

Implementation in NoSql (MongoDB)

Used MongoDB online playground to run below queries (<https://mongoplayground.net/>)

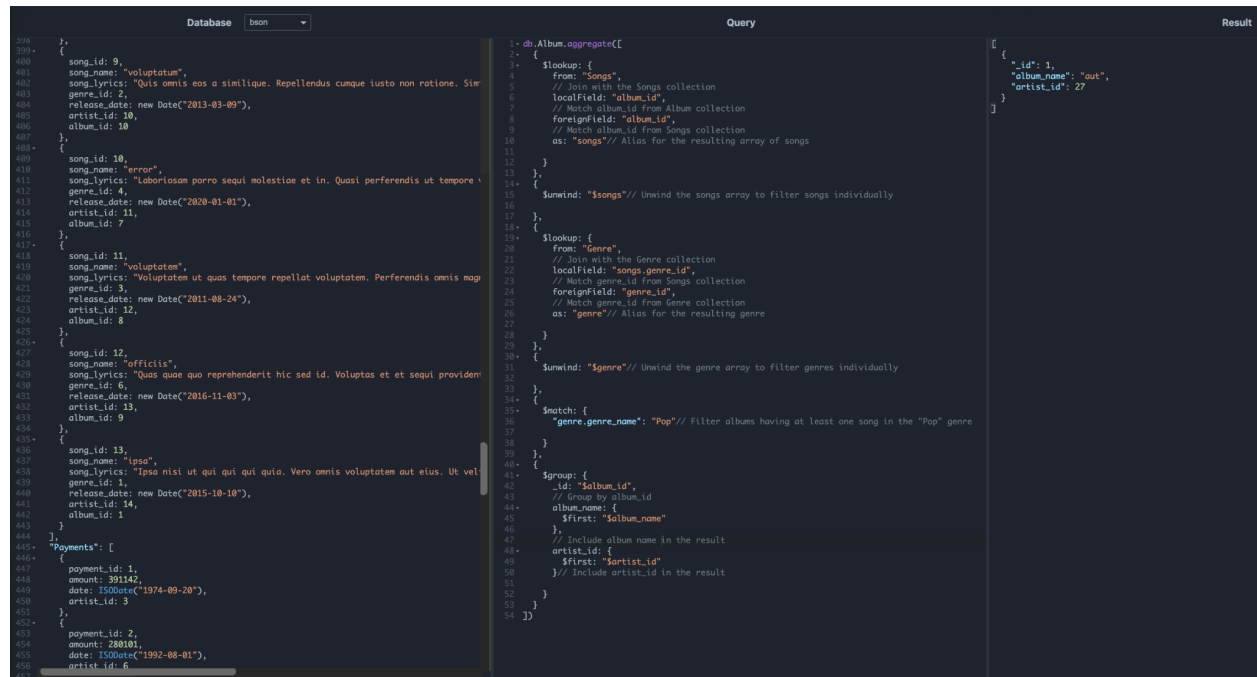
Query 1: Join query (Find albums with at least one song in the 'Pop' genre)

```
db.Album.aggregate([
  {
    $lookup: {
      from: "Songs",
      // Join with the Songs collection
      localField: "album_id",
      // Match album_id from Album collection
      foreignField: "album_id",
      // Match album_id from Songs collection
      as: "songs" // Alias for the resulting array of songs
    }
  },
  {
    $unwind: "$songs" // Unwind the songs array to filter songs individually
  },
  {
    $lookup: {
      from: "Genre",
      // Join with the Genre collection
      localField: "songs.genre_id",
      // Match genre_id from Songs collection
      foreignField: "genre_id",
      // Match genre_id from Genre collection
      as: "genre" // Alias for the resulting genre
    }
  },
  {
    $unwind: "$genre" // Unwind the genre array to filter genres individually
  },
  {
    $match: {
      "genre.genre_name": "Pop" // Filter albums having at least one song in the "Pop"
genre
    }
  },
  {
    $group: {
      _id: "$album_id",
      // Group by album_id
      album_name: {
```

```

        $first: "$album_name"
    },
    // Include album name in the result
    artist_id: {
        $first: "$artist_id"
    } // Include artist_id in the result
}
}
1)

```



Query 2: SimpleQuery (List the song_id, song_name that does not belong to any album)

```

// MongoDB query for documents where album_id is null
db.Songs.find({
  "album_id": null
}),
// Condition to find album_id as null
{
  "song_id": 1,
  "song_name": 1,
  "_id": 0
} // Projection to include only song_id and song_name
)

```

Database	bson	Query	Result
<pre>425 }, 426 { 427 song_id: 12, 428 song_name: "officiis", 429 song_lyrics: "Quas quae quo reprehenderit hic sed id. Volup", 430 genre_id: 6, 431 release_date: new Date("2016-11-03"), 432 artist_id: 13, 433 album_id: 9 434 }, 435 { 436 song_id: 13, 437 song_name: "ipsa", 438 song_lyrics: "Ipsa nisi ut qui qui qui quia. Vero omnis vol", 439 genre_id: 1, 440 release_date: new Date("2015-10-10"), 441 artist_id: 14, 442 album_id: 1 443 }, 444 { 445 song_id: 39, 446 song_name: "odio", 447 song_lyrics: "Eum ratione exercitationem ut nisi et et. Nih", 448 genre_id: 4, 449 release_date: new Date("2020-02-12"), 450 artist_id: 6, 451 album_id: null 452 }, 453 { 454 song_id: 40, 455 song_name: "nemo", 456 song_lyrics: "Ipsam odit ipsa mollitia sed. Enim ea dolorum", 457 genre_id: 5, 458 release_date: new Date("1991-12-07"), 459 artist_id: 10, 460 album_id: null 461 } 462], 463 "Payments": [464 { 465 payment_id: 1, 466 amount: 391142,</pre>	<pre>1 // MongoDB query for documents where album_id is null 2 db.Songs.find({ 3 "album_id": null 4 }, 5 // Condition to find album_id as null 6 { 7 "song_id": 1, 8 "song_name": 1, 9 "_id": 0 10 } // Projection to include only song_id and song_name 11)</pre>	<pre>[{ "song_id": 39, "song_name": "odio" }, { "song_id": 40, "song_name": "nemo" }]</pre>	

Query 3: Nested query (List the artist_name, artist_id having maximum song_count)

// Aggregation pipeline to find artist with maximum song_count

```

db.Songs.aggregate([
  {
    $group: {
      _id: "$artist_id",
      // Group by artist_id
      song_count: {
        $count: {}
      } // Count the number of songs for each artist
    }
  },
  {
    $lookup: {
      from: "Artist",
      // Join with the Artist collection
      localField: "_id",
      // Match artist_id from Songs with artist_id in Artist
      foreignField: "artist_id",
      // Match artist_id in Artist collection
      as: "artist_info" // Add artist information to the result
    }
  },
  {
    $unwind: "$artist_info" // Unwind the artist_info array to access artist_name
  }
])

```

```

    },
    {
      $sort: {
        song_count: -1
      } // Sort by song_count in descending order
    },
    {
      $limit: 1 // Limit to 1 document, the one with the maximum song_count
    },
    {
      $project: {
        _id: 0,
        // Exclude the _id field
        artist_id: "$_id",
        // Include artist_id
        artist_name: "$artist_info.artist_name",
        // Include artist_name from the Artist collection
        song_count: 1 // Include song_count
      }
    }
  }
)

```

Database	Query	Result
<pre> 1 // Aggregation pipeline to find artist with maximum song_count 2 db.Songs.aggregate([3 { 4 \$group: { 5 _id: "\$artist_id", 6 // Group by artist_id 7 song_count: { 8 \$count: {} 9 } 10 } 11 }, 12 { 13 \$lookup: { 14 from: "Artist", 15 // Join with the Artist collection 16 localField: "_id", 17 // Match artist_id from Songs with artist_id in Artist 18 foreignField: "artist_id", 19 // Match artist_id in Artist collection 20 as: "artist_info" // Add artist information to the result 21 } 22 }, 23 { 24 \$unwind: "\$artist_info" // Unwind the artist_info array to access artist_name 25 }, 26 { 27 \$sort: { 28 song_count: -1 29 } // Sort by song_count in descending order 30 }, 31 { 32 \$limit: 1 // Limit to 1 document, the one with the maximum song_count 33 }, 34 { 35 \$project: { 36 _id: 0, 37 // Exclude the _id field 38 artist_id: "\$_id", 39 // Include artist_id 40 artist_name: "\$artist_info.artist_name", 41 // Include artist_name from the Artist collection 42 song_count: 1 // Include song_count 43 } 44 } 45]) </pre>	<pre> 1 // Aggregation pipeline to find artist with maximum song_count 2 db.Songs.aggregate([3 { 4 \$group: { 5 _id: "\$artist_id", 6 // Group by artist_id 7 song_count: { 8 \$count: {} 9 } 10 } 11 }, 12 { 13 \$lookup: { 14 from: "Artist", 15 // Join with the Artist collection 16 localField: "_id", 17 // Match artist_id from Songs with artist_id in Artist 18 foreignField: "artist_id", 19 // Match artist_id in Artist collection 20 as: "artist_info" // Add artist information to the result 21 } 22 }, 23 { 24 \$unwind: "\$artist_info" // Unwind the artist_info array to access artist_name 25 }, 26 { 27 \$sort: { 28 song_count: -1 29 } // Sort by song_count in descending order 30 }, 31 { 32 \$limit: 1 // Limit to 1 document, the one with the maximum song_count 33 }, 34 { 35 \$project: { 36 _id: 0, 37 // Exclude the _id field 38 artist_id: "\$_id", 39 // Include artist_id 40 artist_name: "\$artist_info.artist_name", 41 // Include artist_name from the Artist collection 42 song_count: 1 // Include song_count 43 } 44 } 45]) </pre>	<pre> { "artist_id": 6, "artist_name": "Mr. Ladarus Grimes", "song_count": 3 } </pre>

Query 4: Join Query (List the artist_name of all artists who have payment amount < 1,000,000)

// Aggregation pipeline to join Artist and Payments collections, and filter by total payment amount

```
db.Payments.aggregate([
  {
    $lookup: {
      from: "Artist",
      // Join with the Artist collection
      localField: "artist_id",
      // Match artist_id in Payments with artist_id in Artist
      foreignField: "artist_id",
      // Match artist_id in Artist collection
      as: "artist_info" // Add artist information to the result
    }
  },
  {
    $unwind: "$artist_info" // Unwind the artist_info array to access artist_name
  },
  {
    $group: {
      _id: "$artist_id",
      // Group by artist_id
      total_payment: {
        $sum: "$amount"
      },
      // Sum the payment amounts per artist
      artist_name: {
        $first: "$artist_info.artist_name"
      } // Get the artist name
    }
  },
  {
    $match: {
      total_payment: {
        $lt: 1000000
      } // Filter for artists whose total payment is below 1000000
    }
  },
  {
    $project: {
      _id: 0,
      // Exclude the _id field
      artist_name: 1,
      // Include artist_name
      artist_id: "$_id",
      // Include artist_id from the group
      total_payment: 1 // Include total payment
    }
  }
])
```

```

    }
  },
  {
    $sort: {
      artist_id: 1// Sort by artist_id in ascending order
    }
  }
})

```

Database	Query	Result
<pre> 1 // Aggregation pipeline to join Artist and payments collections, and filter by total payment 2 db.payments.aggregate([3 { 4 \$lookup: { 5 from: "Artist", 6 // Join with the Artist collection 7 localField: "artist_id", 8 // Match artist_id in Payments with artist_id in Artist 9 foreignField: "artist_id", 10 // Match artist_id in Artist collection 11 as: "artist_info" // Add artist information to the result 12 } 13 }, 14 { 15 \$unwind: "\$artist_info" // Unwind the artist_info array to access artist_name 16 }, 17 { 18 \$group: { 19 _id: "\$artist_id", 20 // Group by artist_id 21 total_payment: { 22 \$sum: "\$amount" 23 } 24 }, 25 // Sum the payment amounts per artist 26 artist_name: { 27 \$first: "\$artist_info.artist_name" 28 } // Get the artist name 29 }, 30 { 31 \$match: { 32 total_payment: { 33 \$lt: 1000000 34 } // Filter for artists whose total payment is below 1000000 35 } 36 }, 37 { 38 \$project: { 39 _id: 0, 40 // Exclude the _id field 41 artist_name: 1, 42 // Include artist_name 43 artist_id: "\$_id", 44 // Include artist_id from the group 45 total_payment: 1 // Include total payment 46 } 47 }, 48 { 49 \$sort: { 50 artist_id: 1 // Sort by artist_id in ascending order 51 } 52 } 53]) </pre>	<pre> [{ "artist_id": 16, "artist_name": "Isperanza Dicki", "total_payment": 729779 }, { "artist_id": 22, "artist_name": "Graciela Greenholt PhD", "total_payment": 433843 }, { "artist_id": 24, "artist_name": "Syble Conn", "total_payment": 593971 }] </pre>	