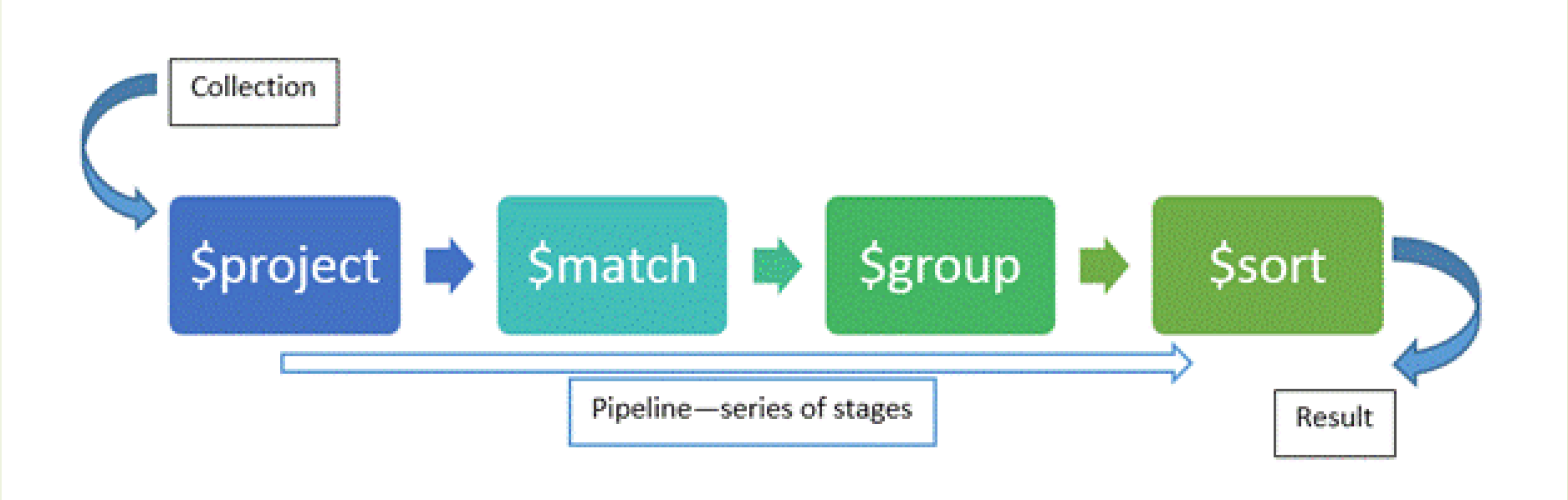
mongodb

Aggregation Frameworks

- Aggregation pipeline
- Map-reduce function
- Single purpose aggregation methods

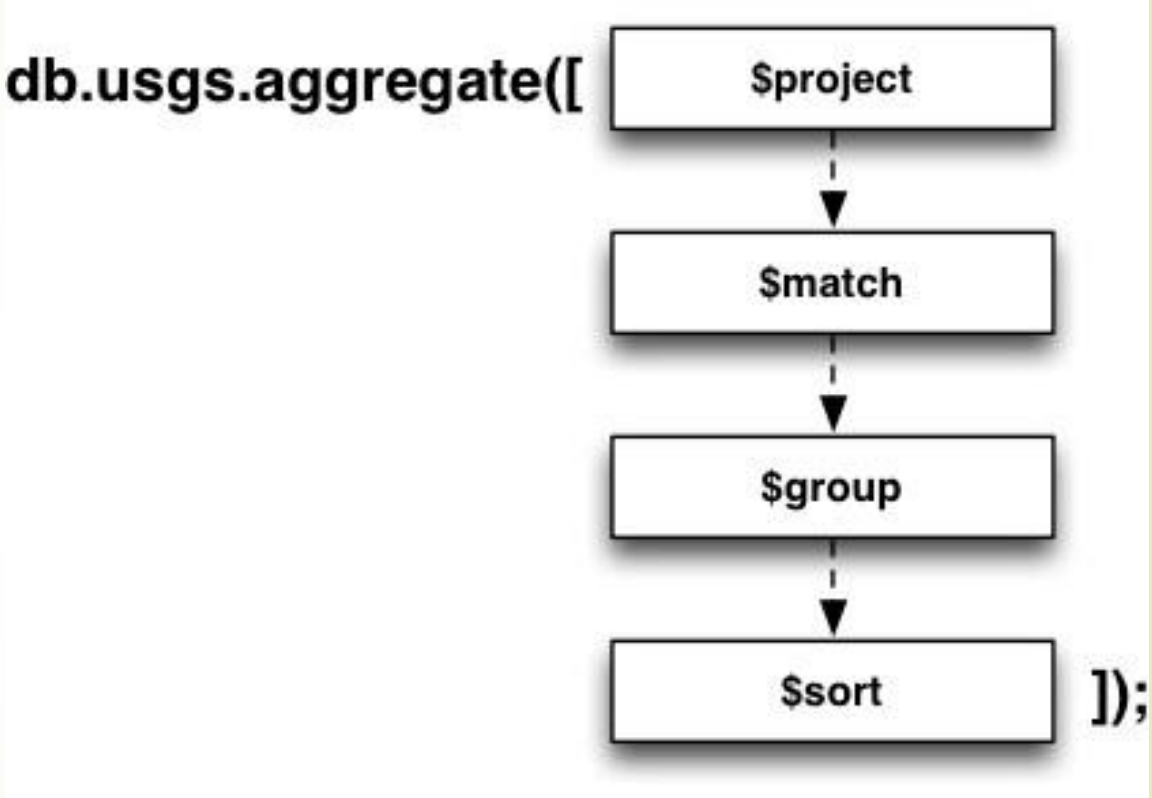
Aggregation Pipeline



Aggregation Pipeline

- The MongoDB aggregation pipeline consists of **stages**.

db.collection.aggregate([{ <stage> }, ...])



Stage

- Each stage transforms the documents as they pass through the pipeline.
- Stages can appear multiple times in a pipeline.
- Examples:
  - $match
  - $group
  - $sort
  - $limit

$group stage

- Groups documents by some specified expression and outputs to the next stage a document for each distinct grouping.

{ $group: { _id: <expression>, <field1>: { <accumulator1> : <expression1> }, ... } }

$unwind stage

- Deconstructs an array field from the input documents to output a document for *each* element.
- Each output document is the input document with the value of the array field replaced by the element.

```
{ $unwind: <field path> }
```

$bucket stage

- Categorizes incoming documents into groups, called buckets, based on a specified expression and bucket boundaries.

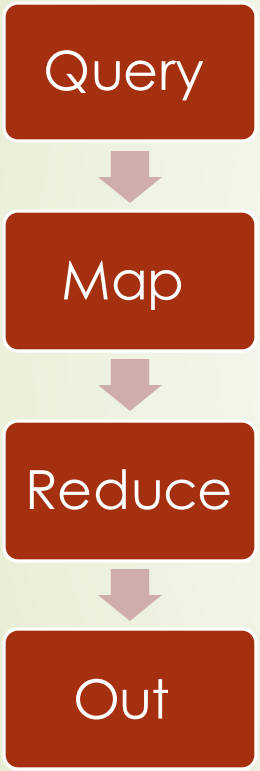
```
{
  $bucket: {
      groupBy: <expression>,
      boundaries: [ <lowerbound1>, <lowerbound2>, ... ],
      default: <literal>,
      output: {
         <output1>: { <$accumulator expression> },
         ...
         <outputN>: { <$accumulator expression> }
      }
   }
}
```

$lookup

- Performs a left outer join to an unsharded collection in the *same* database.
- To each input document, the $lookup stage adds a new array field whose elements are the matching documents from the "joined" collection.

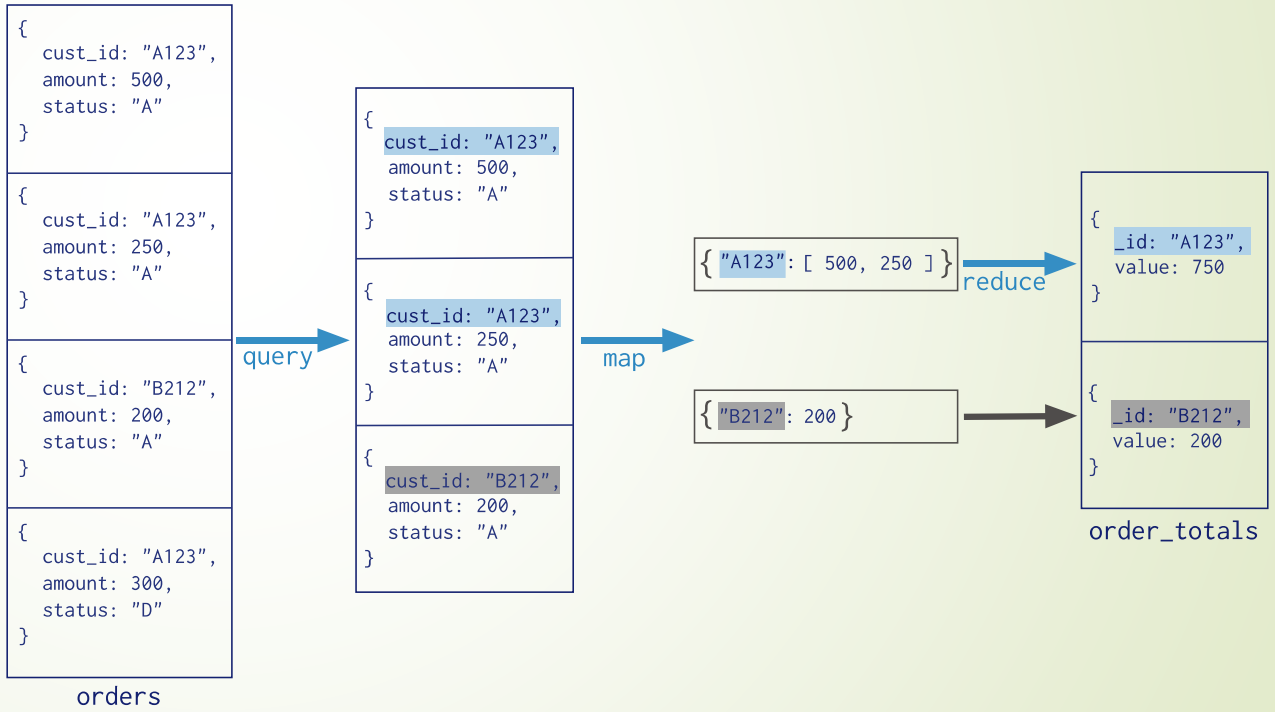
```
{
  $lookup:
   {
     from: <collection to join>,
     localField: <field from the input documents>,
     foreignField: <field from the documents of the "from" collection>,
     as: <output array field>
   }
}
```

Map-Reduce



Collection

```
db.orders.mapReduce(
        map      →  function() { emit( this.cust_id, this.amount ); },
        reduce   →  function(key, values) { return Array.sum( values ) },
                    {
        query    →    query: { status: "A" },
        output   →    out: "order_totals"
                    }
                )
```



Single purpose aggregation methods

- db.collection.estimatedDocumentCount()
- db.collection.count()
- db.collection.distinct()

Collection

```
db.orders.distinct( "cust_id" )
```

```
{
  cust_id: "A123",
  amount: 500,
  status: "A"
}
```

```
{
  cust_id: "A123",
  amount: 250,
  status: "A"
}
```

```
{
  cust_id: "B212",
  amount: 200,
  status: "A"
}
```

```
{
  cust_id: "A123",
  amount: 300,
  status: "D"
}
```

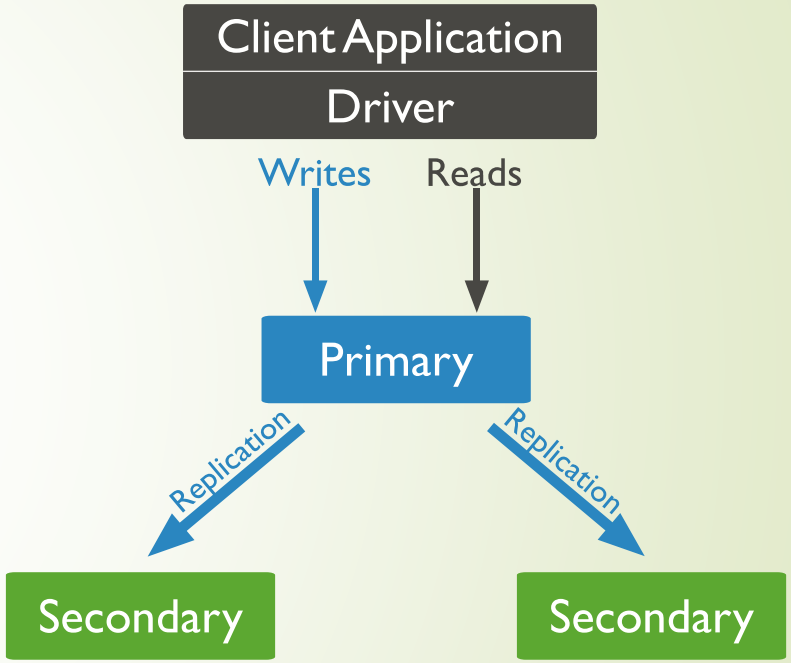
orders

distinct → `[ "A123", "B212" ]`

Replica Set

- Group of mongod processes that maintain the same data set.
- Replication provides :

1. Redundancy.
2. Increases data availability.
3. Fault tolerance.
4. Increased read capacity (If configured).



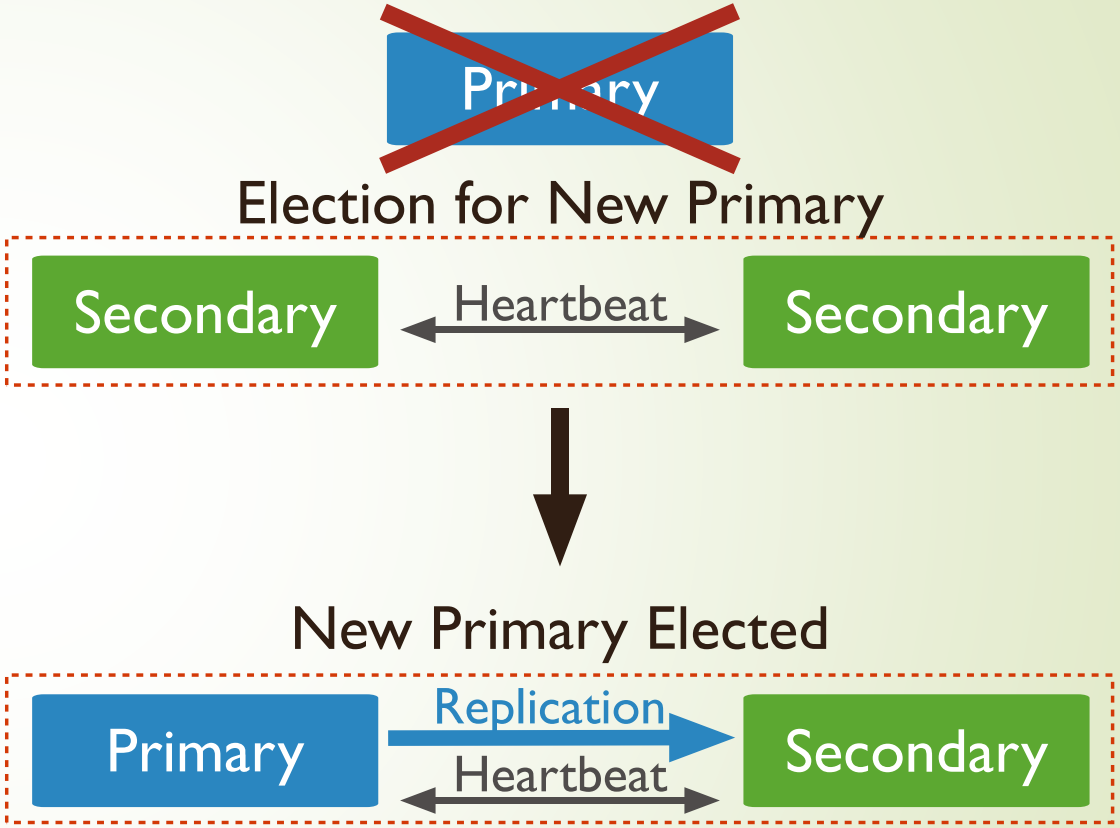
Replica Set

- One primary node, while the other nodes are secondary nodes.
- The primary node receives all write operations.
- The primary records all changes to its data sets in its operation log.
- The secondaries replicate the primary's oplog and apply the operations to their data sets such that the secondaries' data sets reflect the primary's data set.

Replica Set Elections

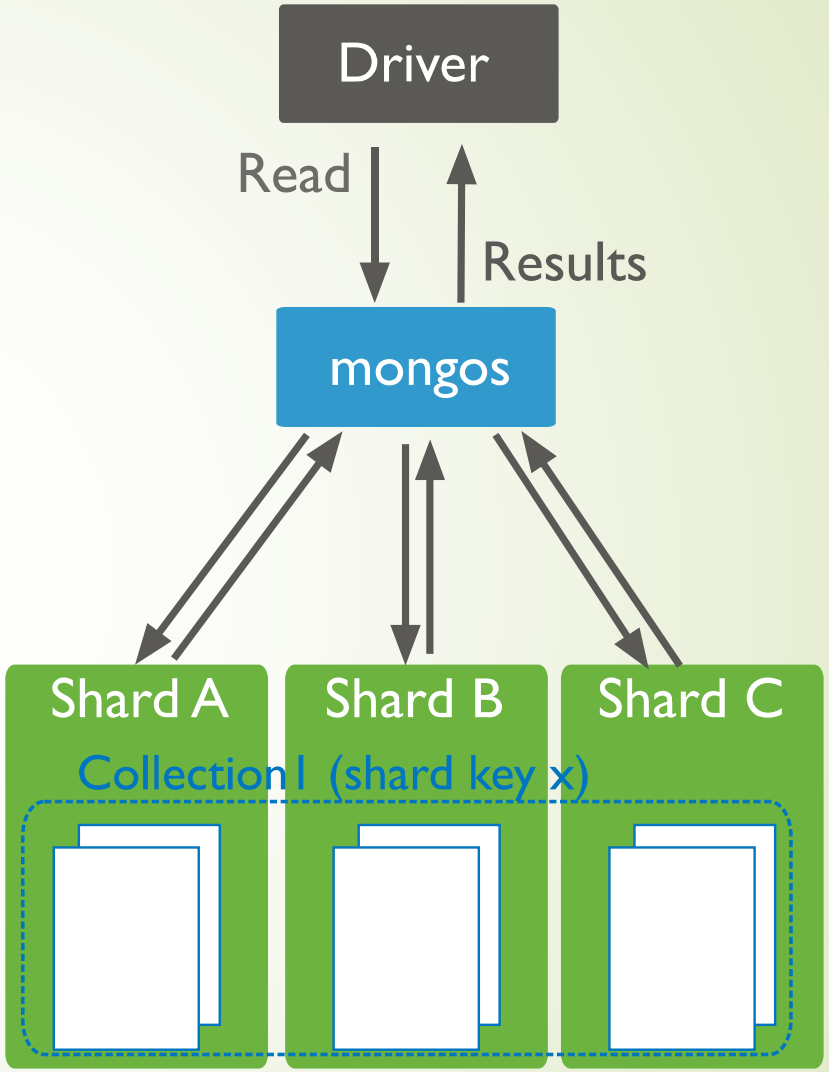
- Replica sets use elections to determine which set member will become primary.
- When??

1. Add new node
2. Create replica set.
3. Loosing connectivity to primary.



Sharding

- Sharding is a method for distributing data across multiple machines.



Lab 3

- Import inventory database.
- Display number of products per category.
- Display max category products price.
- Display user ahmed orders populated with product.
- Get user ahemd highest order price.