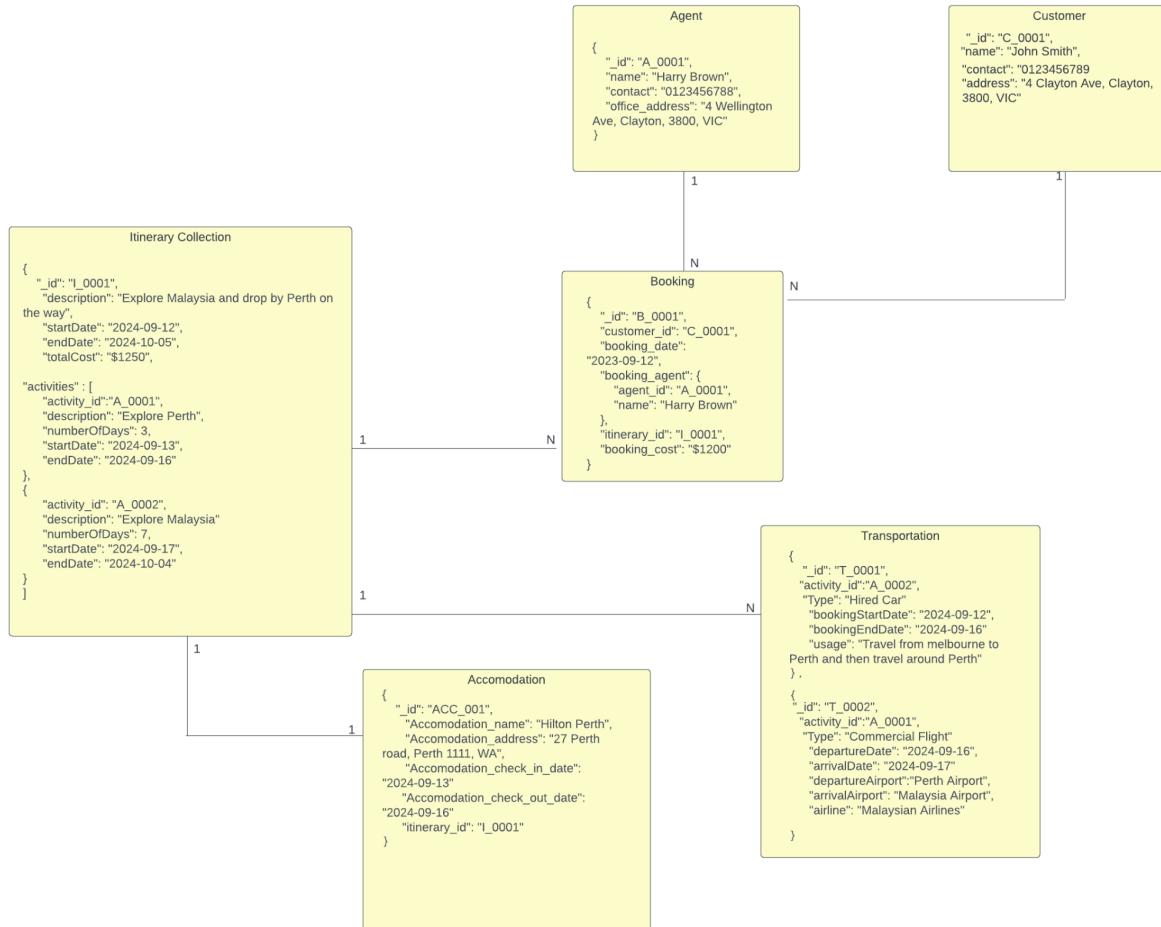


TaskA:
Data model for Monash Travel Agency (MTA)



Customer:

localhost:27017

alee0050_TaskA.Customer

0 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options

ADD DATA EXPORT DATA

```
_id: "C_0001"
name: "John Smith"
address: "4 Clayton Ave, Clayton, 3800, VIC"
contact: "0123456789"
```

1-1 of 1 ⏪ ⏩ ⏴ ⏵ ⏷ ⏸ ⏹

My Queries Databases Search alee0050_TaskA...

- CIBarChart
- C2
- C3
- C4
- parks
- wildlife
- alee0050_TaskA
 - Accommodation
 - Agent
 - Booking
 - Customer**
 - Itinerary
 - Transportation
 - booking
- config
- dbtest
- labweek7app
- libDB
- local
- sample_restaurants

Agent:

localhost:27017

alee0050_TaskA.Agent

0 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options

ADD DATA EXPORT DATA

```
_id: "A_0001"
name: "Harry Brown"
contact: "0123456789"
office_address: "4 Wellington Ave, Clayton, 3800, VIC"
```

1-1 of 1 ⏪ ⏩ ⏴ ⏵ ⏷ ⏸ ⏹

My Queries Databases Search alee0050_TaskA...

- CIBarChart
- C2
- C3
- C4
- parks
- wildlife
- alee0050_TaskA
 - Accommodation
 - Agent**
 - Booking
 - Customer
 - Itinerary
 - Transportation
 - booking
- config
- dbtest
- labweek7app
- libDB
- local
- sample_restaurants

Booking:

localhost:27017

alee0050_TaskA.Booking

0 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options ↗

ADD DATA EXPORT DATA ↗

```
_id: "B_0001"
customer_id: "C_0001"
booking_date: "2023-09-12"
booking_agent: Object
itinerary_id: "I_0001"
booking_cost: "$1200"
```

1-1 of 1 ⏪ ⏩ ⏴ ⏵ ⏷ ⏸

My Queries **Databases** Search alee0050_TaskA... +

- C1BarChart
- C2
- C3
- C4
- parks
- wildlife
- alee0050_TaskA
 - Accommodation
 - Agent
 - Booking** ...
 - Customer
 - Itinerary
 - Transportation
 - booking
- config
- dbtest
- labweek7app
- libDB
- local
- sample_restaurants

Itinerary:

localhost:27017

alee0050_TaskA.Itinerary

0 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options ↗

ADD DATA EXPORT DATA ↗

```
_id: "I_0001"
description: "Explore Malaysia and drop by Perth on the way"
startDate: "2024-09-12"
endDate: "2024-10-05"
totalCost: "$1250"
activities: Array (2)
  ▾ 0: Object
    activity_id: "A_0001"
    description: "Explore Perth"
    numberOfDays: 3
    startDate: "2024-09-13"
    endDate: "2024-09-16"
  ▾ 1: Object
    activity_id: "A_0002"
    description: "Explore Malaysia"
    numberOfDays: 7
    startDate: "2024-09-17"
    endDate: "2024-10-04"
```

1-1 of 1 ⏪ ⏩ ⏴ ⏵ ⏷ ⏸

My Queries **Databases** Search alee0050_TaskA... +

- C1BarChart
- C2
- C3
- C4
- parks
- wildlife
- alee0050_TaskA
 - Accommodation
 - Agent
 - Booking
 - Customer
 - Itinerary** ...
 - Transportation
 - booking
- config
- dbtest
- labweek7app
- libDB
- local
- sample_restaurants

Accommodation:

localhost:27017 ...

My Queries Databases Search alee0050_TaskA...

alee0050_TaskA.Accommodation

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options ↗

ADD DATA EXPORT DATA ↗

1-1 of 1 ⏪ ⏩ ⏴ ⏵ ⏷ ⏸ ⏹

```
_id: "ACC_001"
Accommodation_name: "Hilton Perth"
Accommodation_address: "27 Perth road, Perth 1111, WA"
Accommodation_check_in_date: "2024-09-13"
Accommodation_check_out_date: "2024-09-16"
itinerary_id: "I_0001"
```

0 DOCUMENTS 1 INDEXES

Accommodation ...

- CIBarChart
- C2
- C3
- C4
- parks
- wildlife
- alee0050_TaskA
- Accommodation ...
- Agent
- Booking
- Customer
- Itinerary
- Transportation
- booking
- config
- dbtest
- labweek7app
- libDB
- local
- sample_restaurants

Transportation:

localhost:27017 ...

My Queries Databases Search alee0050_TaskA...

alee0050_TaskA.Transportation

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options ↗

ADD DATA EXPORT DATA ↗

1-2 of 2 ⏪ ⏩ ⏴ ⏵ ⏷ ⏸ ⏹

```
_id: "T_0001"
activity_id: "A_0002"
Type: "Hired Car"
bookingStartDate: "2024-09-12"
bookingEndDate: "2024-09-16"
usage: "Travel from melbourne to Perth and then travel around Perth"
```

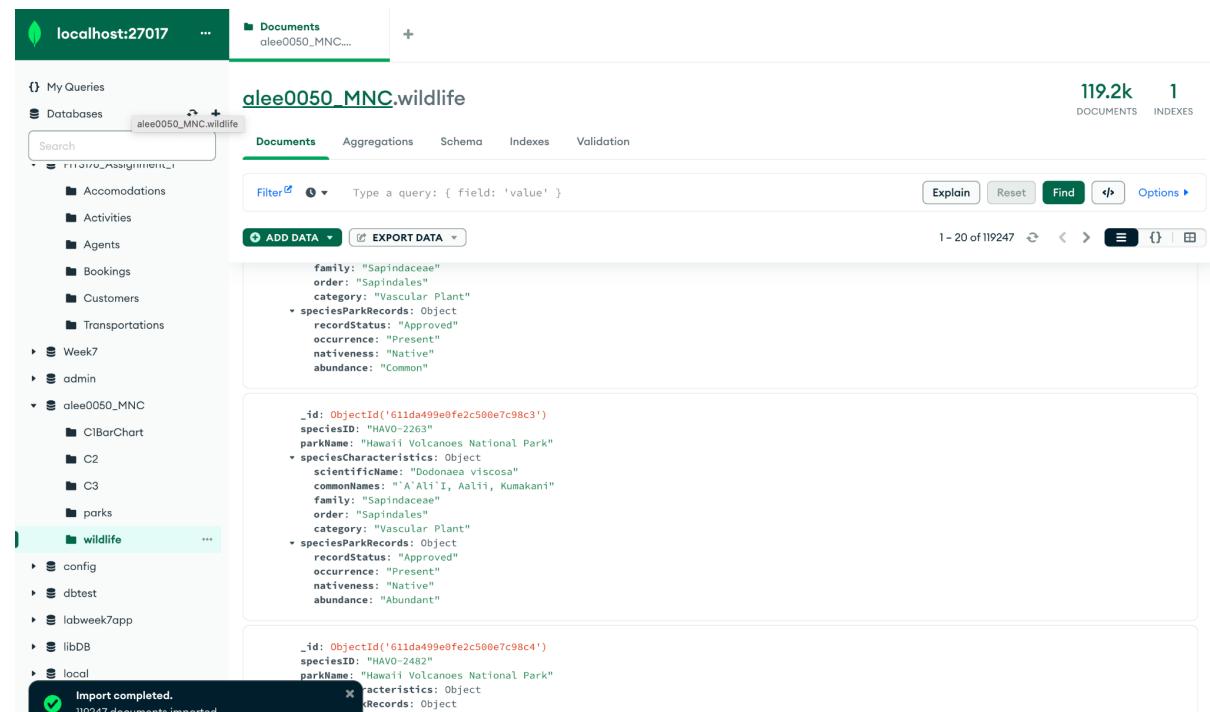
```
_id: "T_0002"
activity_id: "A_0001"
Type: "Commercial Flight"
departureDate: "2024-09-16"
arrivalDate: "2024-09-17"
departureAirport: "Perth Airport"
arrivalAirport: "Malaysia Airport"
airline: "Malaysian Airlines"
```

0 DOCUMENTS 1 INDEXES

Transportation ...

- CIBarChart
- C2
- C3
- C4
- parks
- wildlife
- alee0050_TaskA
- Accommodation
- Agent
- Booking
- Customer
- Itinerary
- Transportation ...
- booking
- config
- dbtest
- labweek7app
- libDB
- local
- sample_restaurants

Task B1.1:



The screenshot shows the MongoDB Compass interface. The top navigation bar includes 'localhost:27017', 'Documents', and '+'. On the right, it shows '119.2k DOCUMENTS' and '1 INDEXES'. The main area is titled 'alee0050_MNC.wildlife' and shows the 'Documents' tab selected. A search bar at the top says 'Type a query: { field: 'value' }'. Below it are 'ADD DATA' and 'EXPORT DATA' buttons. The document list displays several entries, with the first one expanded to show its internal structure:

```

family: "Sapindaceae"
order: "Sapindales"
category: "Vascular Plant"
+ speciesParkRecords: Object
  recordStatus: "Approved"
  occurrence: "Present"
  nativeness: "Native"
  abundance: "Common"

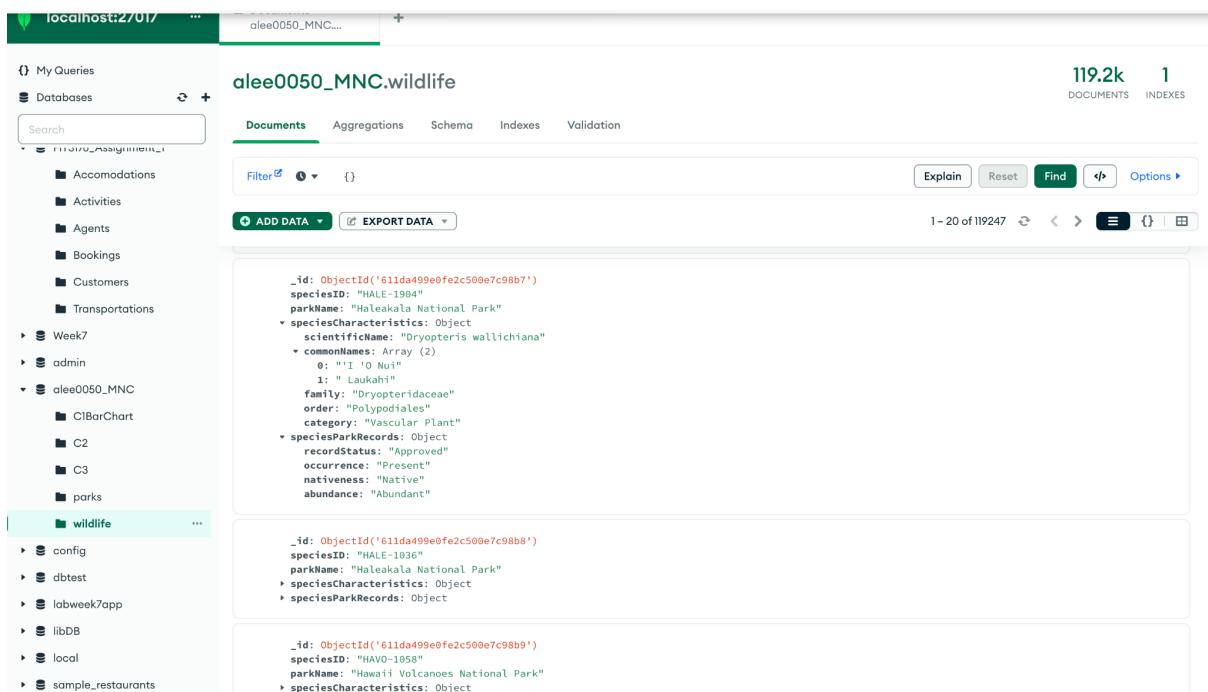
```

The sidebar on the left lists various databases and collections, including 'Accommodations', 'Activities', 'Agents', 'Bookings', 'Customers', 'Transportations', 'Week7', 'admin', and 'alee0050_MNC' which contains 'C1BarChart', 'C2', 'C3', 'parks', and 'wildlife'. The 'wildlife' collection is currently selected. At the bottom of the interface, there is an 'Import completed.' message.

```

db.wildlife.updateMany( //updateMany just modifies all documents
  {"speciesCharacteristics.commonNames": {$regex: ","}}, //regular expression
  //is a form of pattern matching to check if the token is equals to ,
  [
    {
      $set: { // $set operator is used to set the data, in this context the
        // the subarray of splitted commas, will be displayed here
        "speciesCharacteristics.commonNames": {
          $split: ["$speciesCharacteristics.commonNames", ","]
        }
      }
    }
  ]
)
  
```

```
alee0050_MNC> db.wildlife.updateMany( //updateMany just modifies all documents
...     {"speciesCharacteristics.commonNames": {$regex: ","}}, //regular expression
...     //is a form of pattern matching to check if the token is equals to ,
...     [
...         {
...             $set: { // $set operator is ussed to set the data, in this context the
...                   //the subarray of splitted commas, will be displayed here
...                   "speciesCharacteristics.commonNames": {
...                       $split: ["$speciesCharacteristics.commonNames", ","]
...                   }
...             }
...         }
...     ]
... )
{
    acknowledged: true,
    insertedId: null,
    matchedCount: 21611,
    modifiedCount: 21611,
    upsertedCount: 0
}
```



The screenshot shows the MongoDB Compass interface. On the left, there's a sidebar with a tree view of databases and collections. The 'wildlife' collection is selected. The main area shows three documents from the 'wildlife' collection. Each document has an '_id' field, a 'speciesID' field, a 'parkName' field, a 'speciesCharacteristics' object, and a 'speciesParkRecords' object. The 'speciesCharacteristics' object contains a 'commonNames' array with two elements: 'I 'O Nui' and 'Laukahi'. The 'speciesParkRecords' object contains fields like 'recordStatus', 'occurrence', 'nativeness', and 'abundance'.

Task B1.2:

```
db.wildlife.find().forEach(function(ts) { //iterate through each of the documents
    db.wildlife.updateMany( //update all for each of the documents taking in the id
        { _id: ts._id }, // _id reference to the each of the documents id in wildlife
        { $set: { ts: ts._id.getTimestamp() } } //get the timestamp from the id
    );
    print("Timestamp: " + ts._id.getTimestamp()); //print that out, debugging purpose
});
```

Using print statements, we are able to accurately display what is being printed, and being set under ts

The screenshot shows the MongoDB Compass interface with the following details:

- Top Bar:** Shows "localhost:27017" and a "Documents" section for "alee0050_MNC...".
- Left Sidebar:** Lists various databases and collections, including "Accommodations", "Activities", "Agents", "Bookings", "Customers", "Transportations", "Week7", "admin", "alee0050_MNC" (which is expanded to show "C1BarChart", "C2", "C3", "parks", and "wildlife"), "config", "dbtest", "labweek7app", "libDB", "local", and "sample_restaurants".
- Main Area:** The "wildlife" collection is selected. It contains two documents, each representing a plant species record with fields like "_id", "speciesID", "parkName", "speciesCharacteristics", "speciesParkRecords", and "ts".
- Header Bar:** Includes "119.2k DOCUMENTS", "1 INDEXES", "Filter", "ADD DATA", "EXPORT DATA", "Explain", "Reset", "Find", "Options", and navigation icons.
- Bottom Bar:** Shows document IDs: "611da499e0fe2c500e7c98b2", "611da499e0fe2c500e7c98b3", and "611da499e0fe2c500e7c98b4".

Task B1.3:

localhost:27017 ... Documents aleee0050_MNC.p...

My Queries Databases Search aleee0050_MNC.parks

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find Options ▾

ADD DATA EXPORT DATA 1–20 of 56 ⏪ ⏴ ⏵ ⏷ ⏸ ⏹ ⏺

Import completed. 5 documents imported.

`db.parks.updateMany({}, {$set: { dateEst: { $dateFromString: { $dateString: { $concat: [{ $toString: "$sparkEstYear"}, "-", { $toString: "$sparkEstMonth"}, "-", { $toString: "$sparkEstDay"}] } } } } }, {$unset: ["parkEstYear", "parkEstMonth", "parkEstDay"] })`

`{ acknowledged: true, insertedId: null, matchedCount: 56, modifiedCount: 56, upsertedCount: 0 }`

`alee0050_MNC> db.parks.find()`

```

{
  "_id": ObjectId("611da00be0fe2c509e7c987a"),
  "areaInAcres": 47390,
  "latitude": 44.35,
  "longitude": -68.21,
  "parkEstDay": 26,
  "parkEstMonth": 2,
  "parkEstYear": 1919,
  "parkName": "Acadia National Park",
  "stateCode": "ME"
}

{
  "_id": ObjectId("611da00be0fe2c509e7c987b"),
  "areaInAcres": 76519,
  "latitude": 38.68,
  "longitude": -109.57,
  "parkEstDay": 12,
  "parkEstMonth": 11,
  "parkEstYear": 1971,
  "parkName": "Arches National Park",
  "stateCode": "UT"
}

{
  "_id": ObjectId("611da00be0fe2c509e7c987c"),
  "areaInAcres": 242756,
  "latitude": 43.75,
  "longitude": -102.5,
  "parkEstDay": 10,
  "parkEstMonth": 11,
  "sparkEstYear": 1978,
  "parkName": "Badlands National Park",
  "stateCode": "SD"
}

```

localhost:27017

alee0050_MNC.parks

56 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Explain Reset Find Options ▾

ADD DATA EXPORT DATA

1 - 20 of 56

```
_id: ObjectId('611da00be0fe2c500e7c987a')
areaInAcres: 47390
latitude: 44.35
longitude: -68.21
parkName: "Acadia National Park"
stateCode: "ME"
dateEst: 1919-02-26T00:00:00.000+00:00
```

```
_id: ObjectId('611da00be0fe2c500e7c987b')
areaInAcres: 76519
latitude: 38.68
longitude: -109.57
parkName: "Arches National Park"
stateCode: "UT"
dateEst: 1971-11-12T00:00:00.000+00:00
```

```
_id: ObjectId('611da00be0fe2c500e7c987c')
areaInAcres: 242756
latitude: 43.75
longitude: -102.5
parkName: "Badlands National Park"
stateCode: "SD"
dateEst: 1978-11-10T00:00:00.000+00:00
```

```
_id: ObjectId('611da00be0fe2c500e7c987d')
s: 801163
t: 29.25
```

Import completed.
56 documents imported.

Task B1.4:

localhost:27017

alee0050_MNC.wildlife

119.2k DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Explain Reset Find Options ▾

ADD DATA EXPORT DATA

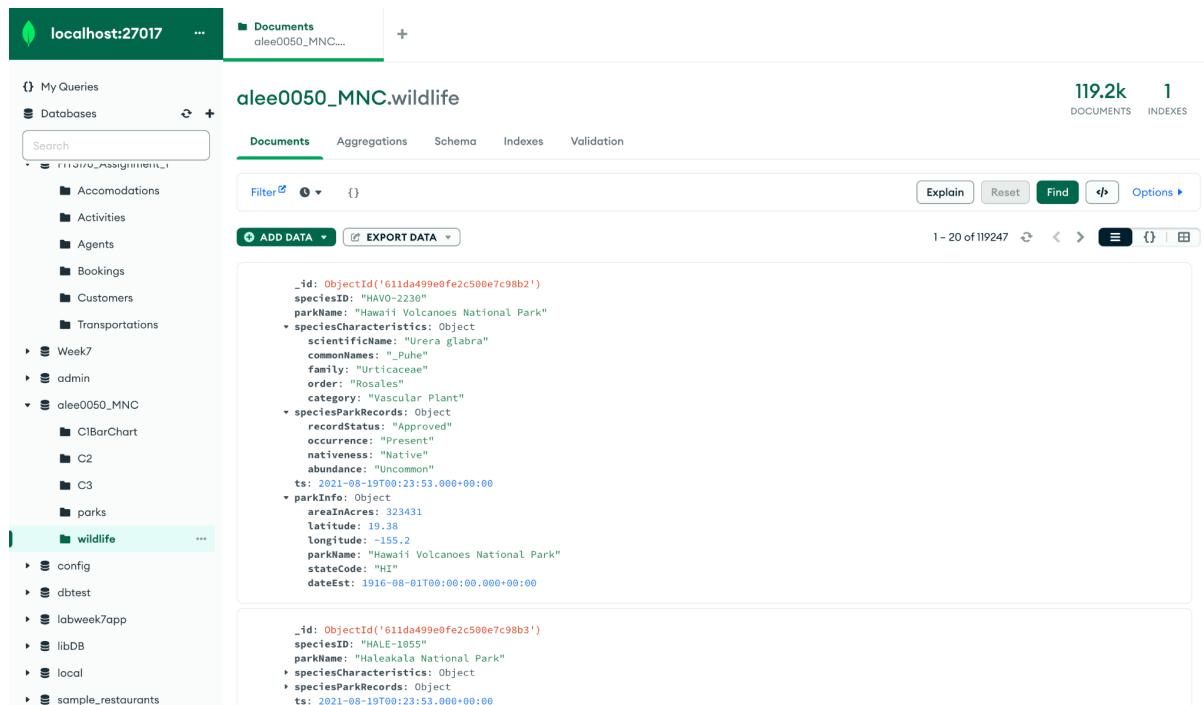
1 - 20 of 119247

```
_id: ObjectId('611da499e0fe2c500e7c98b2')
speciesID: "HAVO-2238"
parkName: "Hawaii Volcanoes National Park"
speciesCharacteristics: Object
scientificName: "Urera glabra"
commonNames: "Puhe"
family: "Urticaceae"
order: "Rosales"
category: "Vascular Plant"
speciesParkRecords: Object
recordStatus: "Approved"
occurrence: "Present"
nativeness: "Native"
abundance: "Uncommon"
ts: 2021-08-19T00:23:53.000+00:00
```

```
_id: ObjectId('611da499e0fe2c500e7c98b3')
speciesID: "HALE-1055"
parkName: "Haleakala National Park"
speciesCharacteristics: Object
speciesParkRecords: Object
ts: 2021-08-19T00:23:53.000+00:00
```

```
_id: ObjectId('611da499e0fe2c500e7c98b4')
speciesID: "HALE-1981"
parkName: "Haleakala National Park"
speciesCharacteristics: Object
speciesParkRecords: Object
ts: 2021-08-19T00:23:53.000+00:00
```

```
db.wildlife.aggregate([
  {
    $lookup: {
      from: "parks", //using the collection parks
      localField: "parkName", //this is like a foreign key concept
      foreignField: "parkName", //using left join needs a foreign key
      as: "parkInfo" //save that as parkinfo, this is saved manually as an array
    }
  },
  {
    $unwind: "$parkInfo" //now we use $unwind to deconstruct an array field to output document for each element
  },
  {
    $addFields: [
      "areaInAcres": "$parkInfo.areaInAcres",
      "latitude": "$parkInfo.latitude",
      "longitude": "$parkInfo.longitude",
      "parkName": "$parkInfo.parkName",
      "dateEst": "$parkInfo.dateEst",
      "stateCode": "$parkInfo.stateCode"
    ]
  },
  {
    $project: {
      "parkInfo._id": 0, //exclude the id in the array that contains park collection of ._id
      "areaInAcres": 0,
      "latitude": 0,
      "longitude": 0,
      "dateEst": 0,
      "stateCode": 0
    }
  },
  {
    $out: "wildlife" //using $out the results of aggregation of previous pipeline is saved into a new collection, or replaced.
  }
]);
```



The screenshot shows the MongoDB Compass interface. The top navigation bar includes 'localhost:27017', 'Documents', and a search bar. The main area displays the 'alee0050_MNC.wildlife' collection. The document count is 119.2k documents and 1 index.

The left sidebar shows the database structure:

- My Queries
- Databases
 - alee0050_MNC
- Search
- Week7
- admin
- alee0050_MNC
 - ClBarChart
 - C2
 - C3
 - parks
 - wildlife
- config
- dbtest
- labweek7app
- libDB
- local
- sample_restaurants

The 'wildlife' document is expanded, showing its fields:

```
_id: ObjectId('611da499e0fe2c500e7c98b2')
speciesID: "HAVO-2230"
parkName: "Hawaii Volcanoes National Park"
speciesCharacteristics: Object
  scientificName: "Urena glabra"
  commonNames: ["Puhe"]
  family: "Urticaceae"
  order: "Rosales"
  category: "Vascular Plant"
speciesParkRecords: Object
  recordStatus: "Approved"
  occurrence: "Present"
  nativeness: "Native"
  abundance: "Uncommon"
  ts: 2021-08-19T00:23:53.000+00:00
parkInfo: Object
  areaInAcres: 323431
  latitude: 19.38
  longitude: -155.2
  parkName: "Hawaii Volcanoes National Park"
  stateCode: "HI"
  dateEst: 1916-08-01T00:00:00.000+00:00

_id: ObjectId('611da499e0fe2c500e7c98b3')
speciesID: "HALE-1055"
parkName: "Haleakala National Park"
speciesCharacteristics: Object
speciesParkRecords: Object
  ts: 2021-08-19T00:23:53.000+00:00
```

Task B1.5:

localhost:27017 ... +

Validation alee0050_MNC....

My Queries Databases Search

alee0050_MNC.wildlife

Documents Aggregations Schema Indexes Validation

119.2k 1

DOCUMENTS INDEXES

Accommodations Activities Agents Bookings Customers Transportations Week7 admin alee0050_MNC C1BarChart C2 C3 parks wildlife config dbtest labweek7app libDB local sample_restaurants

Add validation rules

Create rules to enforce data structure of documents on updates and inserts.

Add Rule Learn more about validations

```

db.runCommand({ //it uses the current DB to run whatever logic that is used below
  collMod: "wildlife", //collMod is used to validate
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["speciesParkRecords"],
      properties: {
        speciesParkRecords: {
          bsonType: "object",
          required: ["recordStatus", "occurrence", "nativeness", "abundance"],
          properties: {
            recordStatus: {
              bsonType: "string",
              enum: [/* */],
              'American Crow',
              'Bluebell',
              'Bushtit',
              'Cabezon',
              'Catbird',
              'Cenizo',
              'Chico',
              'Claret Cup',
              'Clover Bush',
              'Cocodrilo De Tumbes',
              'Common Mullein',
              'Common Poorwill',
              'Cranebill',
              'Dames Rocket',
              'Devil's Shoelaces',
              'Downy Chess',
              'Filaree',
              'Fringed Sage',
              'Golden Pea',
              'Goosefoot',
              'Grass-Leaf Loco',
              'Ground Daisy',
              'Kinnikinnick',
              'Leather Flower',
              'Liver Leaf*',
              'Manati',
              'Northern Goshawk',
              'Northern Pintail',
              'Osha',
              'Pigeon Hawk',
              'Purple Cockle',
              'Ranchers' Fireweed',
              'Robin',
              'Rushpink',
              'Shadbush',
              'Short-Tailed Weasel',
              'Skunkbush',
              'Skyrocket Gilia',
              'Speckled Trout',
              'Speedwell',
              'Storksbill',
              'Verdolagas',
              'Wapiti',
              'White-Footed Mouse',
              'Whortleberry',
              'Wild Iris',
              'Wild Rose',
              'Willowherb',
              'Wiregrass',
              'Approved',
              'In Review',
              'None',
              'P.Nut Sedge'],
              description: "recordStatus not valid"
            },
            occurrence: {
              bsonType: "string",
              enum: ['Approved',
              'In Review',
              'Not Confirmed',
              'Not Present',
              'Not Present (False Report)',
              'Not Present (Historical Report)',
              'Present'],
              description: "occurrence not valid"
            },
            nativeness: {
              bsonType: "string",
              enum: [/* */]
            }
          }
        }
      }
    }
  }
}
  
```

```

    "nateness": {
      "bsonType": "string",
      "enum:": ['Native',
      'Not Confirmed',
      'Not Native',
      'Present',
      'Unknown'],
      "description": "nateness not valid"
    },
    "abundance": {
      "bsonType": "string",
      "enum": ['Abundant',
      'Common',
      'Native',
      'Not Native',
      'Occasional',
      'Rare',
      'Uncommon',
      'Unknown'],
      "description": "abundance not valid"
    }
  },
  "validationLevel": "moderate",
  "validationAction": "error"
}

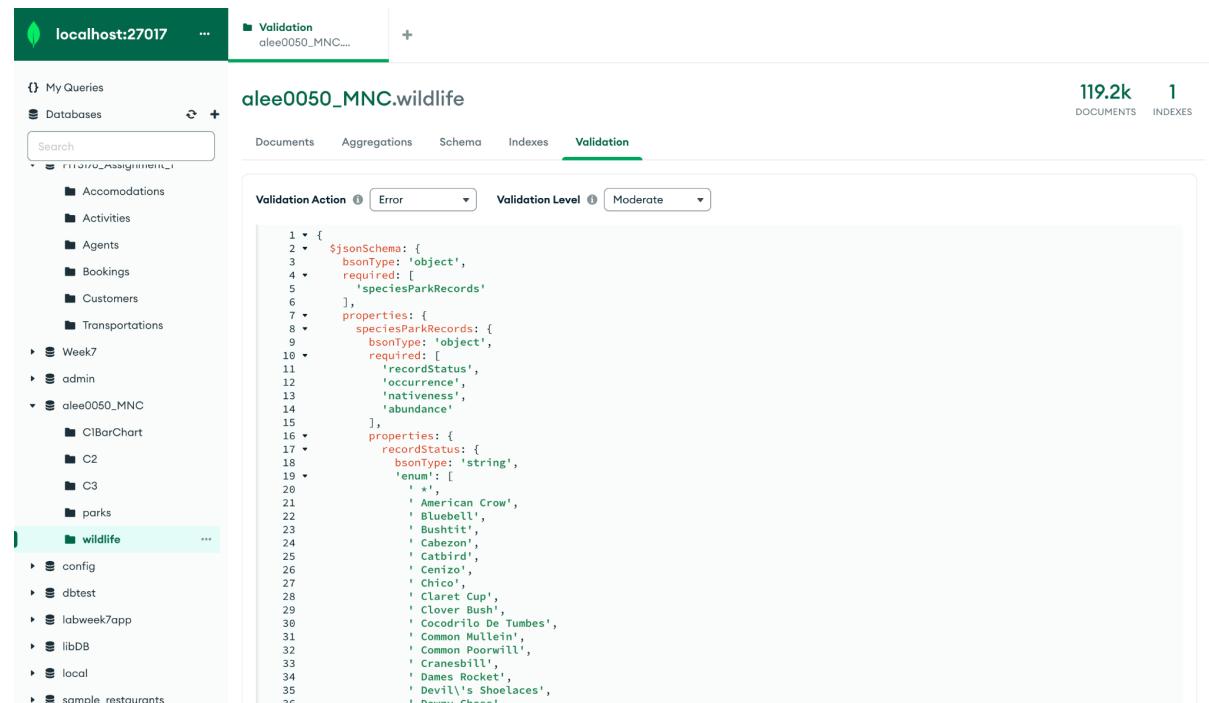
```

Directly copied pasted the enum from command prompt, excuse the formatting.

```

...
}
},
validationLevel: "moderate",
validationAction: "error"
})
{ ok: 1 }

```



The screenshot shows the MongoDB Compass interface. At the top, it says "localhost:27017" and "Validation alee0050_MNC....". Below that, there's a navigation bar with "My Queries", "Databases", "Search", and tabs for "Documents", "Aggregations", "Schema", "Indexes", and "Validation". The "Validation" tab is selected. On the left, there's a sidebar with a tree view of databases and collections, including "wildlife" which is currently selected. The main area displays the validation schema:

```

1 <= {
2   "$jsonSchema": [
3     "bsonType": "object",
4     "required": [
5       "speciesParkRecords"
6     ],
7     "properties": {
8       "speciesParkRecords": {
9         "bsonType": "object",
10      "required": [
11        "recordStatus",
12        "occurrence",
13        "nateness",
14        "abundance"
15      ],
16      "properties": {
17        "recordStatus": {
18          "bsonType": "string",
19          "enum": [
20            "x",
21            "American Crow",
22            "Bluebell",
23            "Bushtit",
24            "Cabezon",
25            "Catbird",
26            "Cenizo",
27            "Chico",
28            "Claret Cup",
29            "Clover Bush",
30            "Cocodrilo De Tumbes",
31            "Common Mullein",
32            "Common Poorwill",
33            "Cranesbill",
34            "Dames Rocket",
35            "Devil's Shoelaces",
36            "Downy Chess"
37          ]
38        }
39      }
40    }
41  }
42}

```

Run db.wildlife.distinct('speciesParkRecords.recordStatus') or any other 3 to find out all the enumerated data types, in order to implement validation.

B2.1:

```
db.wildlife.aggregate([
  {
    $group: {
      _id: {$year: "$parkInfo.dateEst"}, //since we already moved dateEst into parkinfo
      totalParks: {$sum: 1} //1 for true to gather sum of all documents in group
    }
  },
  {
    $sort: {_id: -1} //sort by descending order, this indicates size() - 1, which is the last position
  },
  {
    $project: [
      year: "$_id", //display what year in the grouping
      totalParks: 1, //display totalparks
      _id: 0 //no need for id for the year, since year is already there
    ]
  }
])

```

B2.2:

```
//B2.2
db.wildlife.aggregate([
  {
    $match: {
      "speciesCharacteristics.category": "Bird"
      //match the speciescharacteristics category
      //it can be found using db.wildlife.distinct("speciesCharacteristics.category")
      //this will show all the categories that is in db wildlife
    }
  },
  {
    $group: {
      _id: "$parkInfo.parkName", //group them up by the parkName
      birdCount: {$sum: 1} //count all the birds by adding one incremental, this would indicate for each document it iterates through
      //add 1 to a counter
    }
  },
  {
    $project: {
      parkName: "$_id", //display id as its parkName
      birdCount: 1, //display birdCount
      _id: 0 //don't display ID as there is no need
    }
  }
])

```

B2.3:

```
//B2.3
db.wildlife.updateMany({}, [ //given by the question that we are able to modify our collection

  {
    $set: { //I have opted in to create a field in parkInfo array of GeoJsonLocation
      "parkInfo.GeoJsonLocation": {
        type: "Point", //it is a point, of which longitude and latitude of each record is stored
        coordinates: ["$parkInfo.longitude", "$parkInfo.latitude"]
      }
    }
  }
]

db.wildlife.createIndex({"parkInfo.GeoJsonLocation": "2dsphere"}); //then we create an index for this "2dsphere"
//this will allow us to calculate the distance gathered at one point, and do calculations with radian to find out the distance in one point

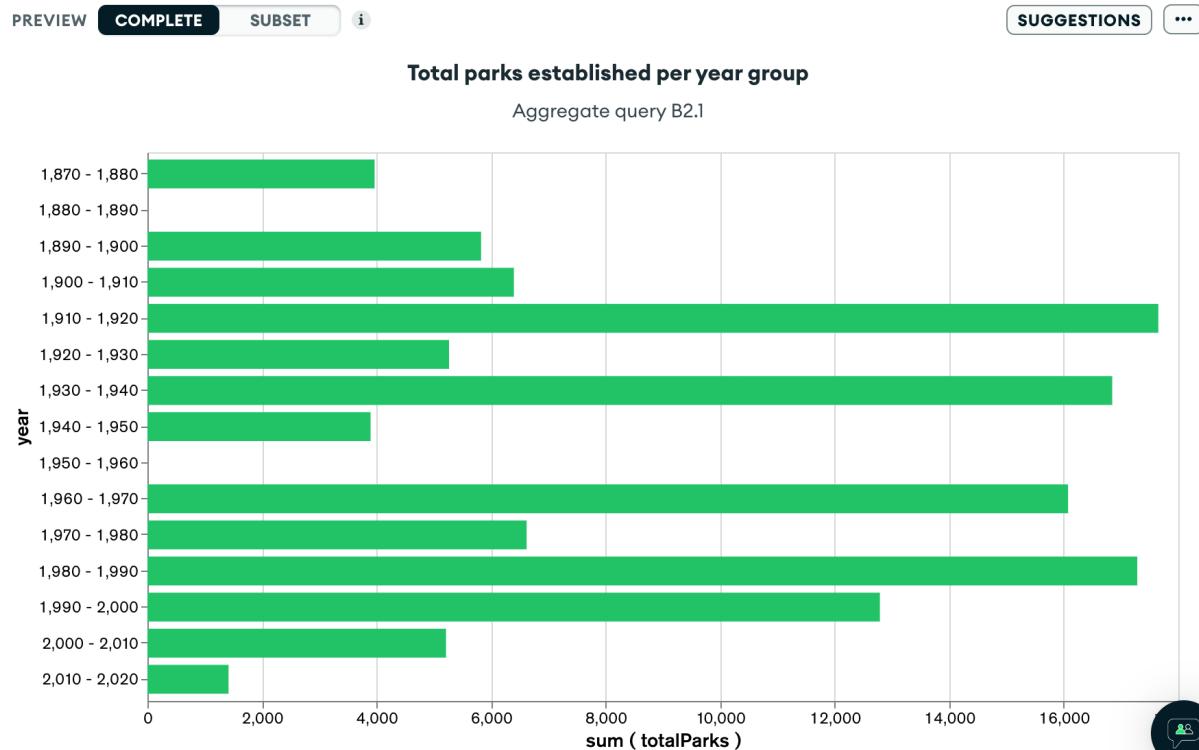
//https://stackoverflow.com/questions/38112376/find-closest-entries-in-mongodb-to-coordinates
db.wildlife.find({
  "parkInfo.GeoJsonLocation": {
    $nearSphere: {
      $geometry: {
        type: "Point",
        coordinates: [-95.8412, 42.1109] //couldn't get this to work
        //using the co-ordinates from example lab, and stack overflow for the query
      },
      $maxDistance: 400000 // 400km is the max distance from point of coords
    }
  }
});
});
```

B2.4:

B2.5:

```
db.wildlife.aggregate([
  {
    $match: {
      "speciesCharacteristics.commonNames": /^a/i //matches everything that starts with "a" case insensitive
    }
  },
  {
    $group: { //group them with parkName, category, order and family
      _id: [
        park: "$sparkName",
        category: "$speciesCharacteristics.category",
        order: "$speciesCharacteristics.order",
        family: "$speciesCharacteristics.family"
      ]
    }
  },
  {
    //https://stackoverflow.com/questions/16368638/mongodb-distinct-aggregation
    //"$addToSet is used to count the number of distinct objects"
    $group: {
      _id: "$_id.park",
      distinctCat: { $addToSet: "$_id.category" }, //addToSet functions like a hashset, where it takes in distinct values, if its duplicate
      //it will discard
      distinctOrder: { $addToSet: "$_id.order" },
      distinctFamily: { $addToSet: "$_id.family" }
    }
  },
  {
    $project: {
      "distinctCatCounter": { "$round": [{ "$size": "$distinctCat" }, 2] }, //round to 2 decimal places
      "distinctOrderCounter": { "$round": [{ "$size": "$distinctOrder" }, 2] }, //size indicates the size of the
      "distinctFamilyCounter": { "$round": [{ "$size": "$distinctFamily" }, 2] }
    }
  }
]).pretty();
```

C.1.1:



Justification:

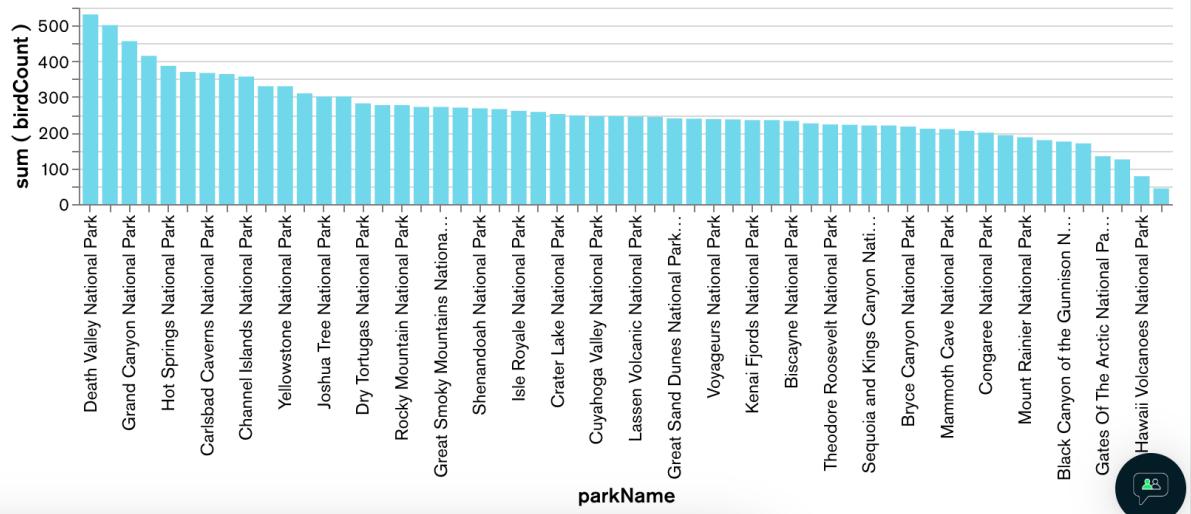
I have used a bar chart for establishing the relationship between total parks and per year groups, this visualisation of a bar chart gives a clear image and indication of how many parks that have been established between a certain specific year group, incremental by 10 years. In my opinion, given its nature, it is not something that requires significant understanding, it is easy to show something that everyone can understand, thus making this data appropriate and easy to access for the public.

C.1.2:

[PREVIEW](#) **COMPLETE** [SUBSET](#) [i](#)
[SUGGESTIONS](#) [...](#)

Average bird Species Per Park

Aggregate query C1.2

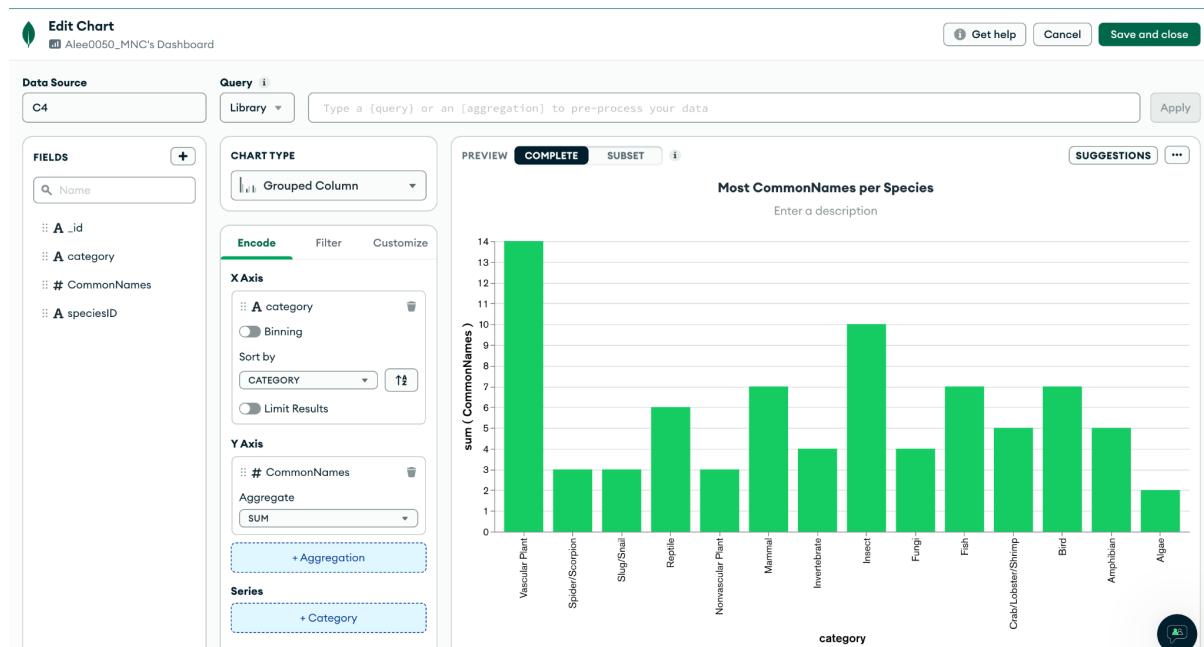


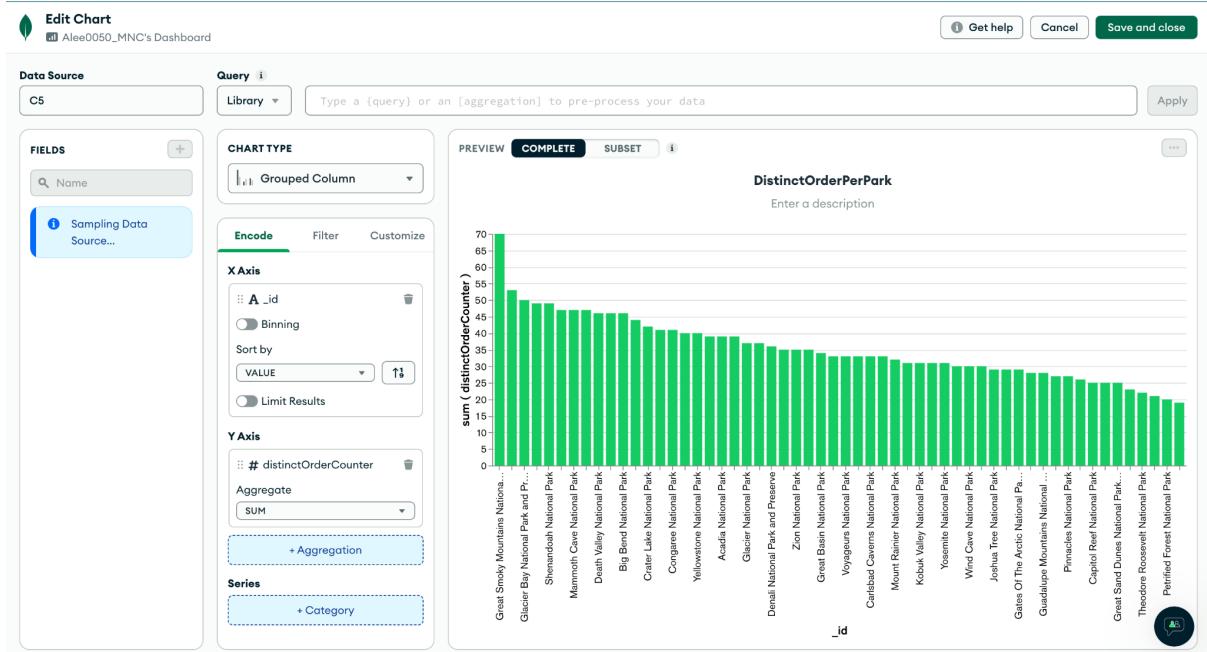
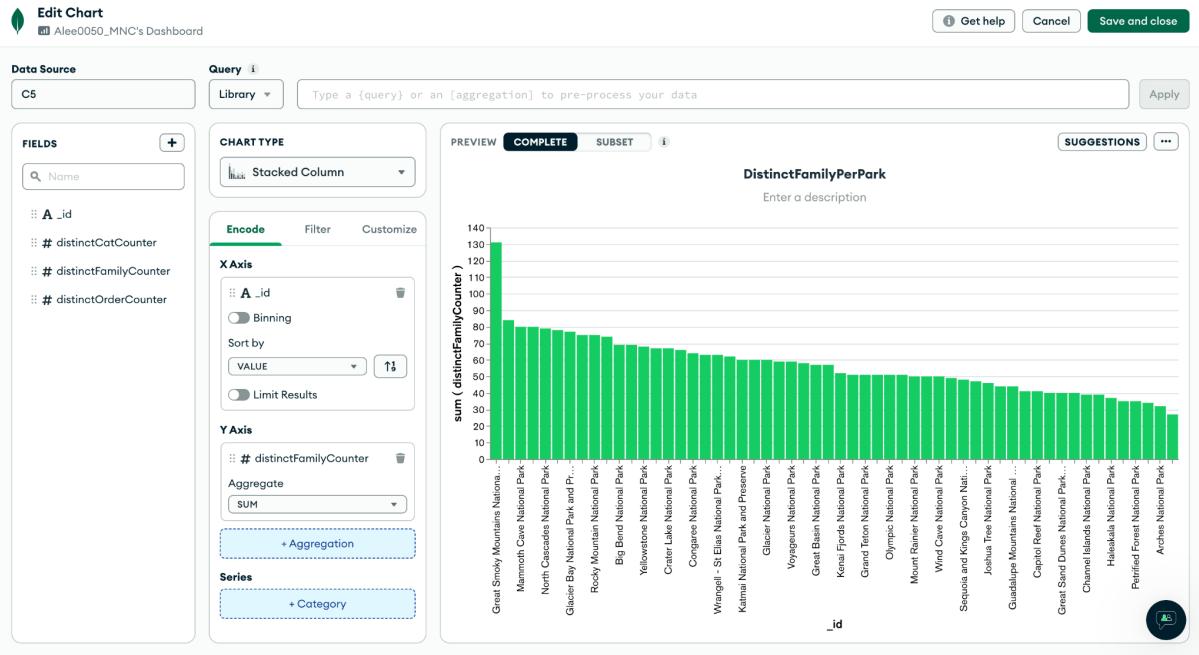
Justification:

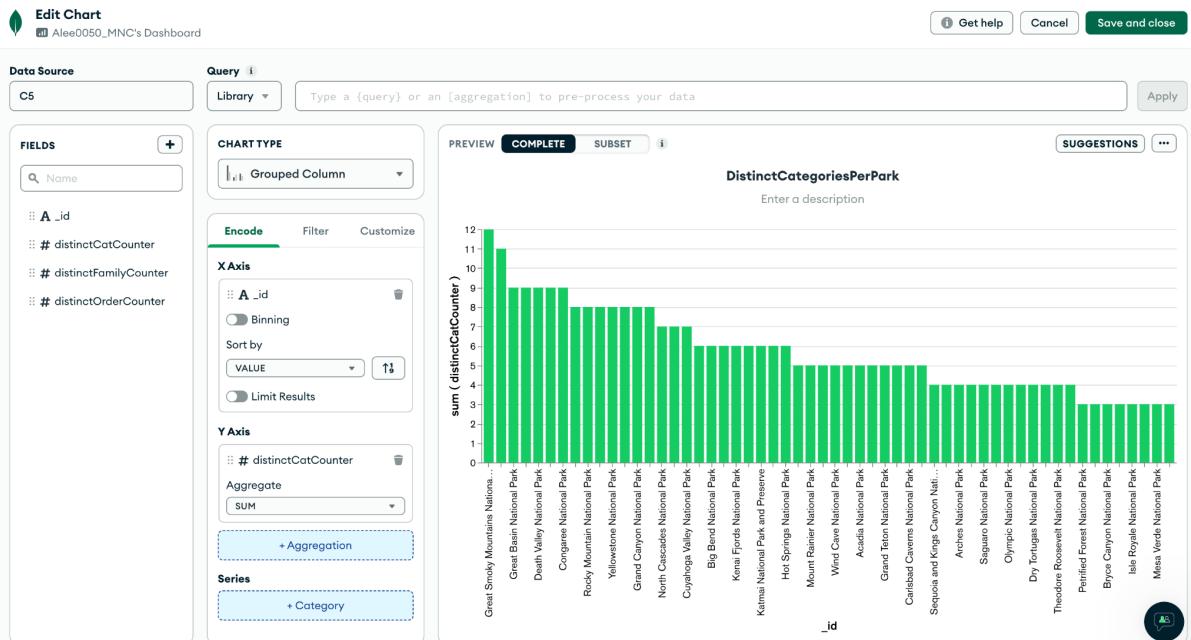
This displays the average bird species in one park, a barchart showcasing the different parks with the amount of birds per park using a barchart. It is easy to make a comparison between the two variables, as their relationship usually revolves around rise over run, furthermore the interpretation is only one, which makes it easy for others to interpret.

C1.3:

C1.4:



C1.5:




Justification:

Since we have three different variables to test the relationship between the park, we want to make 3 different distinct y variables in order to showcase the relationship.