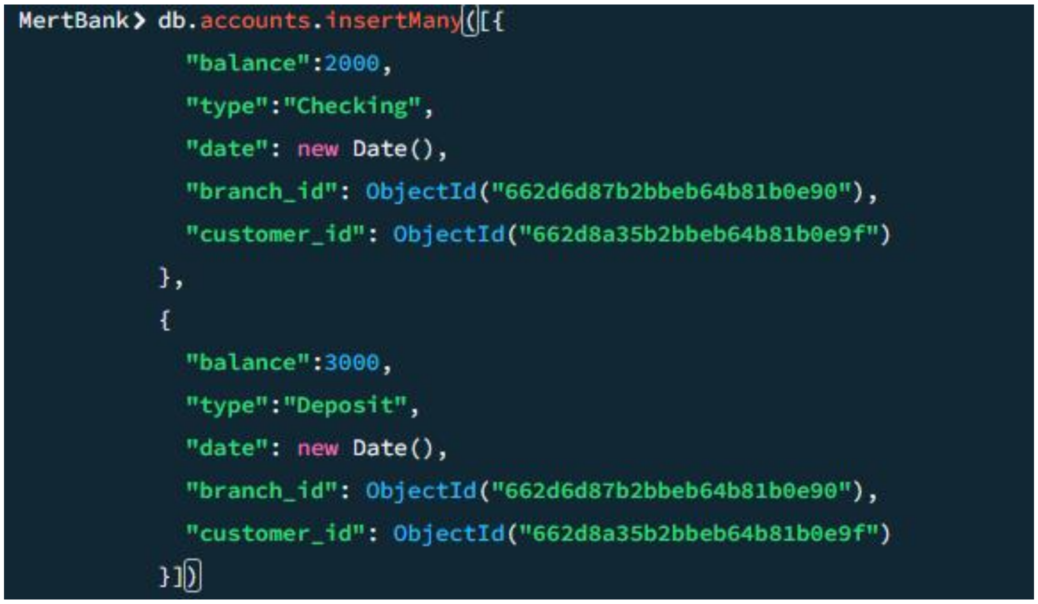
Now customer looks like this:



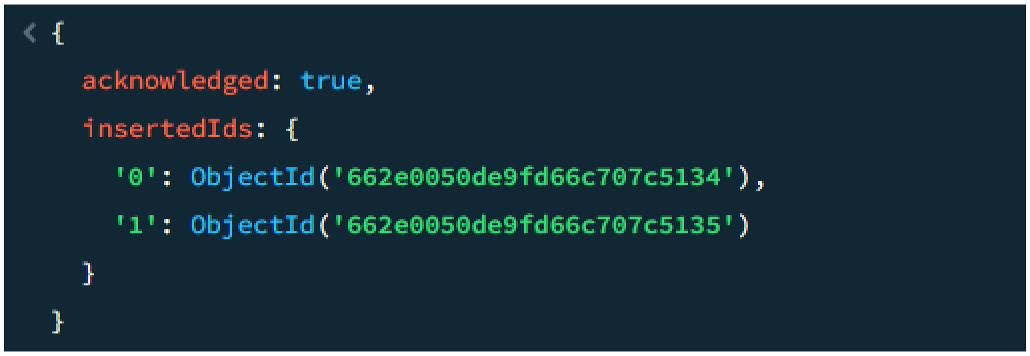
```
_id: ObjectId('662d89e7b2bbeb64b81b0e9e')
▸ name : Object
  email : "mert@gmail.com"
  phone : "+14804188265"
  address : "Apt. 791 3544 Rowe Street, Crooksshire, KS 38949-7512"
▾ accounts : Array (1)
    0: ObjectId('662d8aefb2bbeb64b81b0ea3')
```

I am keeping accounts as array to be flexible. So when a new account is going to be added to the customer, its id is going to be simply pushed to that array.

Insert 2 accounts for user Haruki Murakami (for WestSide branch):

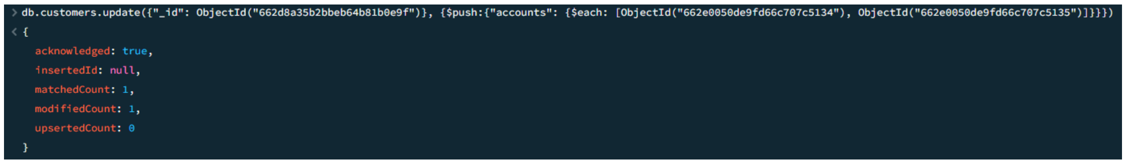


```
MertBank> db.accounts.insertMany([{
            "balance":2000,
            "type":"Checking",
            "date": new Date(),
            "branch_id": ObjectId("662d6d87b2bbeb64b81b0e90"),
            "customer_id": ObjectId("662d8a35b2bbeb64b81b0e9f")
          },
          {
            "balance":3000,
            "type":"Deposit",
            "date": new Date(),
            "branch_id": ObjectId("662d6d87b2bbeb64b81b0e90"),
            "customer_id": ObjectId("662d8a35b2bbeb64b81b0e9f")
          }])
```



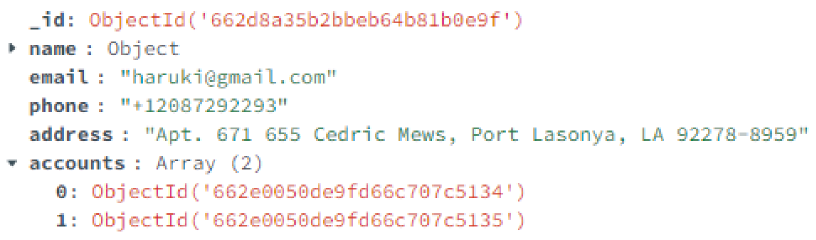
```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('662e0050de9fd66c707c5134'),
      '1': ObjectId('662e0050de9fd66c707c5135')
    }
  }
```

Updating Haruki’s accounts field:



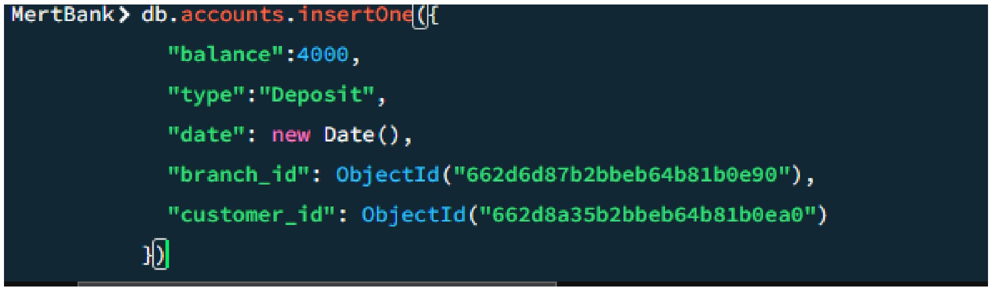
```
> db.customers.update({"_id": ObjectId("662d8a35b2bbeb64b81b0e9f")}, {$push:{"accounts": {$each: [ObjectId("662e0050de9fd66c707c5134"), ObjectId("662e0050de9fd66c707c5135")]}}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Haruki now looks like this:

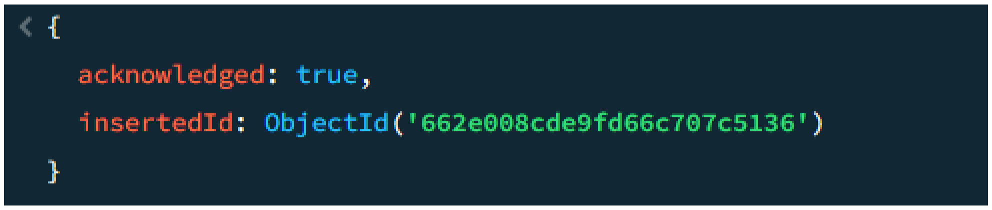


```
  _id: ObjectId('662d8a35b2bbeb64b81b0e9f')
▸ name : Object
  email : "haruki@gmail.com"
  phone : "+12087292293"
  address : "Apt. 671 655 Cedric Mews, Port Lasonya, LA 92278-8959"
▾ accounts : Array (2)
    0: ObjectId('662e0050de9fd66c707c5134')
    1: ObjectId('662e0050de9fd66c707c5135')
```

Insert an account for user Carl Sagan (for WestSide branch):

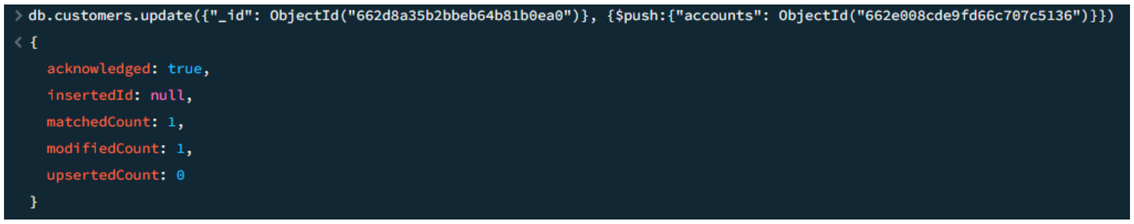


```
MertBank> db.accounts.insertOne({
            "balance":4000,
            "type":"Deposit",
            "date": new Date(),
            "branch_id": ObjectId("662d6d87b2bbeb64b81b0e90"),
            "customer_id": ObjectId("662d8a35b2bbeb64b81b0ea0")
          })
```



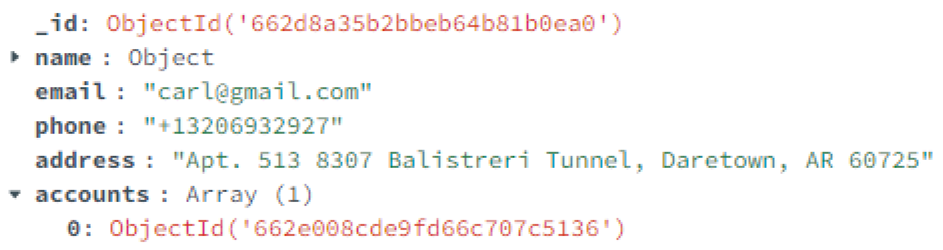
```
< {
    acknowledged: true,
    insertedId: ObjectId('662e008cde9fd66c707c5136')
  }
```

Updating Carl's accounts field:



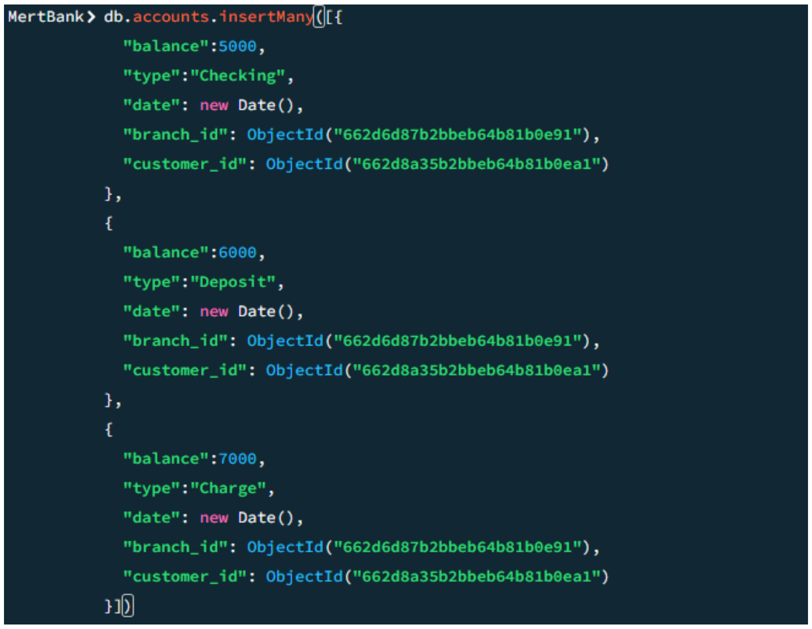
```
> db.customers.update({"_id": ObjectId("662d8a35b2bbeb64b81b0ea0")}, {$push:{"accounts": ObjectId("662e008cde9fd66c707c5136")}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Carl now looks like this:

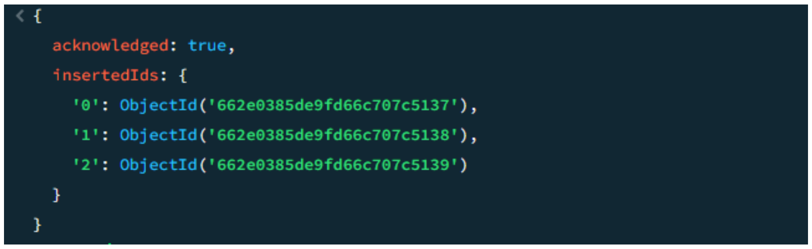


```
  _id: ObjectId('662d8a35b2bbeb64b81b0ea0')
▸ name : Object
  email : "carl@gmail.com"
  phone : "+13206932927"
  address : "Apt. 513 8307 Balistreri Tunnel, Daretown, AR 60725"
▾ accounts : Array (1)
    0: ObjectId('662e008cde9fd66c707c5136')
```

Inserting 3 accounts for user Richard Feynman (for EastSide branch):

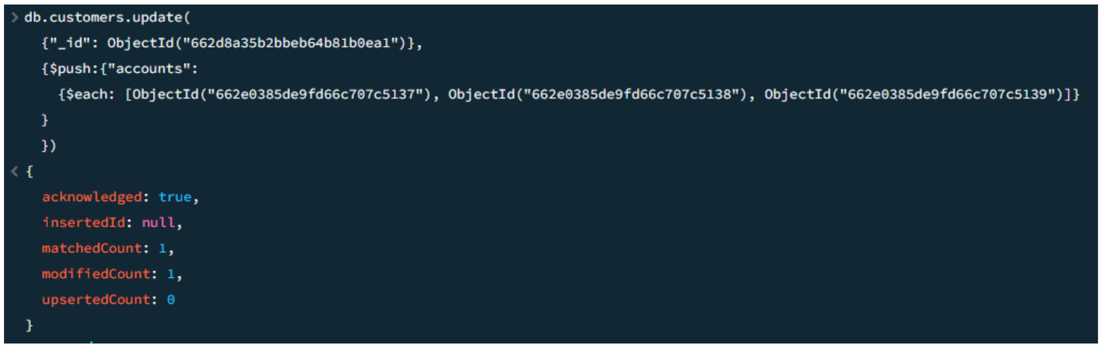


```
MertBank> db.accounts.insertMany([{
            "balance":5000,
            "type":"Checking",
            "date": new Date(),
            "branch_id": ObjectId("662d6d87b2bbeb64b81b0e91"),
            "customer_id": ObjectId("662d8a35b2bbeb64b81b0ea1")
          },
          {
            "balance":6000,
            "type":"Deposit",
            "date": new Date(),
            "branch_id": ObjectId("662d6d87b2bbeb64b81b0e91"),
            "customer_id": ObjectId("662d8a35b2bbeb64b81b0ea1")
          },
          {
            "balance":7000,
            "type":"Charge",
            "date": new Date(),
            "branch_id": ObjectId("662d6d87b2bbeb64b81b0e91"),
            "customer_id": ObjectId("662d8a35b2bbeb64b81b0ea1")
          }])
```



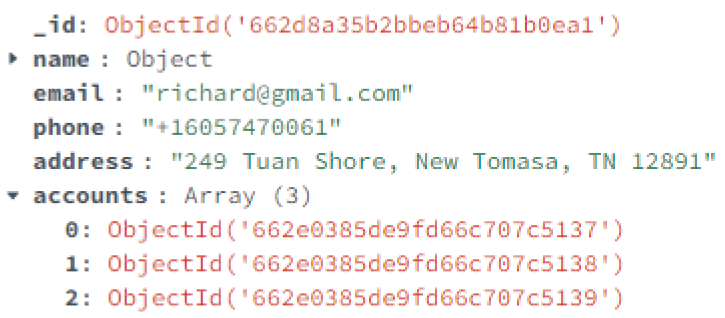
```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('662e0385de9fd66c707c5137'),
      '1': ObjectId('662e0385de9fd66c707c5138'),
      '2': ObjectId('662e0385de9fd66c707c5139')
    }
  }
```

Updating Richard’s accounts field:



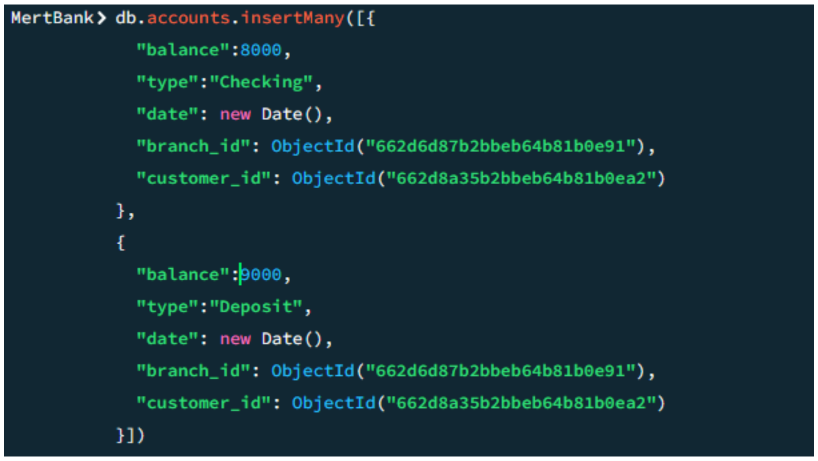
```
> db.customers.update(
    {"_id": ObjectId("662d8a35b2bbeb64b81b0ea1")},
    {$push:{"accounts":
      {$each: [ObjectId("662e0385de9fd66c707c5137"), ObjectId("662e0385de9fd66c707c5138"), ObjectId("662e0385de9fd66c707c5139")]}
    }
    })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Richard now looks like this:

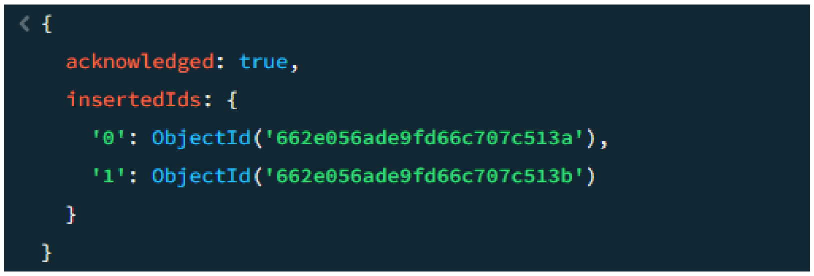


```
  _id: ObjectId('662d8a35b2bbeb64b81b0ea1')
▸ name : Object
  email : "richard@gmail.com"
  phone : "+16057470061"
  address : "249 Tuan Shore, New Tomasa, TN 12891"
▾ accounts : Array (3)
    0: ObjectId('662e0385de9fd66c707c5137')
    1: ObjectId('662e0385de9fd66c707c5138')
    2: ObjectId('662e0385de9fd66c707c5139')
```

Inserting 2 accounts for user Nilgün Marmara (for EastSide branch):

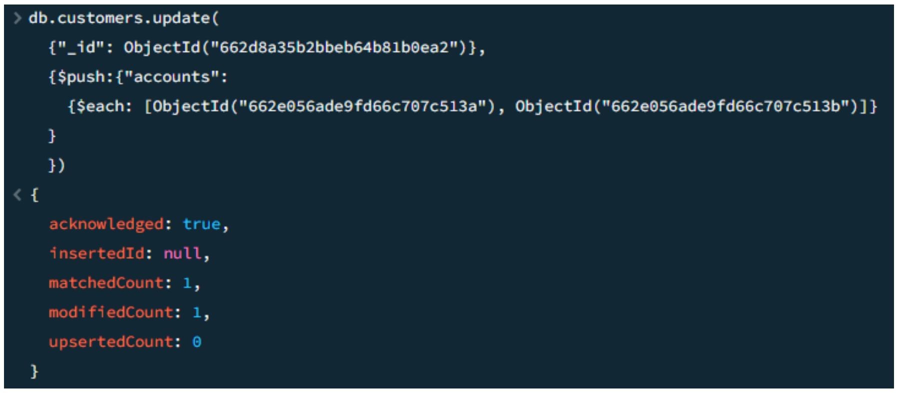


```
MertBank> db.accounts.insertMany([{
            "balance":8000,
            "type":"Checking",
            "date": new Date(),
            "branch_id": ObjectId("662d6d87b2bbeb64b81b0e91"),
            "customer_id": ObjectId("662d8a35b2bbeb64b81b0ea2")
          },
          {
            "balance":9000,
            "type":"Deposit",
            "date": new Date(),
            "branch_id": ObjectId("662d6d87b2bbeb64b81b0e91"),
            "customer_id": ObjectId("662d8a35b2bbeb64b81b0ea2")
          }])
```



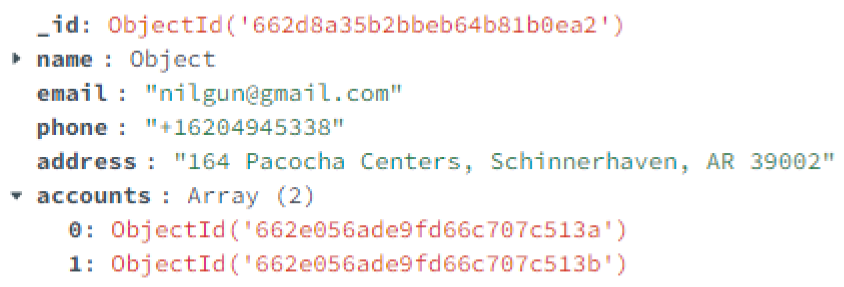
```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('662e056ade9fd66c707c513a'),
      '1': ObjectId('662e056ade9fd66c707c513b')
    }
  }
```

Updating Nilgün’s accounts field:



```
> db.customers.update(
    {"_id": ObjectId("662d8a35b2bbeb64b81b0ea2")},
    {$push:{"accounts":
      {$each: [ObjectId("662e056ade9fd66c707c513a"), ObjectId("662e056ade9fd66c707c513b")]}
    }
    })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

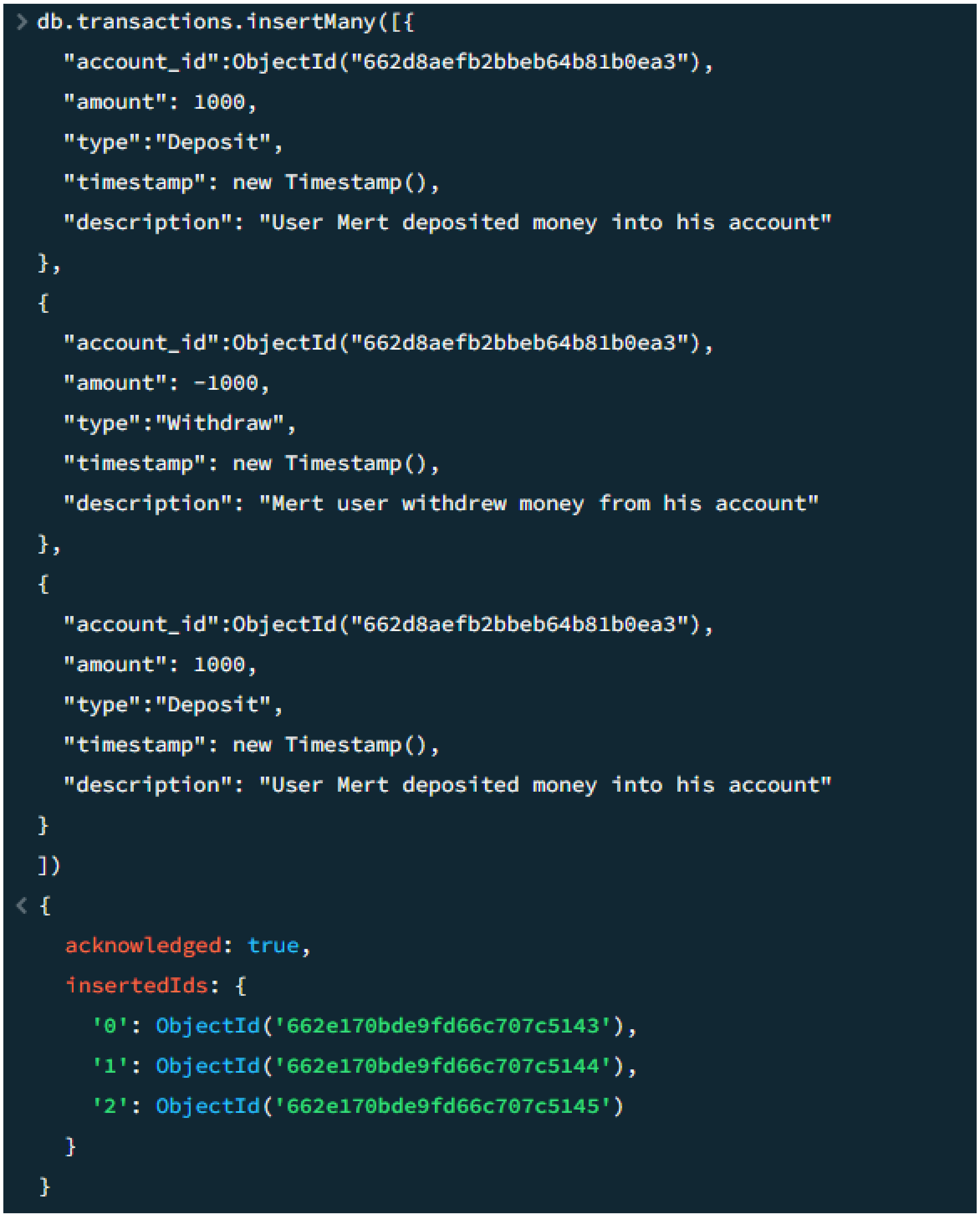
Nilgün now looks like this:



```
    _id: ObjectId('662d8a35b2bbeb64b81b0ea2')
  ▸ name : Object
    email : "nilgun@gmail.com"
    phone : "+16204945338"
    address : "164 Pacocha Centers, Schinnerhaven, AR 39002"
  ▾ accounts : Array (2)
      0: ObjectId('662e056ade9fd66c707c513a')
      1: ObjectId('662e056ade9fd66c707c513b')
```

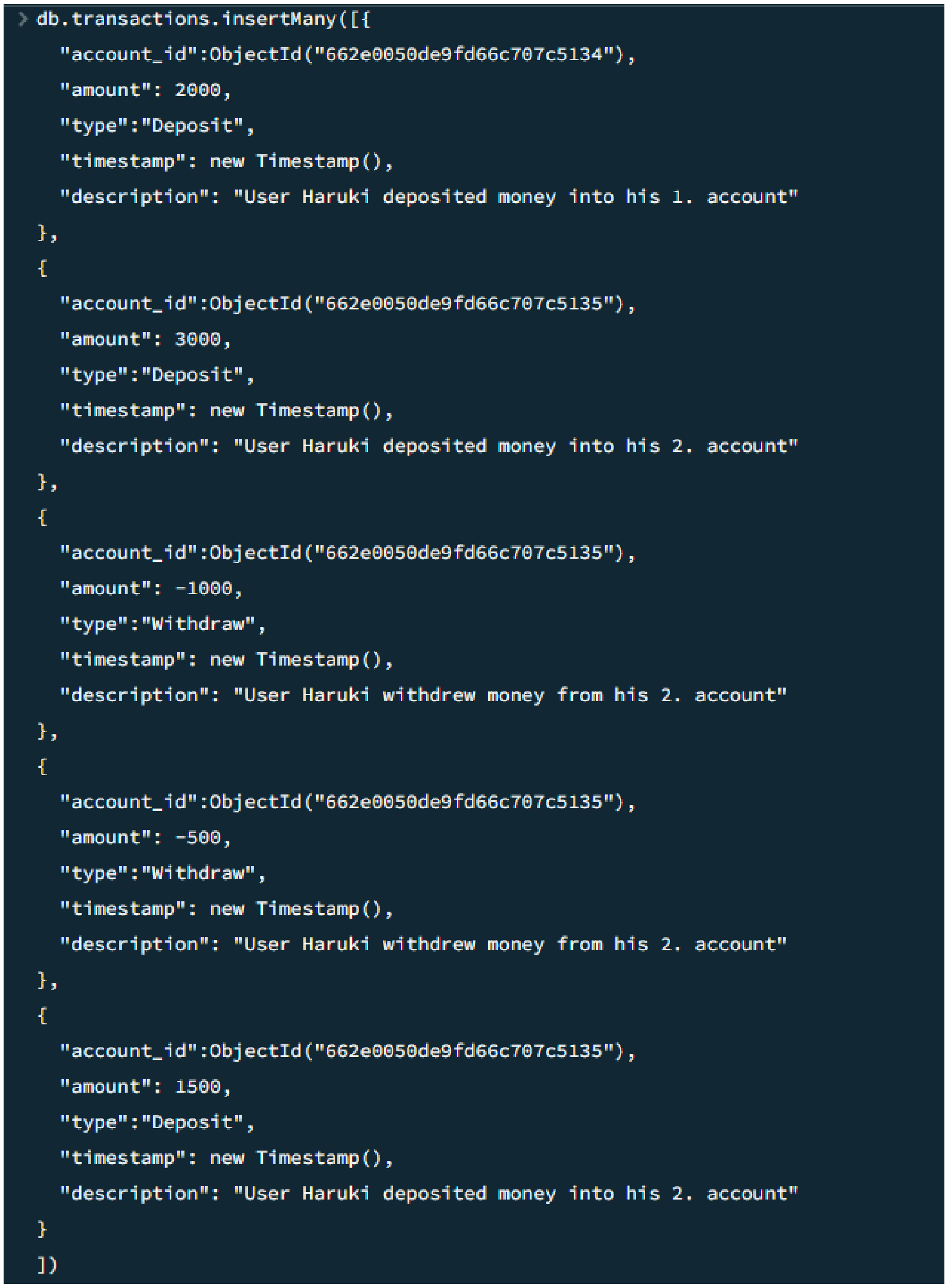
TRANSACTIONS

Now user Mert is making transactions:



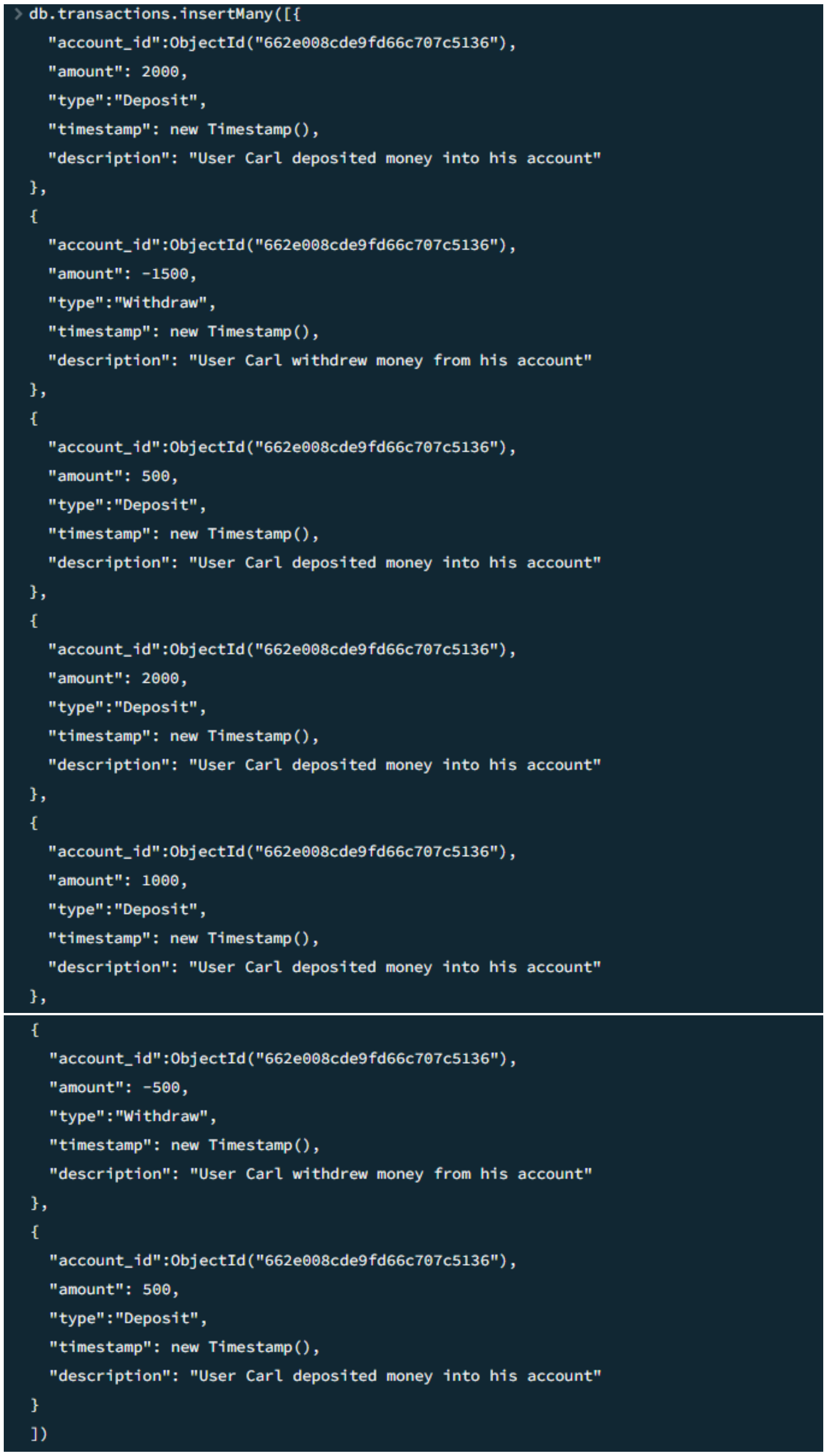
```
> db.transactions.insertMany([{
    "account_id":ObjectId("662d8aefb2bbeb64b81b0ea3"),
    "amount": 1000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Mert deposited money into his account"
  },
  {
    "account_id":ObjectId("662d8aefb2bbeb64b81b0ea3"),
    "amount": -1000,
    "type":"Withdraw",
    "timestamp": new Timestamp(),
    "description": "Mert user withdrew money from his account"
  },
  {
    "account_id":ObjectId("662d8aefb2bbeb64b81b0ea3"),
    "amount": 1000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Mert deposited money into his account"
  }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('662e170bde9fd66c707c5143'),
      '1': ObjectId('662e170bde9fd66c707c5144'),
      '2': ObjectId('662e170bde9fd66c707c5145')
    }
  }
```

User Haruki making transactions to his accounts:



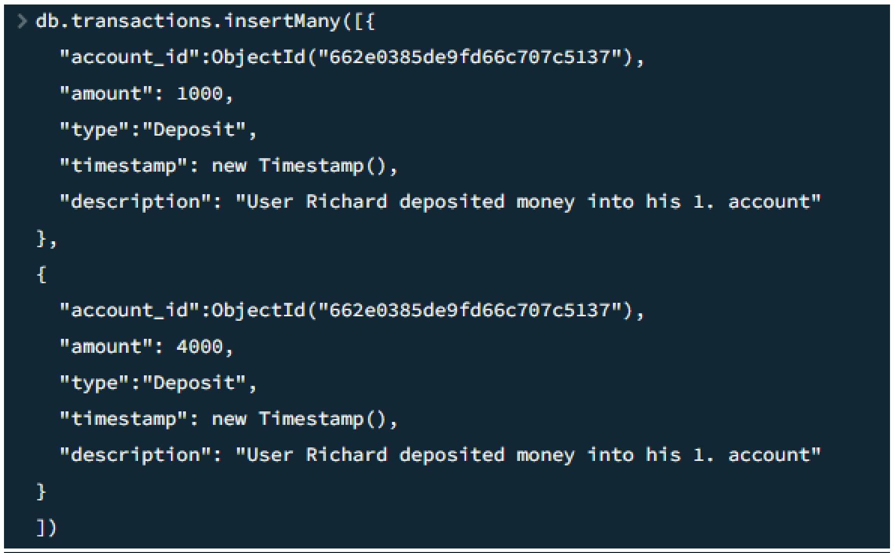
```
> db.transactions.insertMany([{
    "account_id":ObjectId("662e0050de9fd66c707c5134"),
    "amount": 2000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Haruki deposited money into his 1. account"
  },
  {
    "account_id":ObjectId("662e0050de9fd66c707c5135"),
    "amount": 3000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Haruki deposited money into his 2. account"
  },
  {
    "account_id":ObjectId("662e0050de9fd66c707c5135"),
    "amount": -1000,
    "type":"Withdraw",
    "timestamp": new Timestamp(),
    "description": "User Haruki withdrew money from his 2. account"
  },
  {
    "account_id":ObjectId("662e0050de9fd66c707c5135"),
    "amount": -500,
    "type":"Withdraw",
    "timestamp": new Timestamp(),
    "description": "User Haruki withdrew money from his 2. account"
  },
  {
    "account_id":ObjectId("662e0050de9fd66c707c5135"),
    "amount": 1500,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Haruki deposited money into his 2. account"
  }
  ])
```

User Carl making transactions to his account:

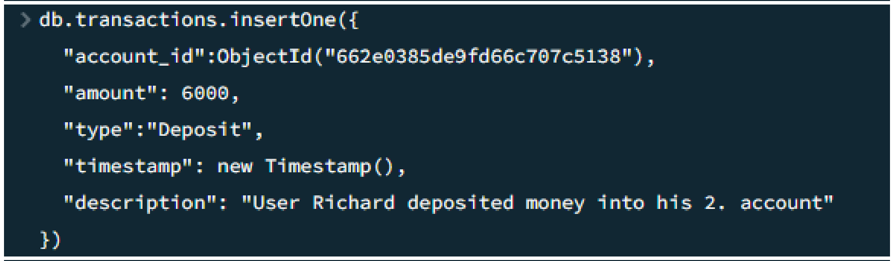


```
> db.transactions.insertMany([{
    "account_id":ObjectId("662e008cde9fd66c707c5136"),
    "amount": 2000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Carl deposited money into his account"
  },
  {
    "account_id":ObjectId("662e008cde9fd66c707c5136"),
    "amount": -1500,
    "type":"Withdraw",
    "timestamp": new Timestamp(),
    "description": "User Carl withdrew money from his account"
  },
  {
    "account_id":ObjectId("662e008cde9fd66c707c5136"),
    "amount": 500,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Carl deposited money into his account"
  },
  {
    "account_id":ObjectId("662e008cde9fd66c707c5136"),
    "amount": 2000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Carl deposited money into his account"
  },
  {
    "account_id":ObjectId("662e008cde9fd66c707c5136"),
    "amount": 1000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Carl deposited money into his account"
  },
  {
    "account_id":ObjectId("662e008cde9fd66c707c5136"),
    "amount": -500,
    "type":"Withdraw",
    "timestamp": new Timestamp(),
    "description": "User Carl withdrew money from his account"
  },
  {
    "account_id":ObjectId("662e008cde9fd66c707c5136"),
    "amount": 500,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Carl deposited money into his account"
  }
  ])
```

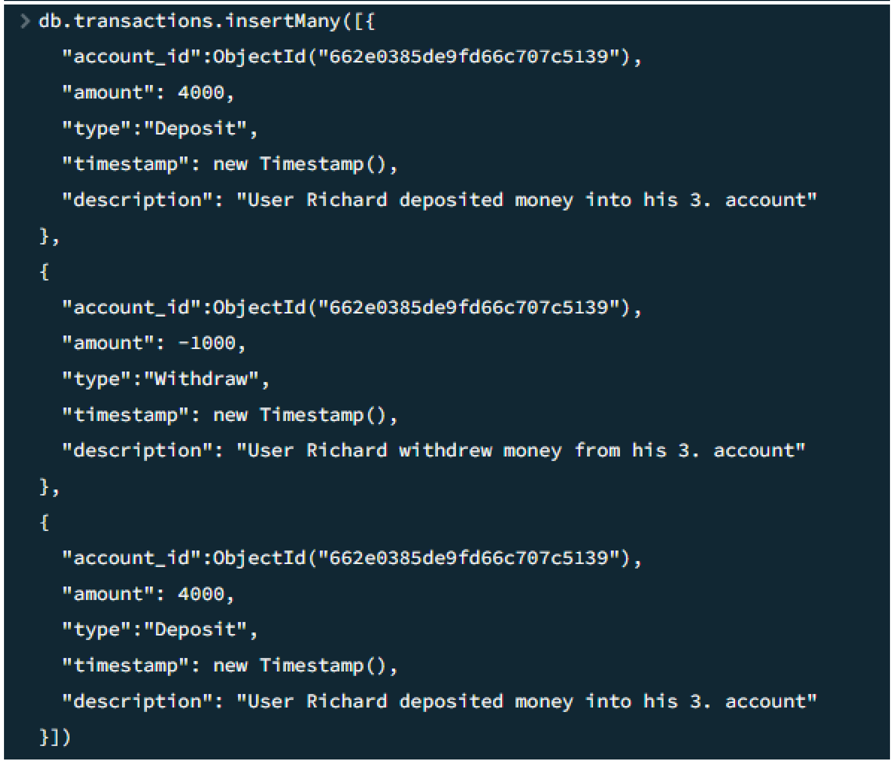
User Richard making transactions to his accounts:



```
> db.transactions.insertMany([{
    "account_id":ObjectId("662e0385de9fd66c707c5137"),
    "amount": 1000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Richard deposited money into his 1. account"
  },
  {
    "account_id":ObjectId("662e0385de9fd66c707c5137"),
    "amount": 4000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Richard deposited money into his 1. account"
  }
  ])
```

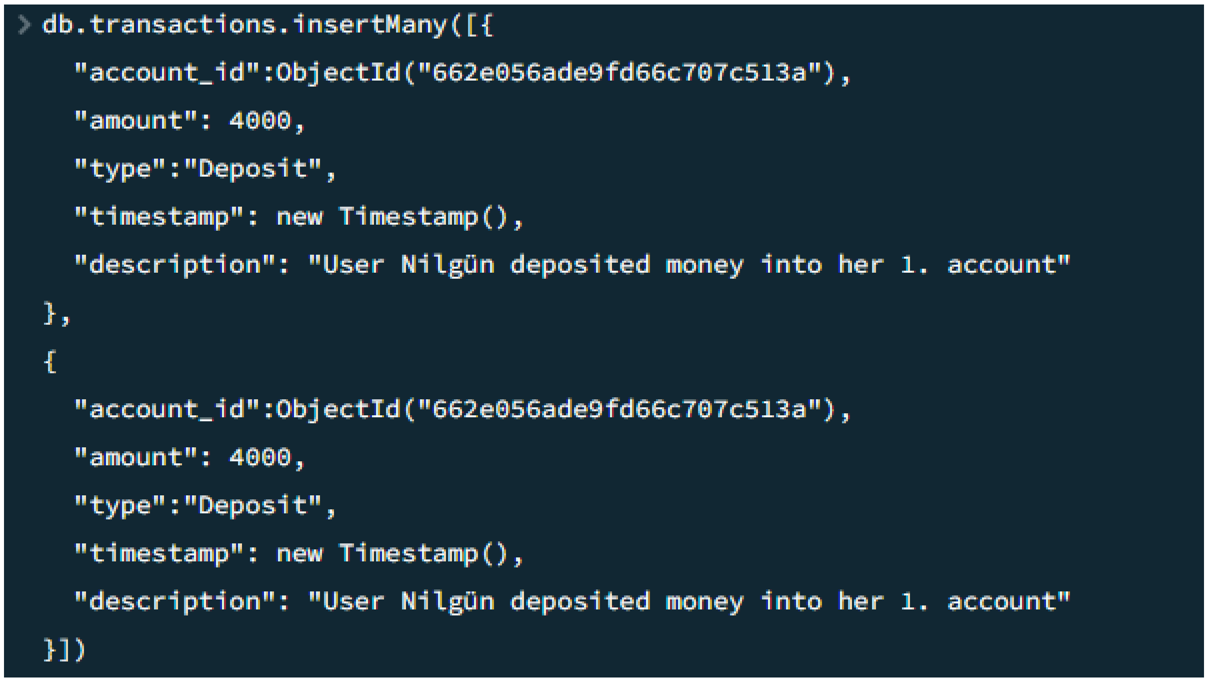


```
> db.transactions.insertOne({
    "account_id":ObjectId("662e0385de9fd66c707c5138"),
    "amount": 6000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Richard deposited money into his 2. account"
  })
```

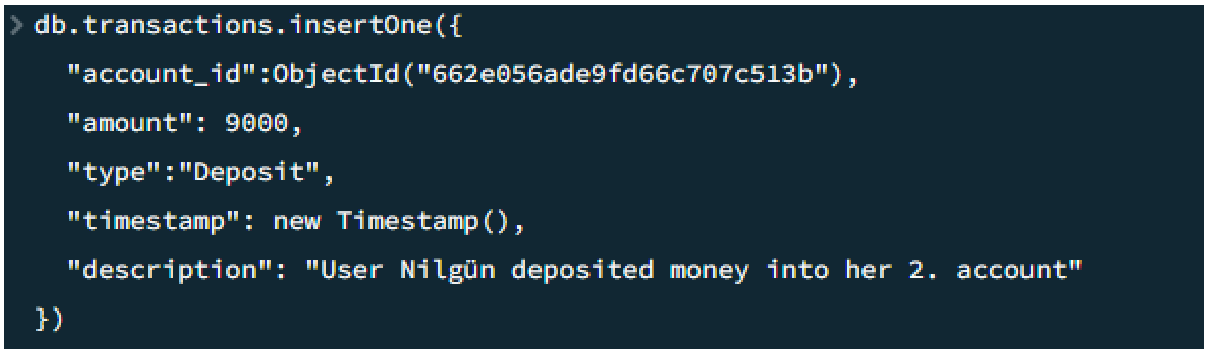


```
> db.transactions.insertMany([{
    "account_id":ObjectId("662e0385de9fd66c707c5139"),
    "amount": 4000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Richard deposited money into his 3. account"
  },
  {
    "account_id":ObjectId("662e0385de9fd66c707c5139"),
    "amount": -1000,
    "type":"Withdraw",
    "timestamp": new Timestamp(),
    "description": "User Richard withdrew money from his 3. account"
  },
  {
    "account_id":ObjectId("662e0385de9fd66c707c5139"),
    "amount": 4000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Richard deposited money into his 3. account"
  }])
```

User Nilgün making transactions to her accounts:



```
db.transactions.insertMany([{
    "account_id":ObjectId("662e056ade9fd66c707c513a"),
    "amount": 4000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Nilgün deposited money into her 1. account"
  },
  {
    "account_id":ObjectId("662e056ade9fd66c707c513a"),
    "amount": 4000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Nilgün deposited money into her 1. account"
  }])
```



```
db.transactions.insertOne({
    "account_id":ObjectId("662e056ade9fd66c707c513b"),
    "amount": 9000,
    "type":"Deposit",
    "timestamp": new Timestamp(),
    "description": "User Nilgün deposited money into her 2. account"
  })
```

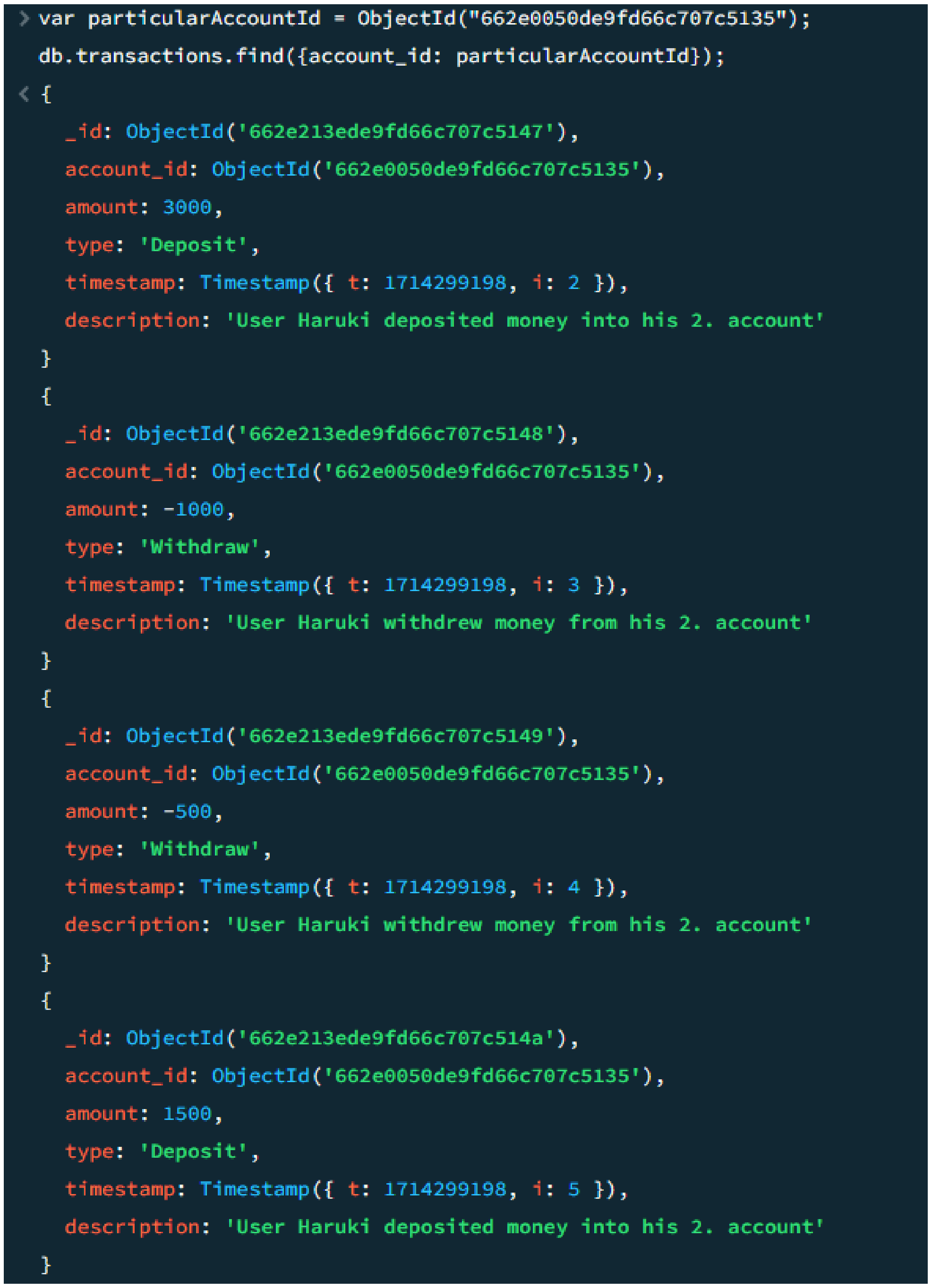
I am summing up all the accounts and transactions below:

- **MERT GÜRŞİMŞİR (1 account, 3 transactions)**
  - Account
    - Transactions: +1000, -1000, +1000
- **HARUKI MURAKAMI (2 accounts, 5 transactions)**
  - Account 1
    - Transactions: +2000
  - Account 2
    - Transactions: +3000, -1000, -500, +1500
- **CARL SAGAN (1 account, 7 transactions)**
  - Account
    - Transactions: +2000, -1500, +500, +2000, +1000, -500, +500
- **RICHARD FEYNMAN (3 accounts, 6 transactions)**
  - Account 1
    - Transactions: +1000, +4000
  - Account 2
    - Transactions: +6000
  - Account 3
    - Transactions: +4000, -1000, +4000
- **NİLGÜN MARMARA (2 accounts, 3 transactions)**
  - Account 1
    - Transactions: +4000, +4000
  - Account 2
    - Transactions: +9000

QUERIES

1 → Retrieve all transactions associated with a particular account

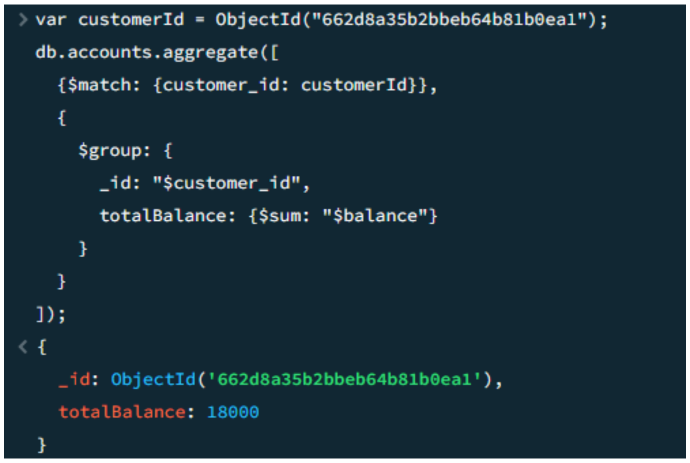
Haruki Murakami's transactions associated with his 2nd account:



```
> var particularAccountId = ObjectId("662e0050de9fd66c707c5135");
  db.transactions.find({account_id: particularAccountId});
< {
    _id: ObjectId('662e213ede9fd66c707c5147'),
    account_id: ObjectId('662e0050de9fd66c707c5135'),
    amount: 3000,
    type: 'Deposit',
    timestamp: Timestamp({ t: 1714299198, i: 2 }),
    description: 'User Haruki deposited money into his 2. account'
  }
  {
    _id: ObjectId('662e213ede9fd66c707c5148'),
    account_id: ObjectId('662e0050de9fd66c707c5135'),
    amount: -1000,
    type: 'Withdraw',
    timestamp: Timestamp({ t: 1714299198, i: 3 }),
    description: 'User Haruki withdrew money from his 2. account'
  }
  {
    _id: ObjectId('662e213ede9fd66c707c5149'),
    account_id: ObjectId('662e0050de9fd66c707c5135'),
    amount: -500,
    type: 'Withdraw',
    timestamp: Timestamp({ t: 1714299198, i: 4 }),
    description: 'User Haruki withdrew money from his 2. account'
  }
  {
    _id: ObjectId('662e213ede9fd66c707c514a'),
    account_id: ObjectId('662e0050de9fd66c707c5135'),
    amount: 1500,
    type: 'Deposit',
    timestamp: Timestamp({ t: 1714299198, i: 5 }),
    description: 'User Haruki deposited money into his 2. account'
  }
```

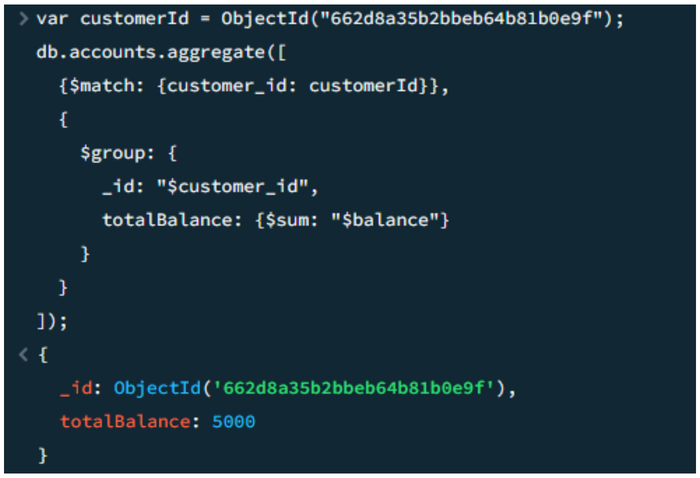
2 → Find the total balance of a customer across all accounts

Finding total balance of Richard Feynman across all of his accounts. His first account has 5000, second account has 6000, third account has 7000 so in total he has 18000:



```
> var customerId = ObjectId("662d8a35b2bbeb64b81b0ea1");
  db.accounts.aggregate([
    {$match: {customer_id: customerId}},
    {
      $group: {
        _id: "$customer_id",
        totalBalance: {$sum: "$balance"}
      }
    }
  ]);
< {
    _id: ObjectId('662d8a35b2bbeb64b81b0ea1'),
    totalBalance: 18000
  }
```

Finding total balance of Haruki Murakami across all of his accounts. His first account has 2000, second account has 3000 so in total he has 5000:



```
> var customerId = ObjectId("662d8a35b2bbeb64b81b0e9f");
  db.accounts.aggregate([
    {$match: {customer_id: customerId}},
    {
      $group: {
        _id: "$customer_id",
        totalBalance: {$sum: "$balance"}
      }
    }
  ]);
< {
    _id: ObjectId('662d8a35b2bbeb64b81b0e9f'),
    totalBalance: 5000
  }
```

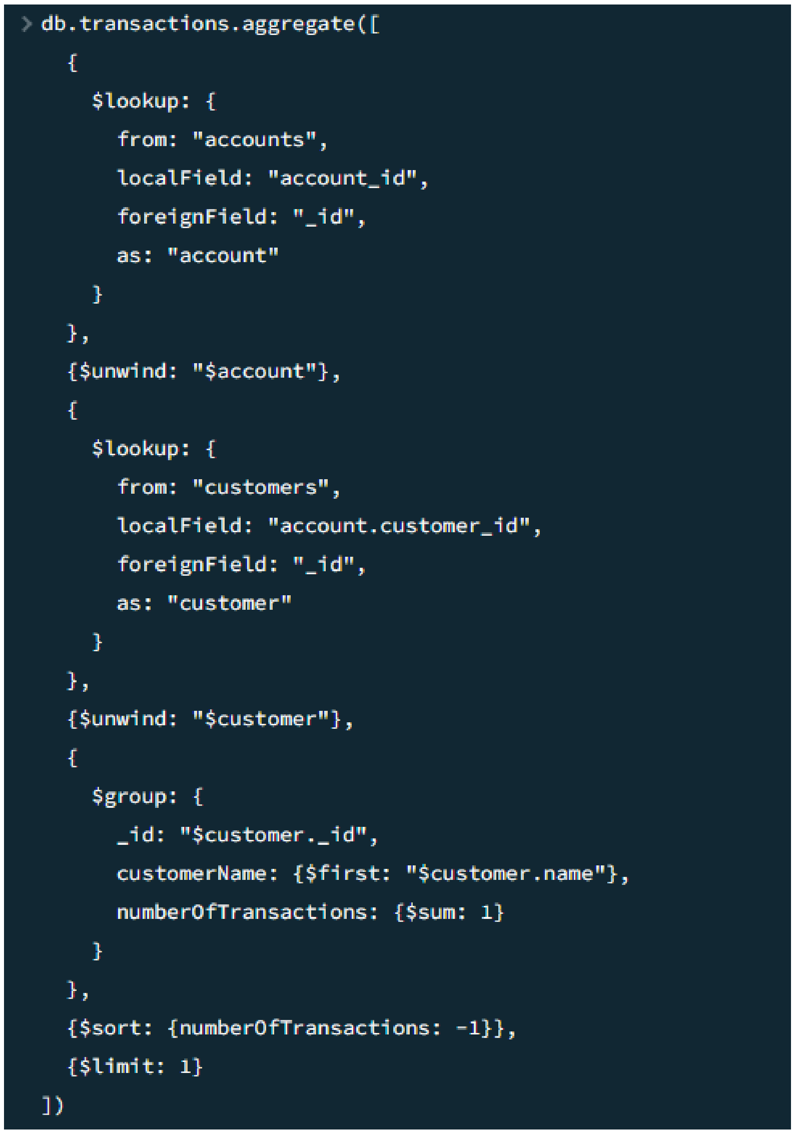
With aggregate, we process data records and return computed results. match and group are stages in aggregation pipeline:

I. $match: Filters documents based on specified criteria similar to find() method. Here we are selecting only those accounts associated with the specified customer based on the customer_id.
II. $group: Groups the documents based on specified criteria and applies expressions to the grouped data. Here we are grouping the accounts by customer_id and calculating the total balance.

3 → Identify customers with the highest number of transactions

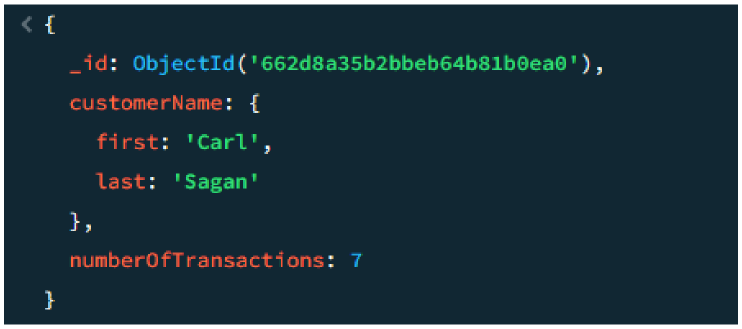
In the current situation, Carl Sagan has the most number of transactions with 7 transactions.

Query:



```
> db.transactions.aggregate([
    {
      $lookup: {
        from: "accounts",
        localField: "account_id",
        foreignField: "_id",
        as: "account"
      }
    },
    {$unwind: "$account"},
    {
      $lookup: {
        from: "customers",
        localField: "account.customer_id",
        foreignField: "_id",
        as: "customer"
      }
    },
    {$unwind: "$customer"},
    {
      $group: {
        _id: "$customer._id",
        customerName: {$first: "$customer.name"},
        numberOfTransactions: {$sum: 1}
      }
    },
    {$sort: {numberOfTransactions: -1}},
    {$limit: 1}
  ])
```

Result:

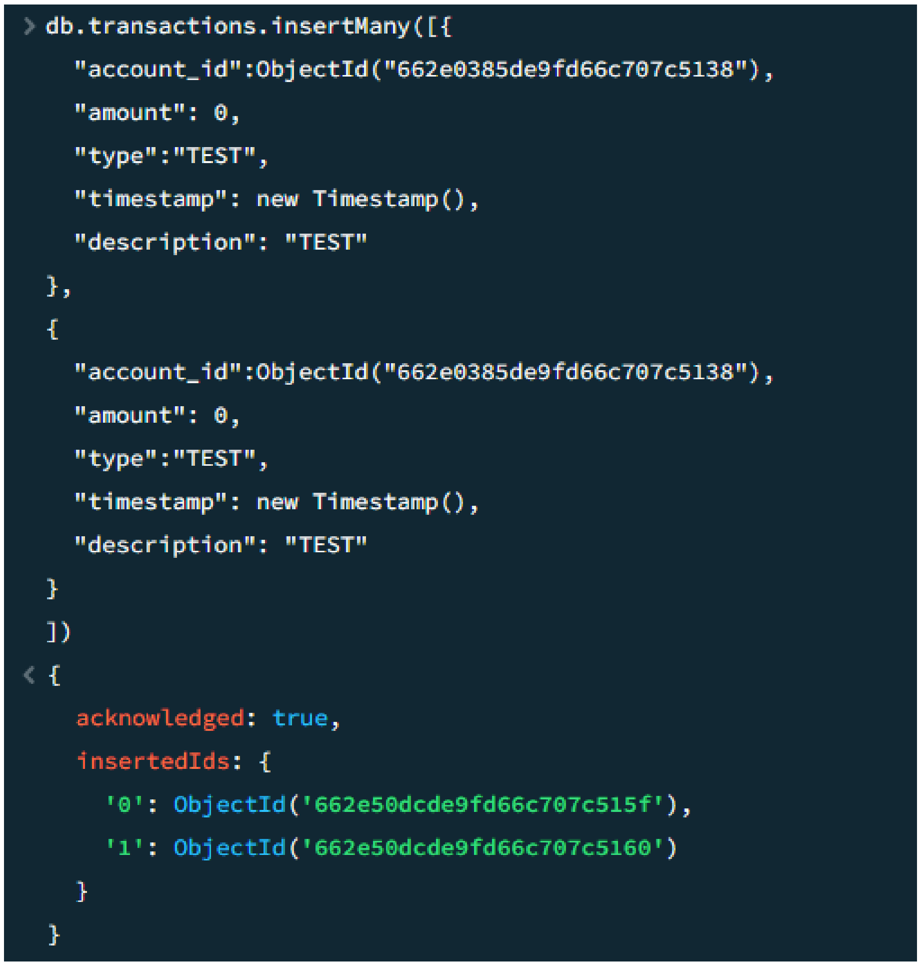


```
< {
    _id: ObjectId('662d8a35b2bbeb64b81b0ea0'),
    customerName: {
      first: 'Carl',
      last: 'Sagan'
    },
    numberOfTransactions: 7
  }
```

Here we have 7 stages:

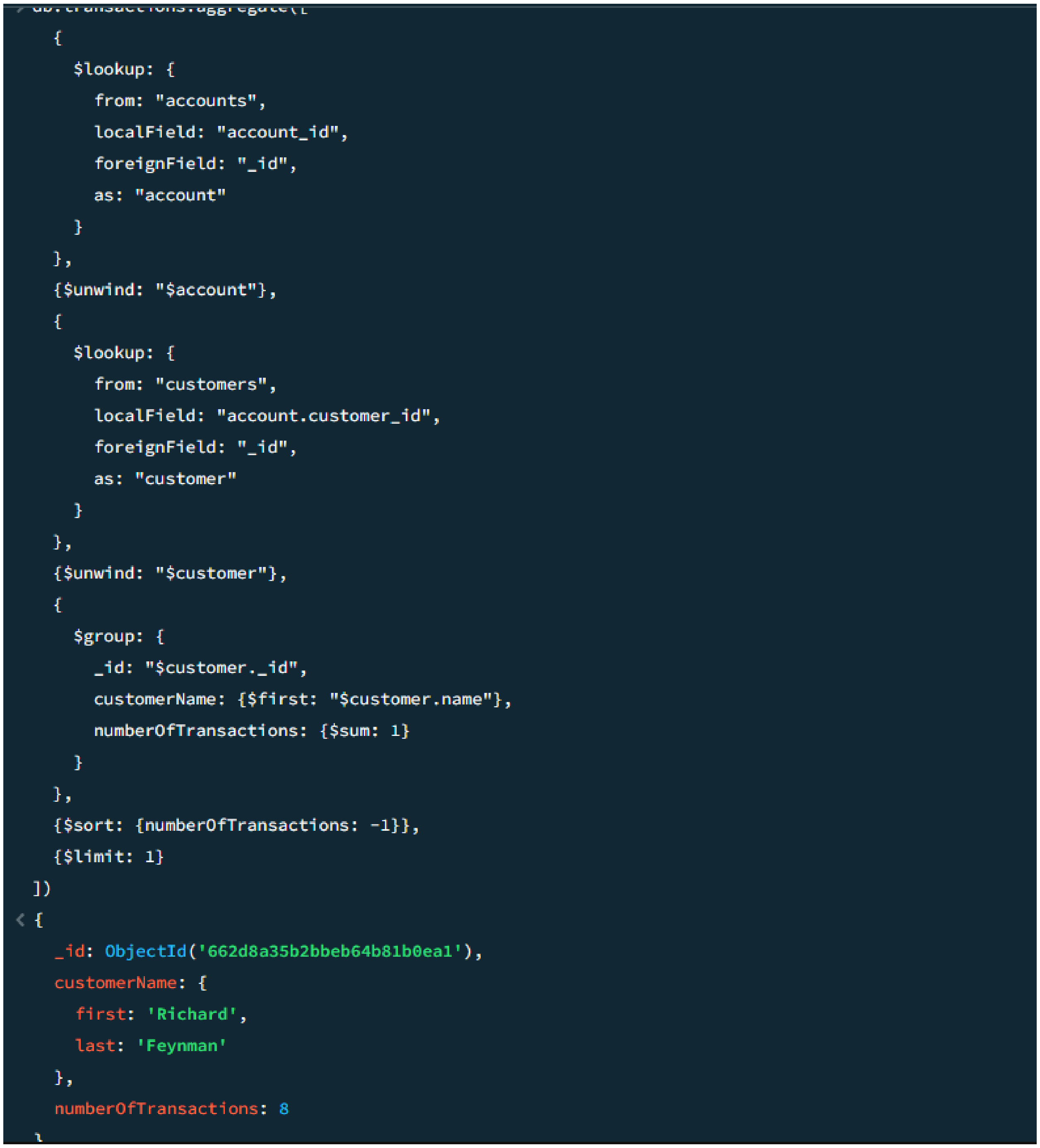
I. $lookup: Here we are joining the transactions collection with the accounts collection based on the account_id.
II. $unwind: Here we are deconstructing the resulting array from the $lookup.
III. $lookup: This another lookup is for joining the accounts collection with the customers collection based on the customer's id.
IV. $unwind: This unwing is deconstructing the resulting array from the second $lookup.
V. $group: In this stage, we are grouping transactions by customer and calculating the total number of transactions for each customer.
VI. $sort: Here we are sorting the customers based on their total number of transactions in descending order specified by -1.
VII. $limit: This is for retrieving only the first result which will be the customer with the maximum number of transactions.

Richard Feynman has 6 transactions. I am adding 2 test transaction for his second account:



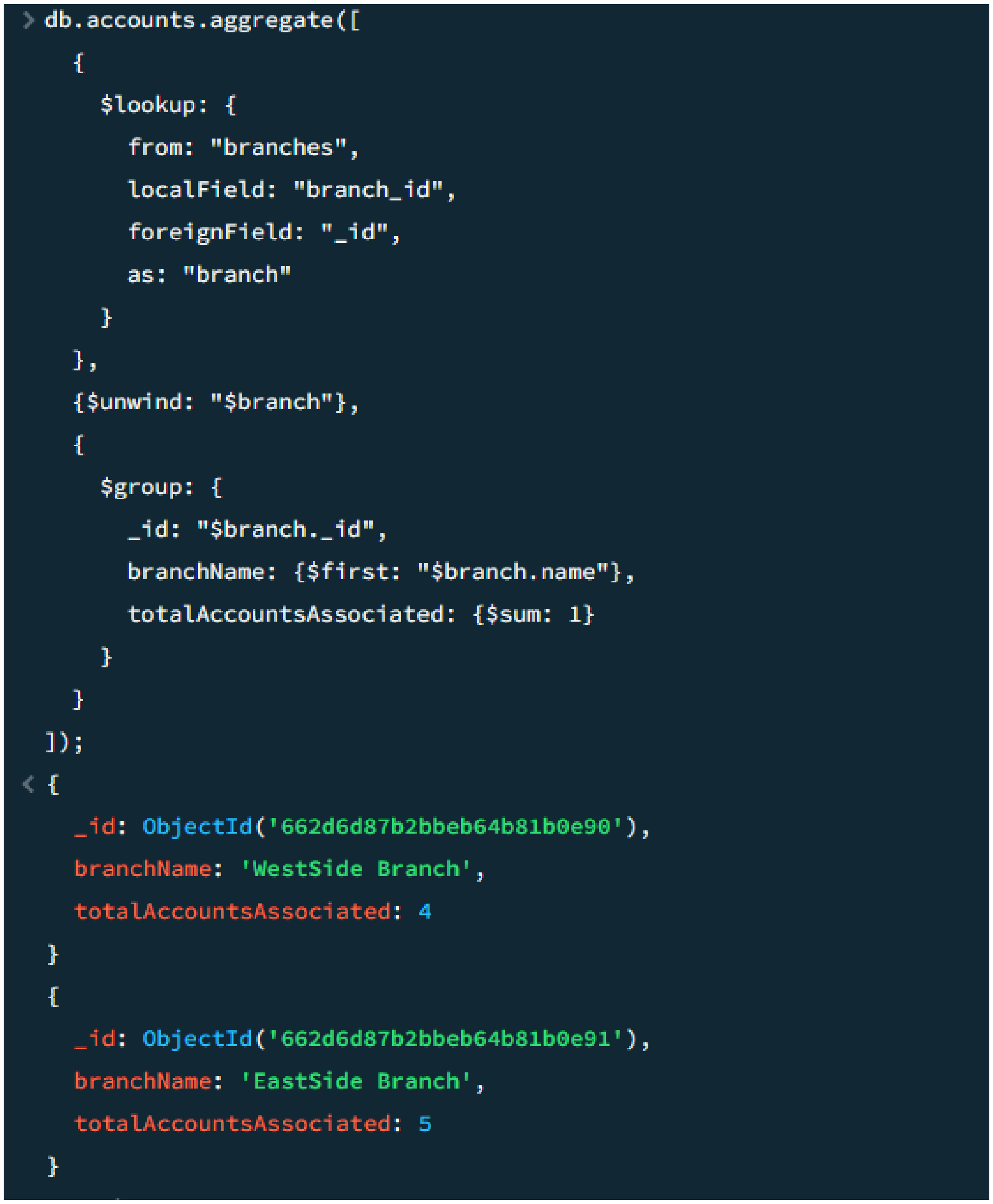
```
> db.transactions.insertMany([{
    "account_id":ObjectId("662e0385de9fd66c707c5138"),
    "amount": 0,
    "type":"TEST",
    "timestamp": new Timestamp(),
    "description": "TEST"
  },
  {
    "account_id":ObjectId("662e0385de9fd66c707c5138"),
    "amount": 0,
    "type":"TEST",
    "timestamp": new Timestamp(),
    "description": "TEST"
  }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('662e50dcde9fd66c707c515f'),
      '1': ObjectId('662e50dcde9fd66c707c5160')
    }
  }
```

Now making the same query I am getting the result:



```
> db.transactions.aggregate([
    {
      $lookup: {
        from: "accounts",
        localField: "account_id",
        foreignField: "_id",
        as: "account"
      }
    },
    {$unwind: "$account"},
    {
      $lookup: {
        from: "customers",
        localField: "account.customer_id",
        foreignField: "_id",
        as: "customer"
      }
    },
    {$unwind: "$customer"},
    {
      $group: {
        _id: "$customer._id",
        customerName: {$first: "$customer.name"},
        numberOfTransactions: {$sum: 1}
      }
    },
    {$sort: {numberOfTransactions: -1}},
    {$limit: 1}
  ])
< {
    _id: ObjectId('662d8a35b2bbeb64b81b0ea1'),
    customerName: {
      first: 'Richard',
      last: 'Feynman'
    },
    numberOfTransactions: 8
  }
```

4 → Get a list of branches along with the total number of accounts associated with each branch



```
> db.accounts.aggregate([
    {
      $lookup: {
        from: "branches",
        localField: "branch_id",
        foreignField: "_id",
        as: "branch"
      }
    },
    {$unwind: "$branch"},
    {
      $group: {
        _id: "$branch._id",
        branchName: {$first: "$branch.name"},
        totalAccountsAssociated: {$sum: 1}
      }
    }
  ]);
< {
    _id: ObjectId('662d6d87b2bbeb64b81b0e90'),
    branchName: 'WestSide Branch',
    totalAccountsAssociated: 4
  }
  {
    _id: ObjectId('662d6d87b2bbeb64b81b0e91'),
    branchName: 'EastSide Branch',
    totalAccountsAssociated: 5
  }
```

Mert's 1 account, Haruki's 2 accounts, Carl's 1 account are associated with WestSide branch;

Richard's 3 accounts, Nilgün's 2 accounts are associated with EastSide branch.

Stages:

I. $lookup: To join accounts collection with the branches collection based on branch_id.
II. $unwind: Deconstructing resulting array from $lookup.
III. $group: Grouping accounts by branch and calculating total number of accounts for each.