Student ID: 3369172 | University of Stirling
CSCU9B4 - NoSQL Databases

Assignment: AirBnB MongoDB Dataset Queries

Preliminary

It is necessary to describe commands that had not been covered in the lab exercises. All information here comes from the MongoDB documentation.[1]

// $unwind

Outputs a document for each element of an array, where each new document is the input document with the array field replaced by the unwound element.

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

// $geoNear

Provides documents sorted nearest to farthest from a given location. Requires `distanceField` and `near`. The former setting the output field for distance, and the latter being the GeoJSON point from which the distance will be calculated.

```
{ $geoNear: { <geoNear options> } }
```

// $addFields

Adds new fields to documents from the previous stage. An alias for `$set`, but I only found this out later. The new fields are computed from given expressions, such as an `$add` with an array.

```
{ $addFields: { <newField>: <expression>, ... } }
```

// $multiply

Multiplies numbers kept in an array. Returns the value.

```
{ $multiply: [ <expression1>, <expression2>, ... ] }
```

// $in

Given an array, returns a Boolean indicating a value's presence within the array.

```
{ $in: [ <expression>, <array expression> ] }
```

[1] https://www.mongodb.com/docs/manual/reference/operator/aggregation/

Q 1.1

// PROMPT: Write a query that counts the total number of reviews for all properties in each country.

// RATIONALE: $group seemed to be the simplest command to use as a substrate for $sum; $project then was the way to isolate the total. Of course, I had to look at a sample document to see the fields available.

```
db.listingsAndReviews.aggregate([
{
    $group:
      {
        _id: {},
        total_reviews: {
          $sum: "$number_of_reviews",
        },
      },
  },
  {
    $project:
      {
        total_reviews: 1,
        _id: 0,
      },
  },
])
```

```
// // // OUTPUT // // //

[ { total_reviews: 153354 } ]

// // // ////// // // //
```

Q 1.2

// PROMPT: What are the 5 amenities that are most often found in Australian properties?

// RATIONALE: We want only documents with their country set to Australia. These documents can be limited to just amenities by projection. This allows for easily unwinding the array without dealing with excessive detail. It is then possible to group and count the array elements, sorting them into counted groups that can be sorted. The top five can then be selected by merely limiting the sorted documents.

```
db.listingsAndReviews.aggregate([
  {
    $match:
      {
        "address.country": "Australia",
      },
  },
  {
    $project:
      {
        amenities: 1,
        _id: 0,
      },
  },
  {
    $unwind:
      {
        path: "$amenities",
      },
  },
  {
    $group:
      {
        _id: "$amenities",
        count: {
          $count: {},
        },
      },
  },
  {
    $sort:
      {
        count: -1,
      },
  },
  {
    $limit:
      5,
  }
])

// // // OUTPUT // // //

[
  { _id: 'Wifi', count: 579 },
  { _id: 'Essentials', count: 576 },
  { _id: 'Kitchen', count: 567 },
  { _id: 'Washer', count: 555 },
  { _id: 'Smoke detector', count: 535 }
]

// // // ////// // // //
```

Q 1.3

// PROMPT: Count how many properties are around point [-73.95018, 40.66503] within a 1km radius.

// RATIONALE: MongoDB advertises itself as having a geospatial query feature, so I went to the docs and checked how to use it. This command is pretty much perfect for the prompt, giving me all the relevant documents. All I had to do was count them.

```
db.listingsAndReviews.aggregate([
  {
    $geoNear:
      {
        near: {
          type: "Point",
          coordinates: [-73.95018, 40.66503],
        },
        distanceField: "distance",
        maxDistance: 1000,
      },
  },
  {
    $count:
      "total_properties_count",
  },
])

// // // OUTPUT // // //

[ { total_properties_count: 10 } ]

// // // ////// // // //
```

Q 1.4

// PROMPT: List 5 properties that are closest to the point [-73.95018, 40.66503] in descending distance order.

// RATIONALE: Similar rationale for geoNear as the previous question. We project to just show the requested fields, limit the output to 5, then sort them in descending order because geoNear already outputs an ascending order array.

```
db.listingsAndReviews.aggregate([
  {
    $geoNear: {
      near: {
        type: "Point",
        coordinates: [-73.95018, 40.66503],
      },
      distanceField: "distance",
    },
  },
  {
    $project: {
      _id: 1,
      distance: 1,
    },
  },
  {
    $limit: 5,
  },
  {
    $sort: {
      distance: -1,
    },
  },
])

// // // OUTPUT // // //

[
  { _id: '20834832', distance: 661.4033134813212 },
  { _id: '9310959', distance: 305.476512803001 },
  { _id: '19828845', distance: 119.65094319432859 },
  { _id: '4952989', distance: 93.69054890961901 },
  { _id: '17822462', distance: 8.443892816939353 }
]

// // // ////// // // //
```

Q 2.1

// PROMPT: From the set of houses fit for 6-8 people and a swimming pool or waterfront amenity, calculate four averages and come to a conclusion about the ideal vacation spot.

// RATIONALE: We acquire only the necessary documents. Those are the houses that fit at least 6 people and at most 8. We assume the "accommodates" field suffices, rather than dealing with guests_included or the number of beds, since their meaning is unknown; checking the AirBnB site yields information that suggests them to not incur extra charges no matter how they are defined.

Beachfront and Waterfront must be included because some beachfronts are not considered waterfront, but still have access to water. We sort by the lowest average price first, since the job market is in a poor state as of writing. Portugal has more options and a significantly lower price, so it seems like the clear winner.

```
db.listingsAndReviews.aggregate([
  {
    $match: {
      accommodates: {
        $gte: 6,
        $lte: 8,
      },
      amenities: {
        $in: [
          "Waterfront",
          "Beachfront",
          "Lake access",
          "Swimming pool",
        ],
      },
      "address.country": {
        $in: ["Spain", "Portugal"],
      },
    },
  },
  {
    $group: {
      _id: "$address.country",
      totalHomes: {
        $sum: 1,
      },
      avgPrice: {
        $avg: "$price",
      },
      avgAmenities: {
        $avg: {
          $size: "$amenities",
        },
      },
      avgReview: {
        $avg: "$review_scores.review_scores_rating",
      },
    },
  },
  {
    $sort: {
      avgPrice: 1,
    },
  },
])
```

```
// // // OUTPUT // // //

[
  {
    _id: 'Portugal',
    totalHomes: 14,
    avgPrice: Decimal128('104.85714285714285714285714285714429'),
    avgAmenities: 41.214285714285715,
    avgReview: 95.61538461538461
  },
  {
    _id: 'Spain',
    totalHomes: 4,
    avgPrice: Decimal128('161.75'),
    avgAmenities: 26.5,
    avgReview: 100
  }
]

// // // ////// // // //
```

Q 3.1

// PROMPT: Compute a score from various fields for each house, and then show the top 5 according to that score.

// RATIONALE: I foremost want to travel around Canada. I only want to entertain locations that cost less than or equal to 100 dollars a night. There must be some sort of internet connection. I think the overall review, cleanliness, and communication matter among the reviews. I only want listings that others have reviewed, since myScore is based upon reviews.

I scale the reviews and the price together for myScore. If a price is at the max value, it adds (100 * -0.1) = -10 to the score. The maximum score of the reviews is 30, so even if we have perfect reviews, a max price house will have a minimum score of (100 * -0.1)+0=20 and a maximum of (100 * -0.1)+30=20. The lowest possible value is thus -10 and the highest possible value is 30. $addFields and $multiply are perfect for this, and can be found within the MongoDB documentation. $addFields is necessary to add the field itself onto the document. $multiply can be used within sum to provide values to be summed.

I display all amenities along with reviews in the output, and the id, because they are essential information in the matter of choosing (and finding, once chosen) a listing. I also included the name of the listing as requested.

```
db.listingsAndReviews.aggregate([
  {
    $match: {
      price: {
        $lte: 100,
      },
      number_of_reviews: {
        $gte: 1,
      },
      amenities: {
        $in: ["Wifi", "Internet"],
      },
      "address.country": {
        $in: ["Canada"],
      },
    },
  },
  {
    $addFields: {
      myScore: {
        $sum: [
          {
            $multiply: [-0.1, "$price"],
          },
          {
            $multiply: [
              1.1,
              "$review_scores.review_scores_cleanliness",
            ],
          },
          {
            $multiply: [
              1.05,
              "$review_scores.review_scores_communication",
            ],
          },
```

```
          {
            $multiply: [
              1.08,
              "$review_scores.review_scores_location",
            ],
          },
        ],
      },
    },
  },
  {
    $project: {
      myScore: 1,
      price: 1,
      amenities: 1,
      review_scores: 1,
      name: 1,
      _id: 1,
    },
  },
  {
    $sort: {
      myScore: -1,
    },
  },
  {
    $limit: 5,
  },
])

// // // OUTPUT // // //

[
  {
    _id: '19904358',
    name: 'Good room',
    amenities: [
      'TV',
      'Wifi',
      'Kitchen',
      'Free street parking',
      'Washer',
      'Dryer',
      'Smoke detector',
      'Carbon monoxide detector',
      'Essentials',
      'Lock on bedroom door',
      'Hot water',
      'Host greets you'
    ],
    price: Decimal128('13.00'),
    review_scores: {
      review_scores_accuracy: 9,
      review_scores_cleanliness: 10,
      review_scores_checkin: 10,
      review_scores_communication: 10,
      review_scores_location: 10,
      review_scores_value: 10,
      review_scores_rating: 100
    },
    myScore: Decimal128('31.000000000000000710542735760100l9')
  },
  {
    _id: '24917821',
```

```
  name: 'Chambre privée au coeur de Rosemont',
  amenities: [
    'TV',
    'Wifi',
    'Kitchen',
    'Free street parking',
    'Heating',
    'Dryer',
    'Smoke detector',
    'Safety card',
    'Essentials',
    'Hangers',
    'Iron',
    'Laptop friendly workspace',
    'Microwave',
    'Coffee maker',
    'Refrigerator',
    'Dishes and silverware',
    'Cooking basics',
    'Oven',
    'Stove',
    'Patio or balcony',
    'Wide hallway clearance',
    'Step-free access',
    'Step-free access'
  ],
  price: Decimal128('16.00'),
  review_scores: {
    review_scores_accuracy: 10,
    review_scores_cleanliness: 10,
    review_scores_checkin: 10,
    review_scores_communication: 10,
    review_scores_location: 10,
    review_scores_value: 10,
    review_scores_rating: 100
  },
  myScore: Decimal128('30.700000000000000710542735760100l9')
},
{
  _id: '28135558',
  name: 'Lux Studio, Olympic Beauty, King size bed,WiFi,A/C',
  amenities: [
    'TV',
    'Wifi',
    'Air conditioning',
    'Kitchen',
    'Heating',
    'Washer',
    'Dryer',
    'Smoke detector',
    'Carbon monoxide detector',
    'Safety card',
    'Fire extinguisher',
    'Essentials',
    'Shampoo',
    'Hangers',
    'Hair dryer',
    'Iron',
    'Private entrance'
  ],
  price: Decimal128('20.00'),
  review_scores: {
    review_scores_accuracy: 10,
    review_scores_cleanliness: 10,
```

```
    review_scores_checkin: 10,
    review_scores_communication: 10,
    review_scores_location: 10,
    review_scores_value: 10,
    review_scores_rating: 100
  },
  myScore: Decimal128('30.300000000000000710542735760100l9')
},
{
  _id: '27720921',
  name: 'Serene in its Simplicity! (Room 1)',
  amenities: [
    'TV',
    'Wifi',
    'Kitchen',
    'Free parking on premises',
    'Indoor fireplace',
    'Heating',
    'Washer',
    'Dryer',
    'Smoke detector',
    'Carbon monoxide detector',
    'First aid kit',
    'Fire extinguisher',
    'Essentials',
    'Shampoo',
    'Hangers',
    'Hair dryer',
    'Iron',
    'Laptop friendly workspace',
    'Self check-in',
    'Smart lock',
    'Bathtub',
    'Baby bath',
    'Children’s books and toys',
    'Pack ’n Play/travel crib',
    'Room-darkening shades',
    'Children’s dinnerware',
    'Hot water',
    'Bed linens',
    'Extra pillows and blankets',
    'Microwave',
    'Coffee maker',
    'Refrigerator',
    'Dishwasher',
    'Dishes and silverware',
    'Cooking basics',
    'Oven',
    'Stove',
    'Long term stays allowed',
    'Wide hallway clearance',
    'Wide doorway',
    'Wide doorway',
    'Accessible-height bed',
    'Handheld shower head'
  ],
  price: Decimal128('20.00'),
  review_scores: {
    review_scores_accuracy: 10,
    review_scores_cleanliness: 10,
    review_scores_checkin: 10,
    review_scores_communication: 10,
    review_scores_location: 10,
    review_scores_value: 10,
```

```
      review_scores_rating: 99
    },
    myScore: Decimal128('30.300000000000000710542735760100l9')
  },
  {
    _id: '21368282',
    name: 'Modestly Waverly',
    amenities: [
      'Wifi',
      'Kitchen',
      'Heating',
      'Washer',
      'Dryer',
      'Smoke detector',
      'First aid kit',
      'Fire extinguisher',
      'Essentials',
      'Lock on bedroom door',
      'Hangers',
      'Private living room'
    ],
    price: Decimal128('20.00'),
    review_scores: {
      review_scores_accuracy: 10,
      review_scores_cleanliness: 10,
      review_scores_checkin: 10,
      review_scores_communication: 10,
      review_scores_location: 10,
      review_scores_value: 10,
      review_scores_rating: 100
    },
    myScore: Decimal128('30.300000000000000710542735760100l9')
  }
]
// // // ////// // // //
```