Chapter 3: Core Aggregation - Combining Information

Lab: $graphLookup

Back to the Question

For this lab the correct answer would be

COPY

```
db.air_alliances.aggregate([
  {
    $match: { name: "OneWorld" }
  },
  {
    $graphLookup: {
      startWith: "$airlines",
      from: "air_airlines",
      connectFromField: "name",
      connectToField: "name",
      as: "airlines",
      maxDepth: 0,
      restrictSearchWithMatch: {
        country: { $in: ["Germany", "Spain", "Canada"] }
      }
    }
  },
  {
    $graphLookup: {
      startWith: "$airlines.base",
      from: "air_routes",
      connectFromField: "dst_airport",
      connectToField: "src_airport",
      as: "connections",
      maxDepth: 1
    }
  },
  {
    $project: {
      validAirlines: "$airlines.name",
      "connections.dst_airport": 1,
      "connections.airline.name": 1
    }
  },
  { $unwind: "$connections" },
  {
    $project: {
      isValid: {
        $in: ["$connections.airline.name", "$validAirlines"]
      },
      "connections.dst_airport": 1
    }
  },
  { $match: { isValid: true } },
  {
    $group: {
      _id: "$connections.dst_airport"
    }
  }
])
```

This pipeline takes the most selective collection first, **air_alliances**, matching the document refering to the *OneWorld* alliance.

COPY

```
db.air_alliances.aggregate([
{
  $match: { name: "OneWorld" }
}
```

It then iterates, with `maxDepth` 0 on the **air_airlines** collection to collect the details on the airlines, specially their base airport, but restricting that `$lookup` to airlines of the requested countries *[Spain, Germany, Canada]*, using `restrictSearchWithMatch`.

COPY

```
{
  $graphLookup: {
    startWith: "$airlines",
    from: "air_airlines",
    connectFromField: "name",
    connectToField: "name",
    as: "airlines",
    maxDepth: 0,
    restrictSearchWithMatch: {
      country: { $in: ["Germany", "Spain", "Canada"] }
    }
  }
}
```

We then iterate over all routes up to maximum of one layover by setting our `maxDepth` to 1. We find all possible destinations when departing from the *base* airport of each carrier by specify **$airlines.base** in `startWith`

COPY

```
{
  $graphLookup: {
    startWith: "$airlines.base",
    from: "air_routes",
    connectFromField: "dst_airport",
    connectToField: "src_airport",
    as: "connections",
    maxDepth: 1
  }
}
```

We now have a document with a field named **connections** that is an array of all routes that are within 1 layover. We use a `$project` here to remove unnecessary information from the documents. We also need to include information about valid airlines that match our initial restriction and the name of the current airline.

COPY

```
{
  $project: {
    validAirlines: "$airlines.name",
    "connections.dst_airport": 1,
    "connections.airline.name": 1
  }
}
```

After this, we'll unwind our **connections** array, and then use `$project` to add a field representing whether this particular route is valid, meaning it is a route flown by one of our desired carriers.

COPY

```
{ $unwind: "$connections" },
{
  $project: {
    isValid: {
      $in: ["$connections.airline.name", "$validAirlines"]
    },
    "connections.dst_airport": 1
  }
}
```

Lastly, we use `$match` to filter out invalid routes, and then `$group` them on the destination.

COPY

```
{ $match: { isValid: true } },
{
  $group: {
    _id: "$connections.dst_airport"
  }
}
```

An important aspect to this pipeline is that the first `$graphLookup` will act as a regular `$lookup` since we are setting a `maxDepth` to zero. The reason why we are taking this approach is due to the match restriction that `$graphLookup` allows, which can make this stage more efficient. Think back to the earlier lab on `$lookup`, can you think of a way to simplify the aggregation using `$graphLookup` instead?

Proceed to next section