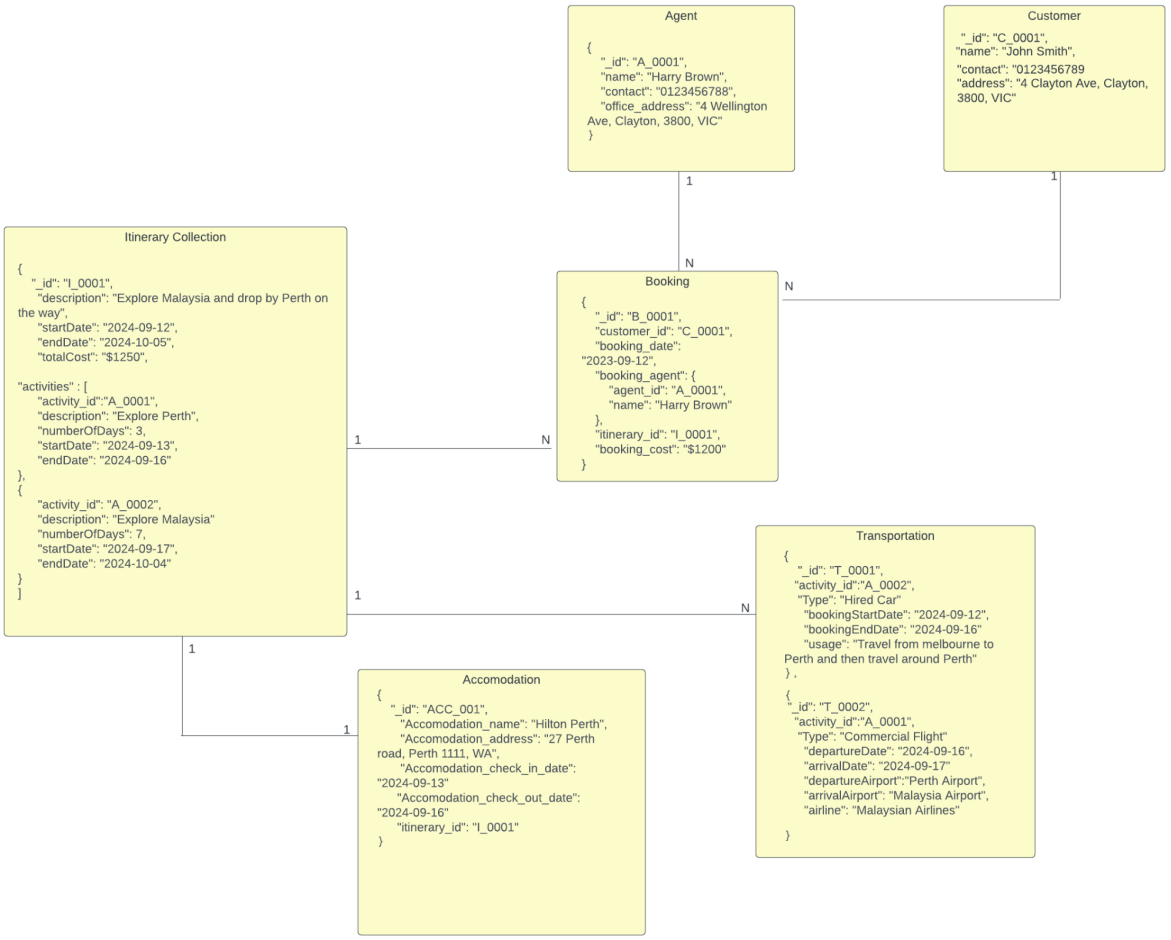
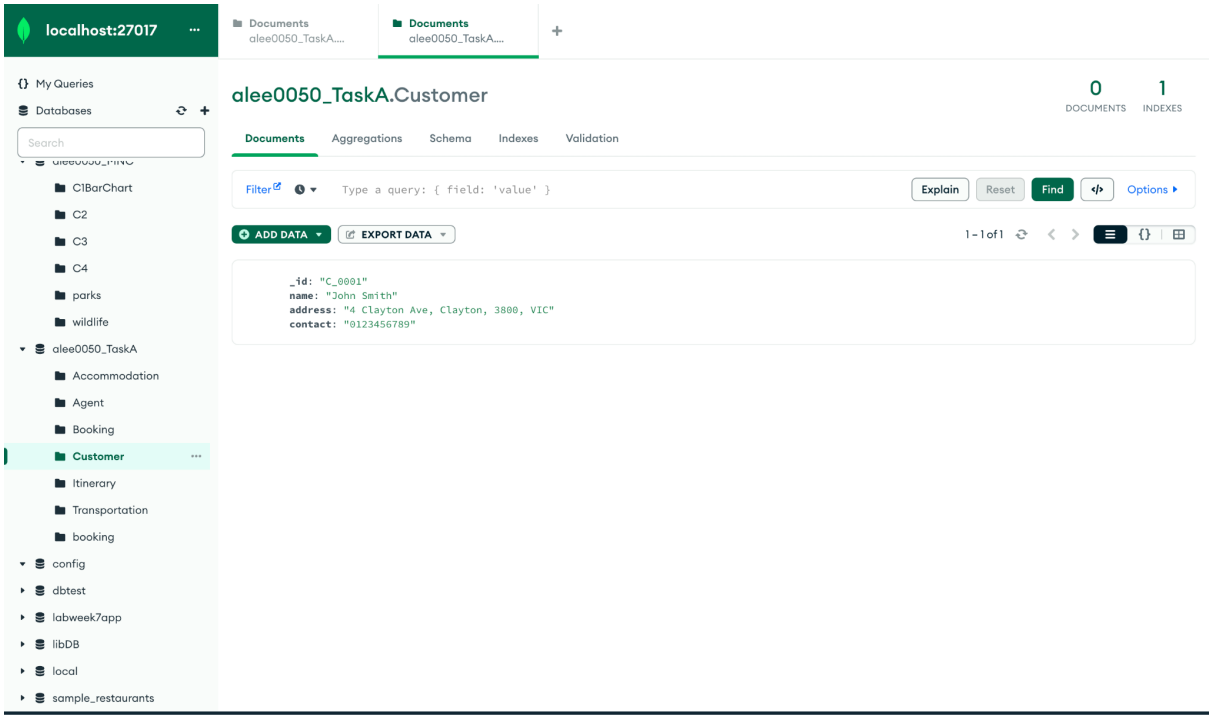
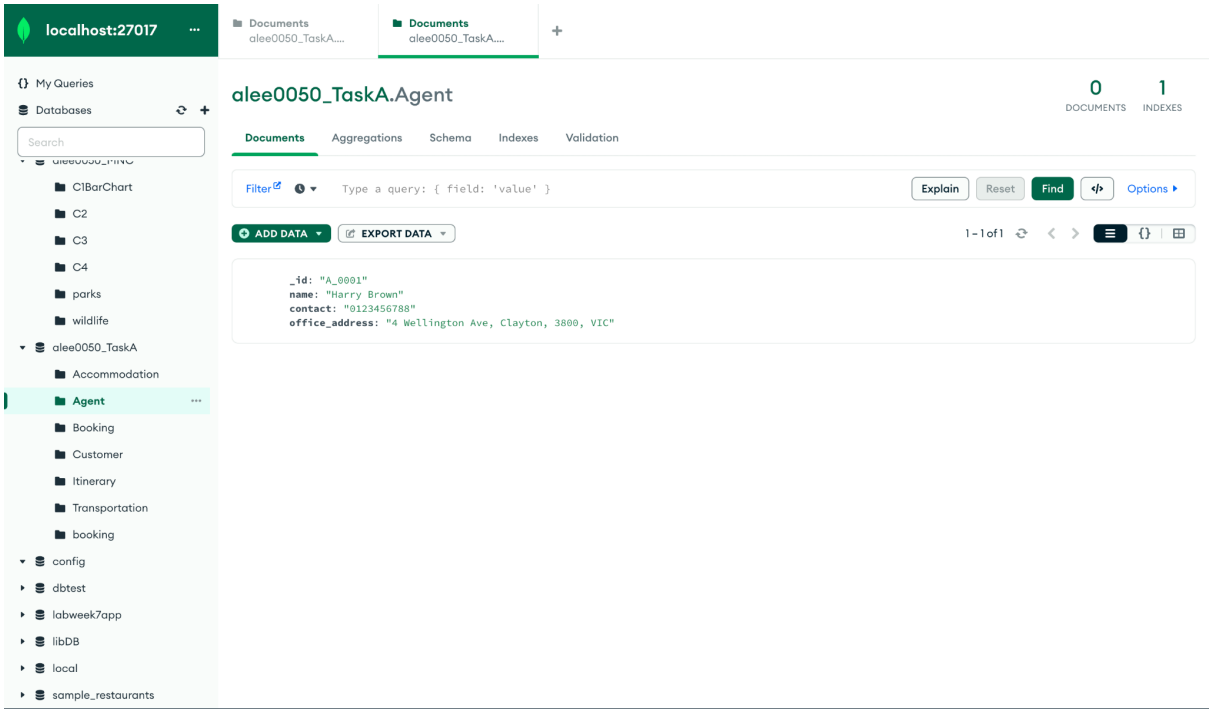
TaskA:
Data model for Monash Travel Agency (MTA)



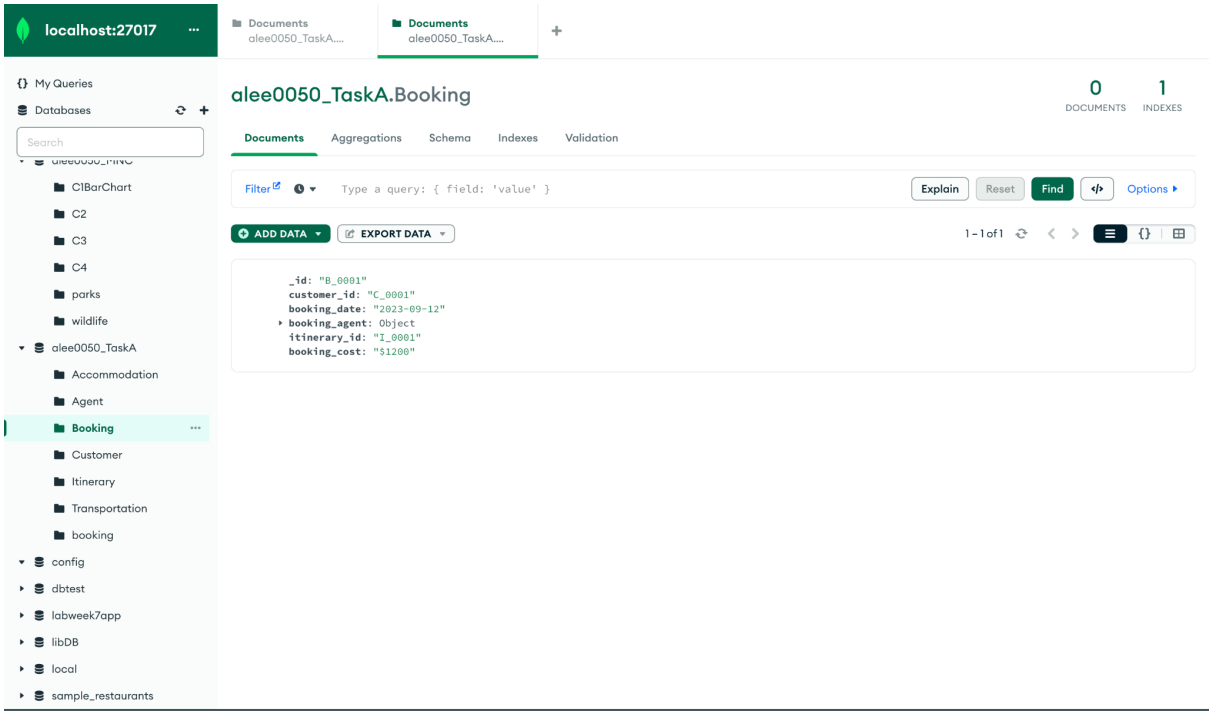
Customer:



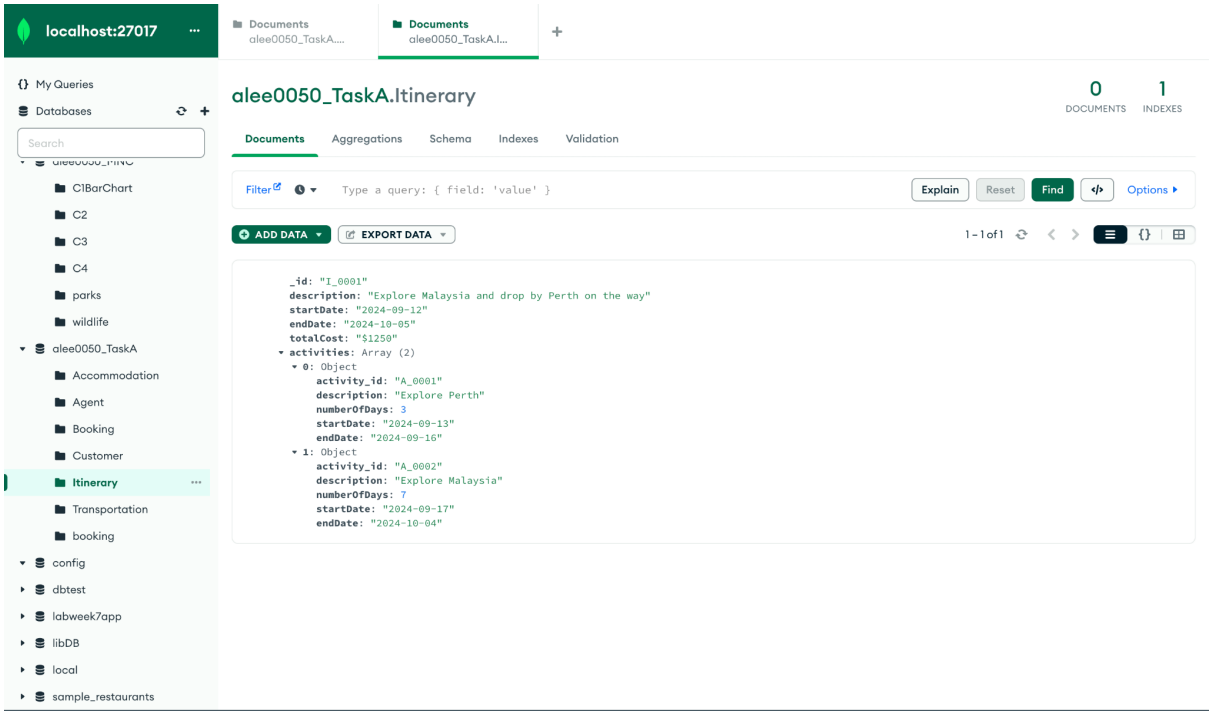
Agent:



Booking:

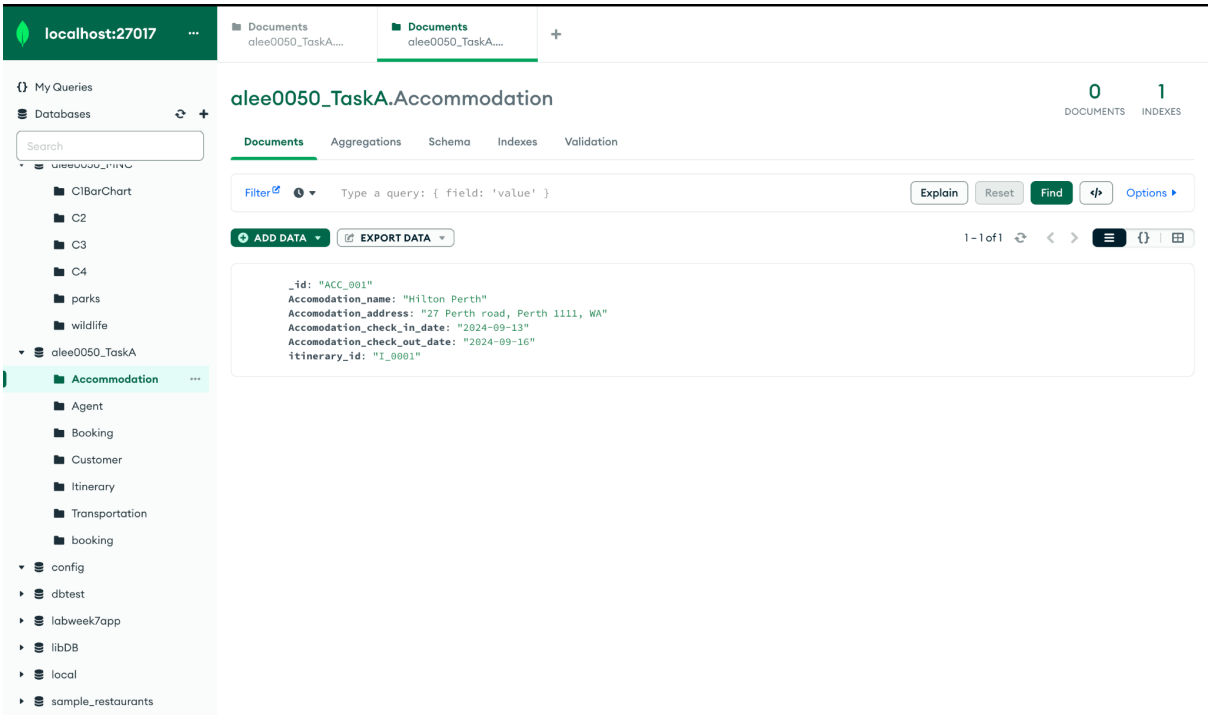


Itinerary:

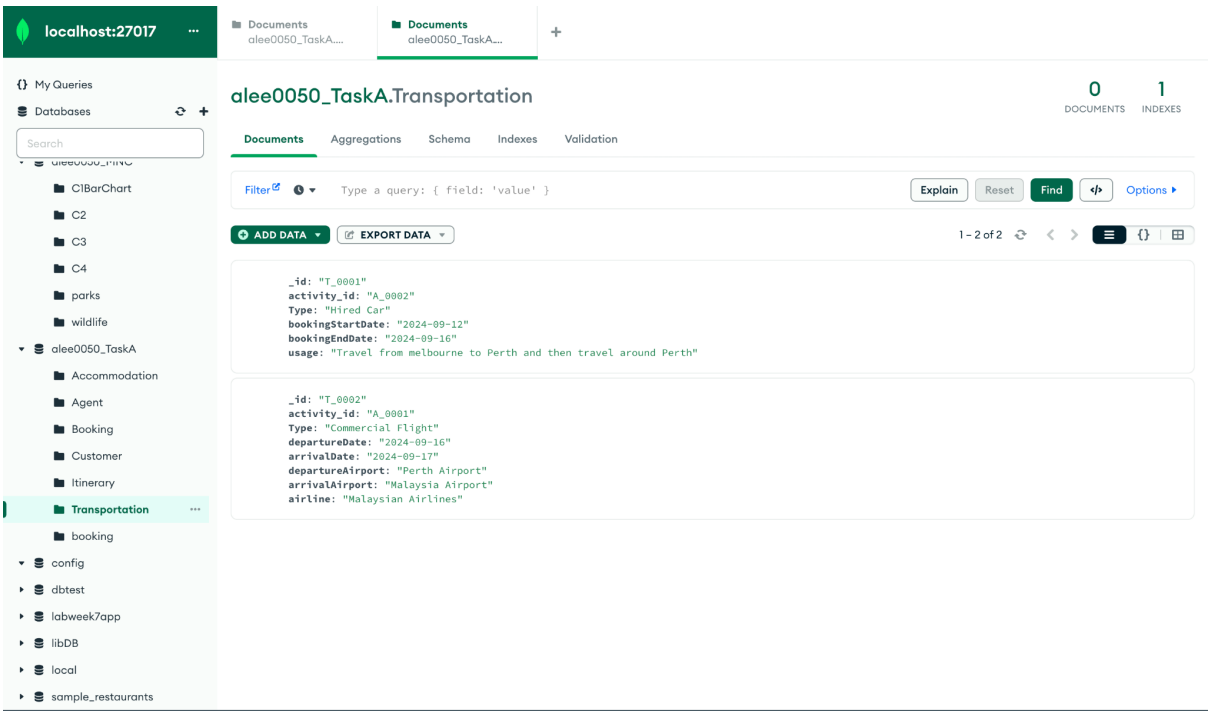


Accommodation:

MONASH University
Information Technology

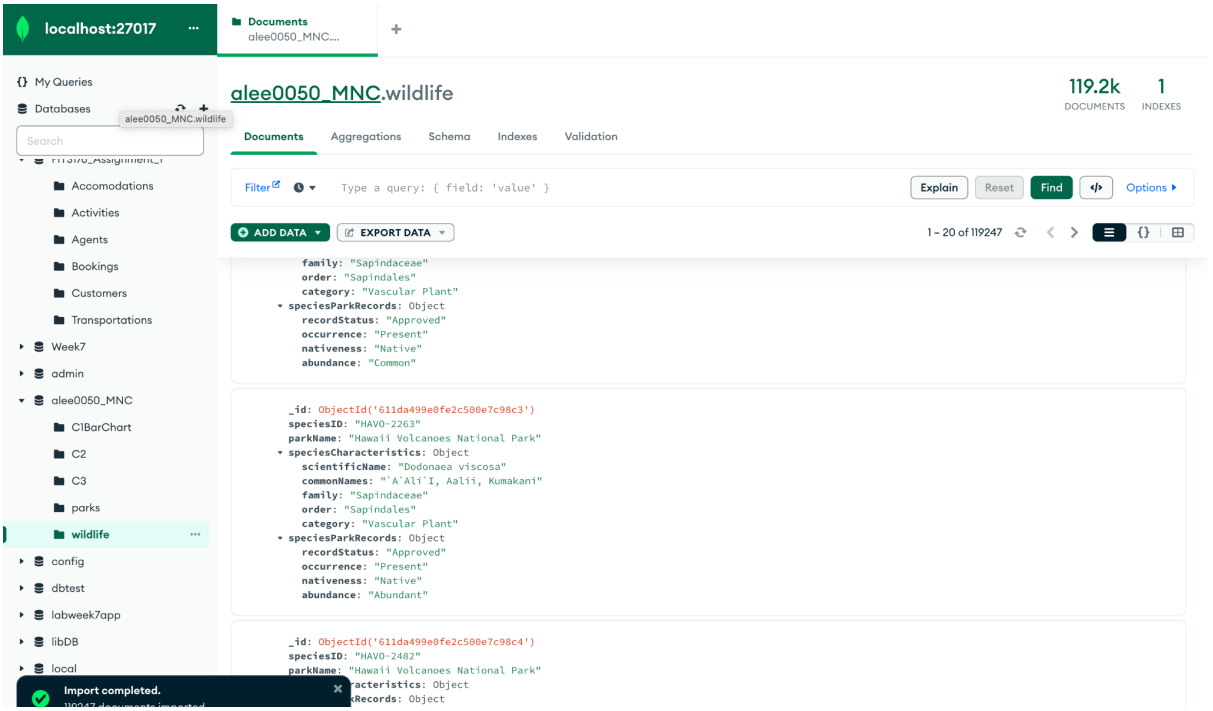


Transportation:



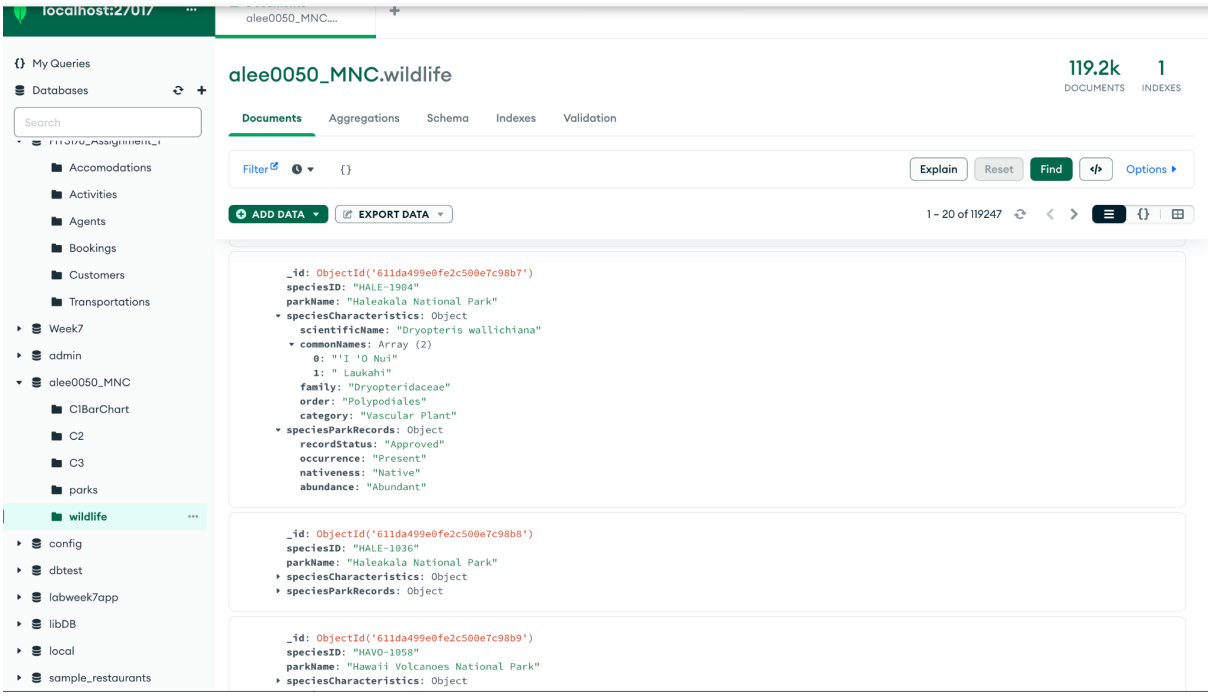
MONASH University
Information Technology

Task B1.1:




```
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", ","]
            }
        }
    }
]
)
```

```
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
}
```



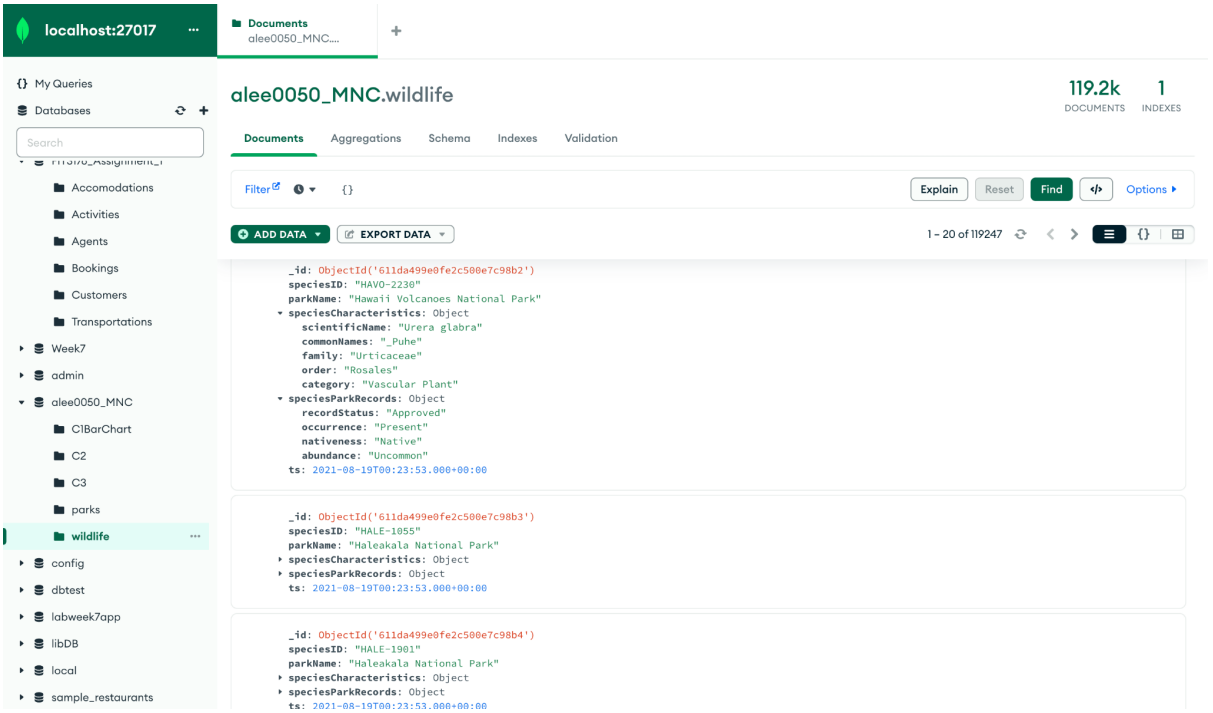
Task B1.2:

```
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

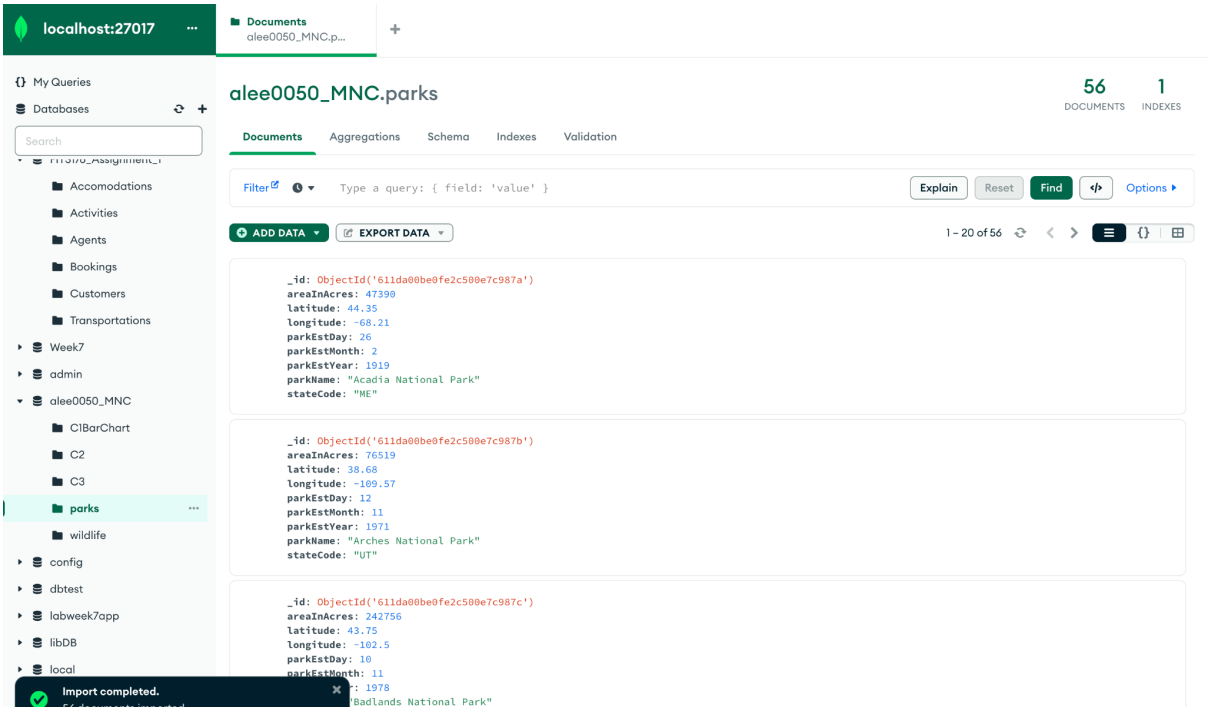
MONASH University
Information Technology

```
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
Timestamp: Thu Aug 19 2021 10:24:48 GMT+1000 (Australian Eastern Standard Time)
```



Task B1.3:

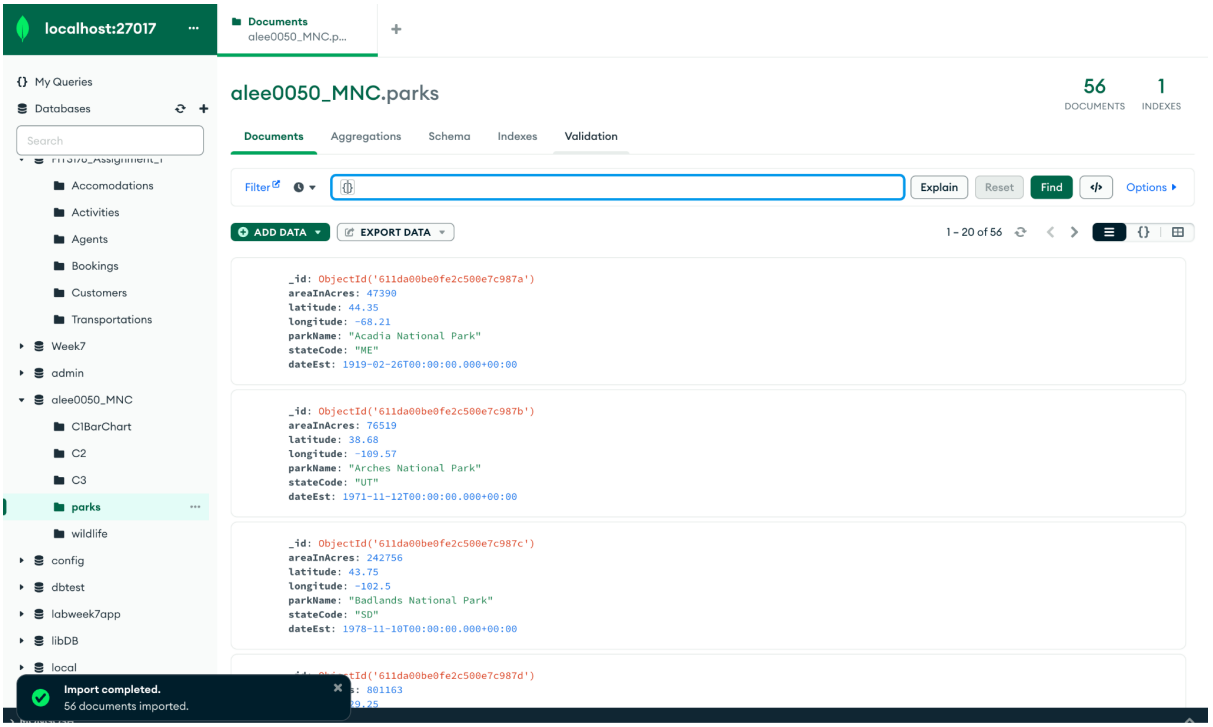
MONASH University
Information Technology



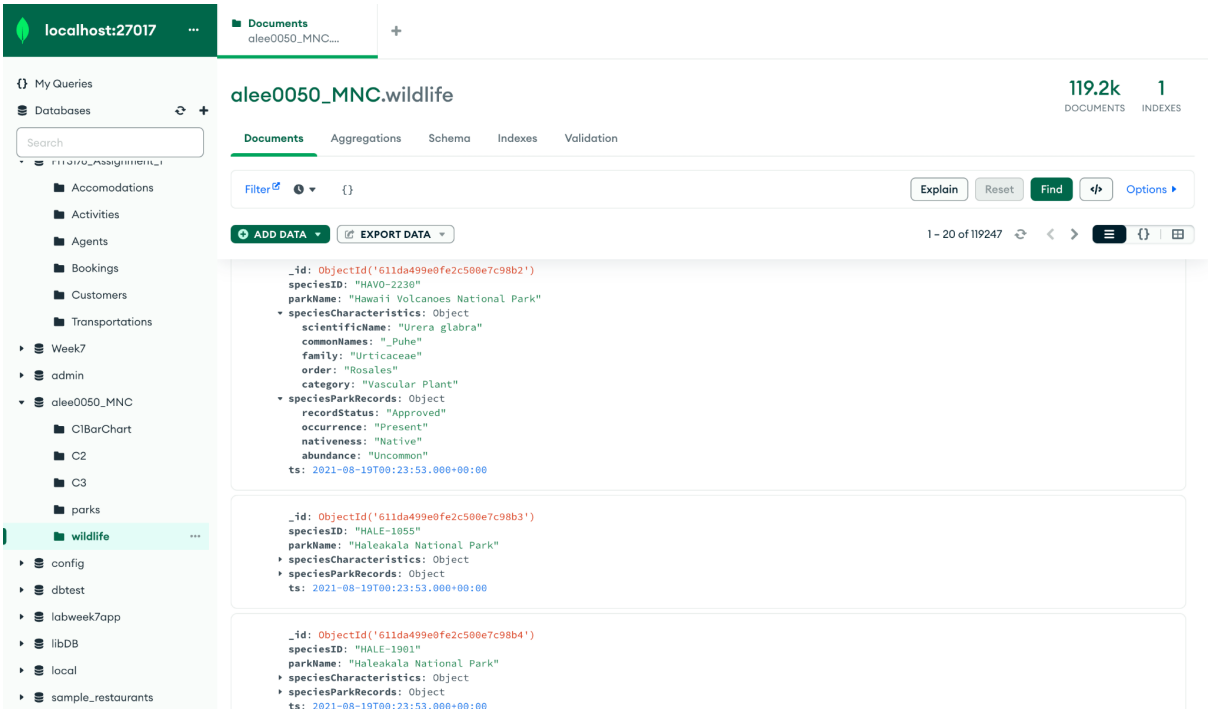

```
db.parks.updateMany(
    {}, // Match all documents in the collection
    [
        {$set: { //using set and unset to do main update operations
                dateEst: { //new field dateEst
                    $dateFromString: { //dateFromString will extract "YYYY-MM-DD"
                        dateString: { //parameter required for dateFromString
                            $concat: [ //concat
                                {$toString: "$parkEstYear"}, //year first
                                "-",
                                {$toString: "$parkEstMonth"}, //month
                                "-",
                                {$toString: "$parkEstDay"} //day
                            ]
                        }
                    }
                }
            },
        {$unset: ["parkEstYear", "parkEstMonth", "parkEstDay"] //remove all 3 fields
        }
    ]
);
```

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 56,
  modifiedCount: 56,
  upsertedCount: 0
}
alee0050_MNC> db.parks.find()
[
```

MONASH University
Information Technology

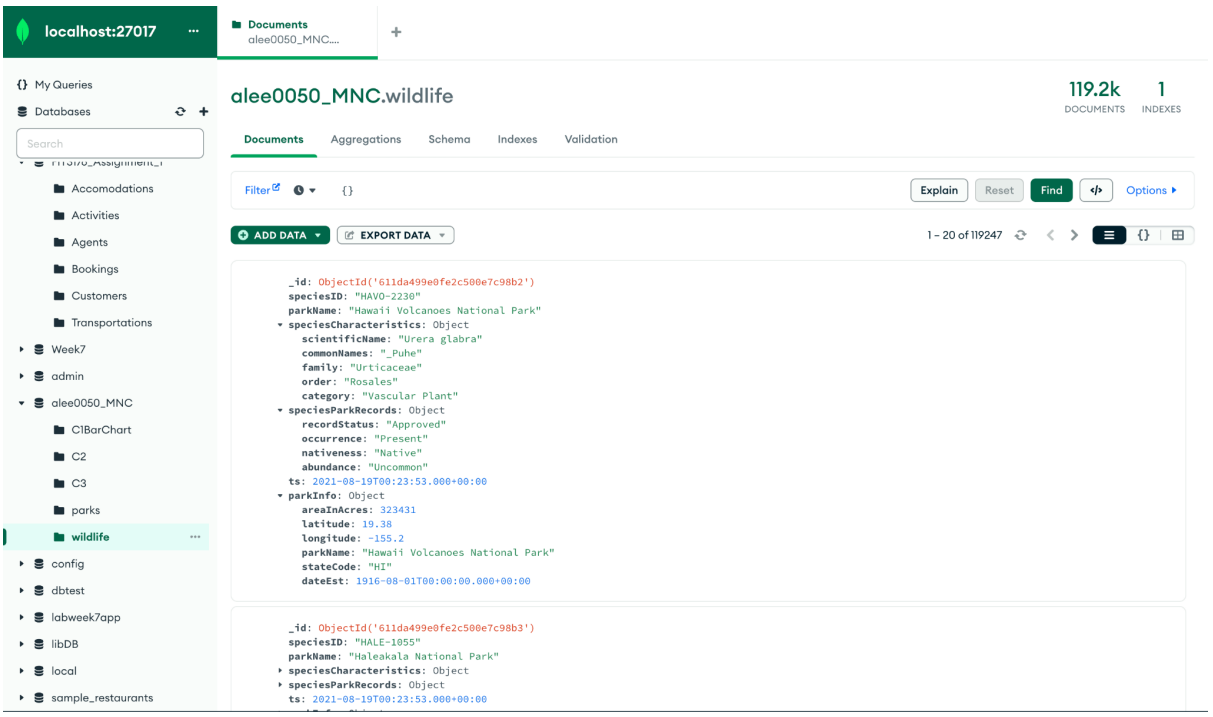


Task B1.4:



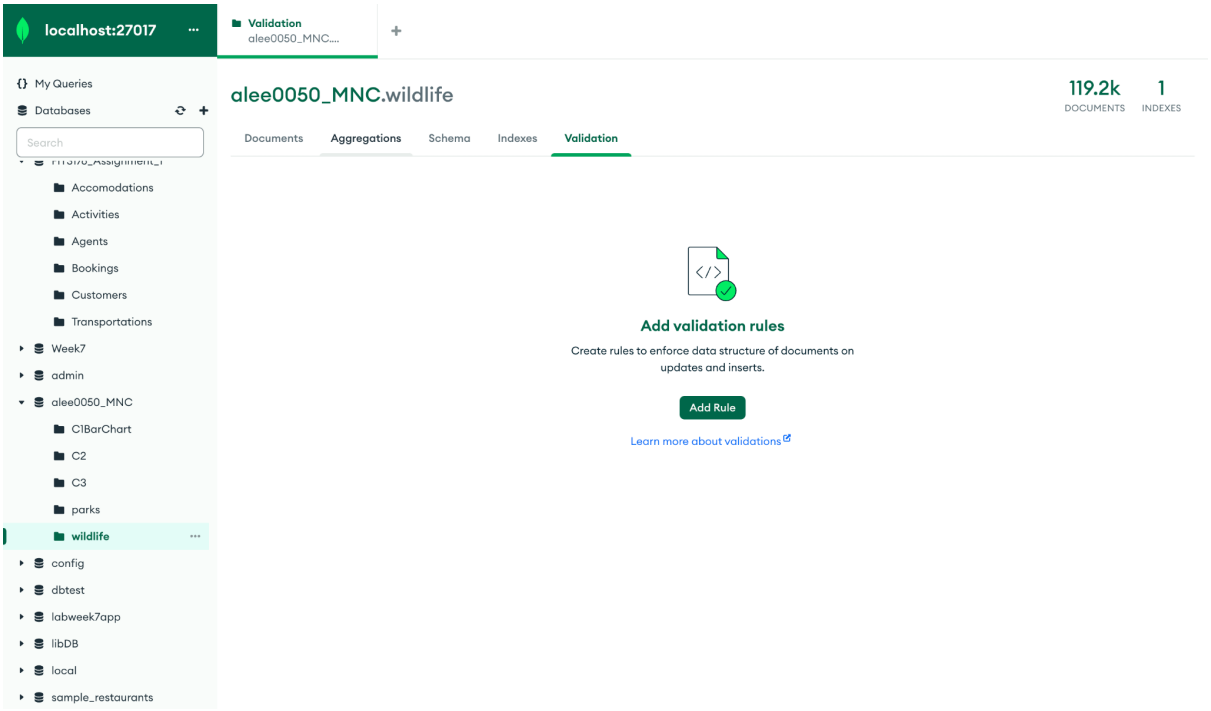
MONASH University
Information Technology

```
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 desconstruct an array field to output docuemnt for each element
    },
    {

    $addFields: { //using add fields to add new fields to our collection for wildlife
        "areaInAcres": "$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.
    }

]);
```



Task B1.5:

MONASH University
Information Technology



MONASH University
Information Technology

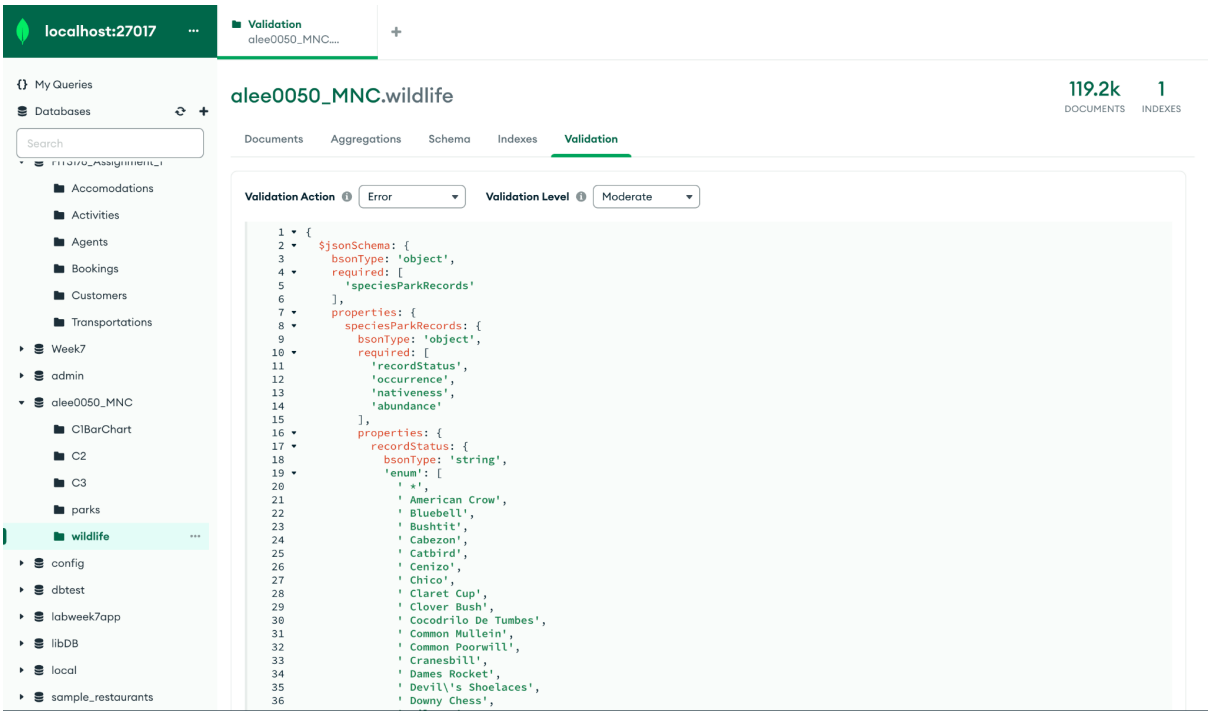
```
db.runCommand({ //it uses the current DB to run whatever logic that is used below
    collMod: "wildlife",  //collMod is used to validate
    validator: {
        $jsonSchema: {
            bsonType: "object",
            required: ["speciesParkRecords"],
            properties: {
                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',
                            ' Cranesbill',
                            ' 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"
                        },
                        occurence: {
                            bsonType: "string",
                            enum: ['Approved',
                            'In Review',
                            'Not Confirmed',
                            'Not Present',
                            'Not Present (False Report)',
                            'Not Present (Historical Report)',
                            'Present'],
                            description: "occurence not valid"
                        },
                        nativeness:{
                            bsonType: "string",
```

MONASH University
Information Technology

```
                    nativeness:{
                        bsonType: "string",
                        enum: ['Native',
                        'Not Confirmed',
                        'Not Native',
                        'Present',
                        'Unknown'],
                        description: "nativeness 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 }
```



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

MONASH University
Information Technology

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:

MONASH University
Information Technology

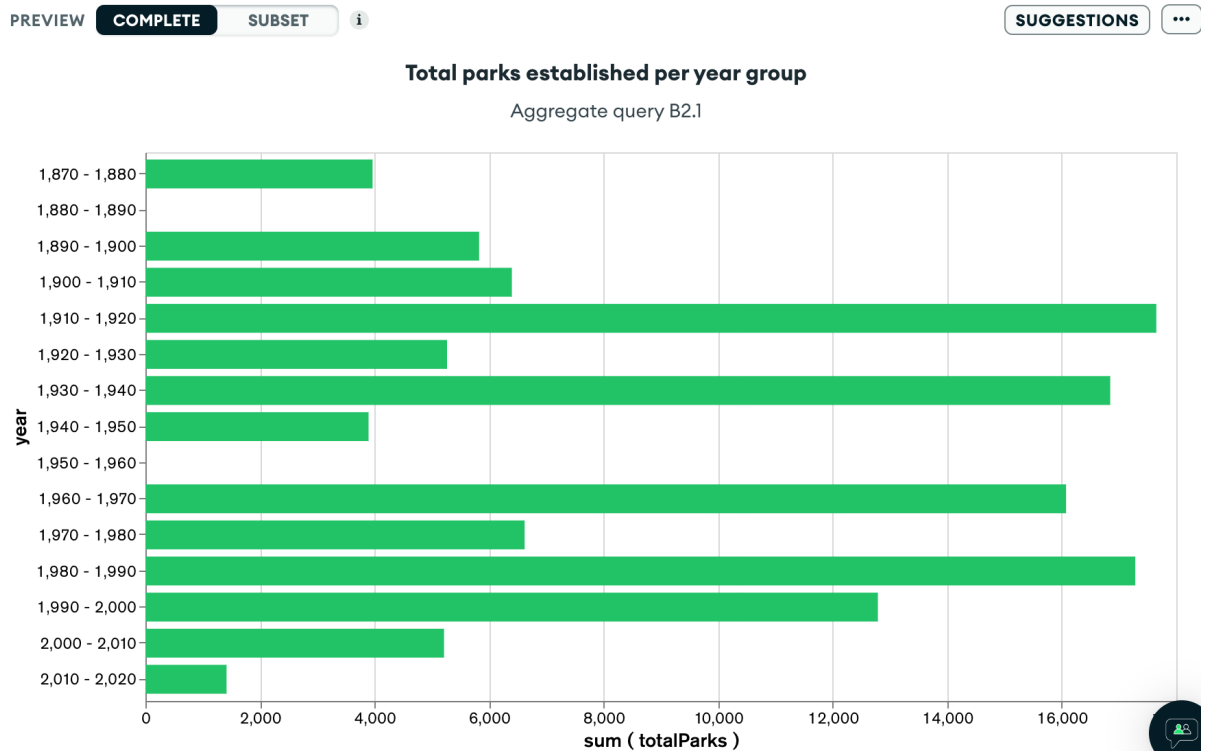
```
//B2.3
db.wildlife.updateMany({},[ //given by the question that we are able to modify our collection

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


//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
        }
    },
    {
      $group: { //group them with parkName, category, order and family
        _id: {
          park: "$parkName",
          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();
```

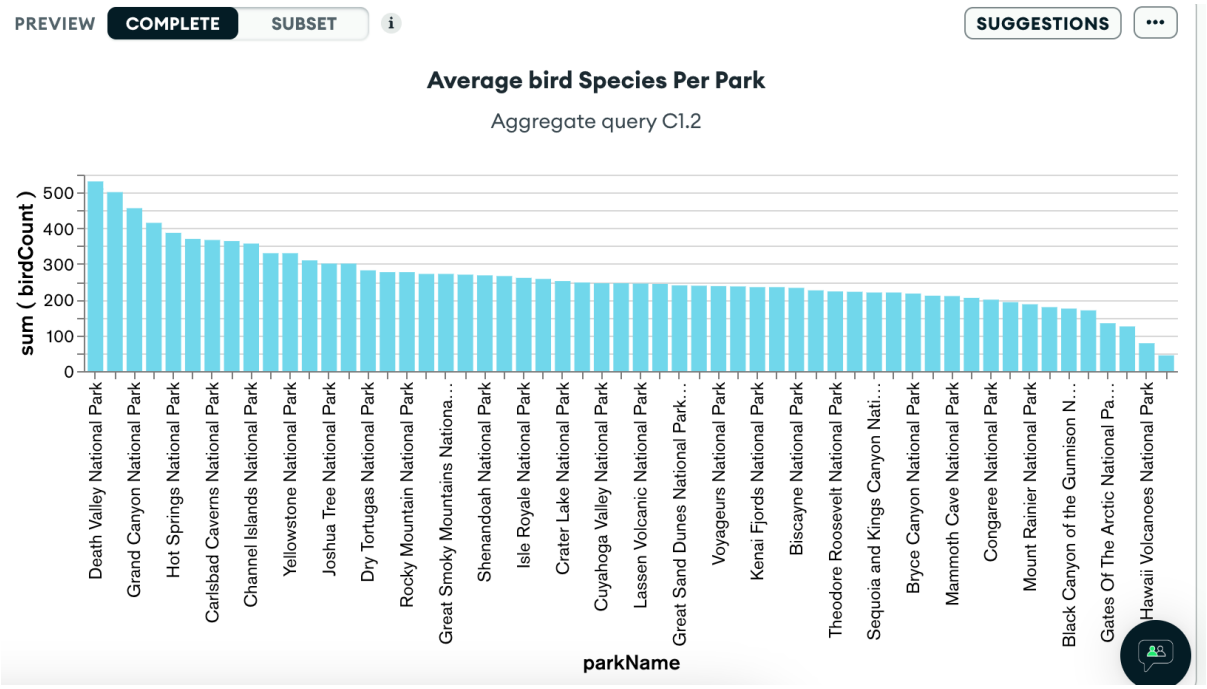
C.1.1:



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:

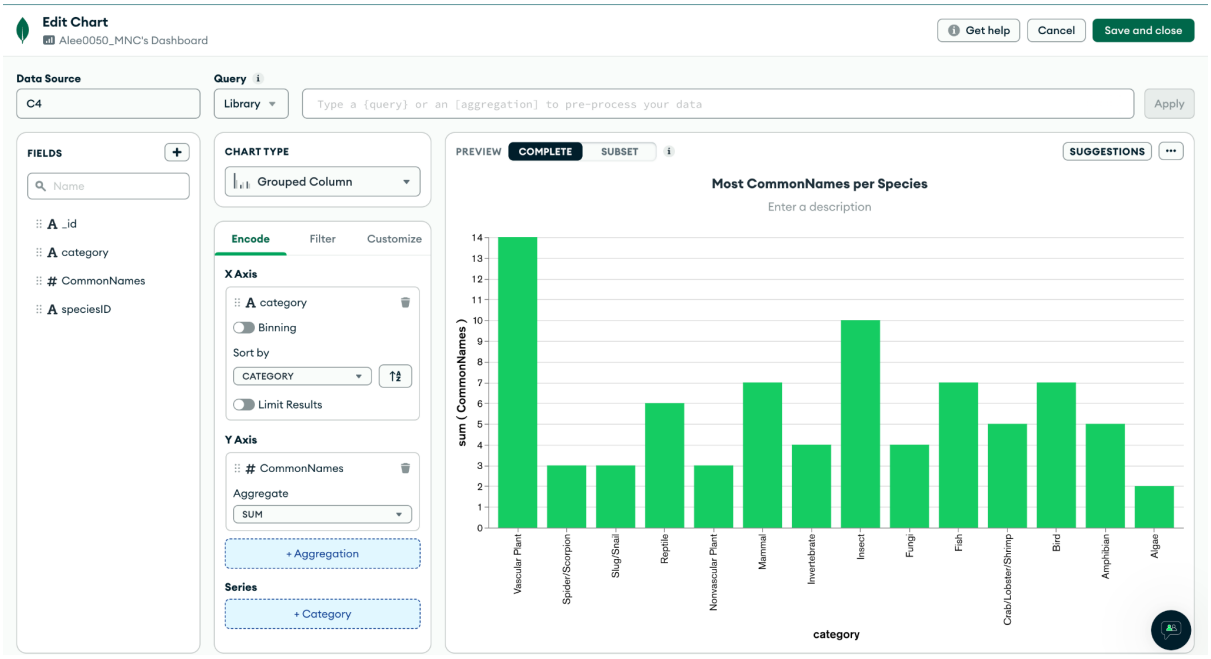


Justification:

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.

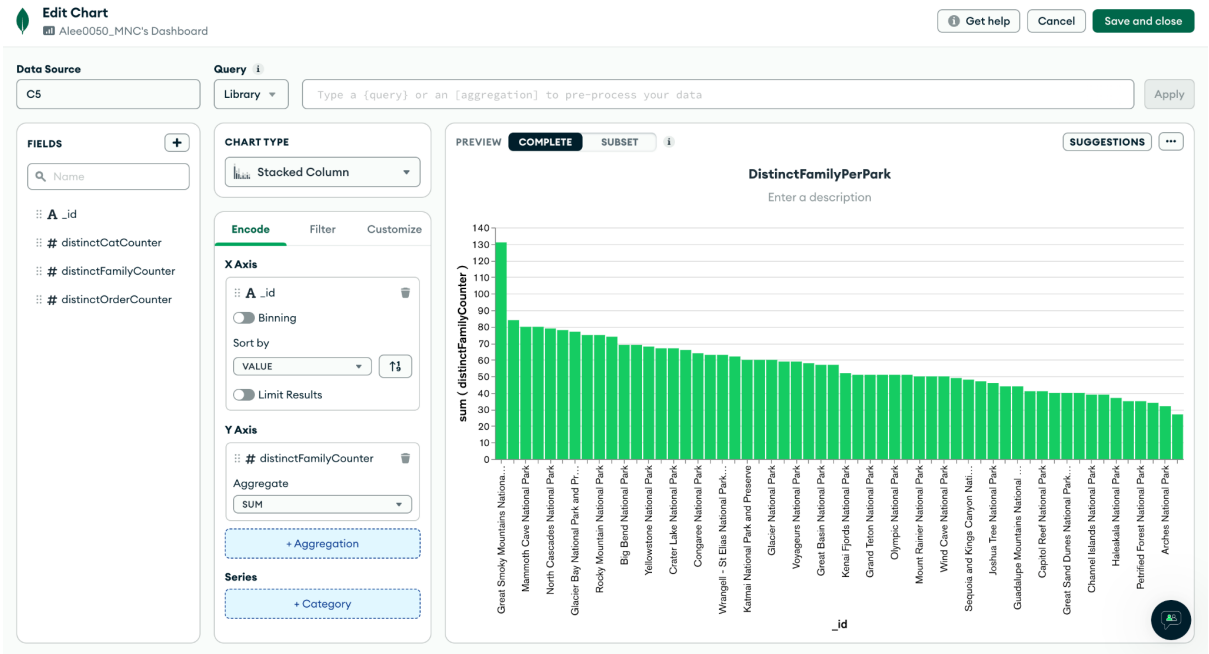
C1.3:

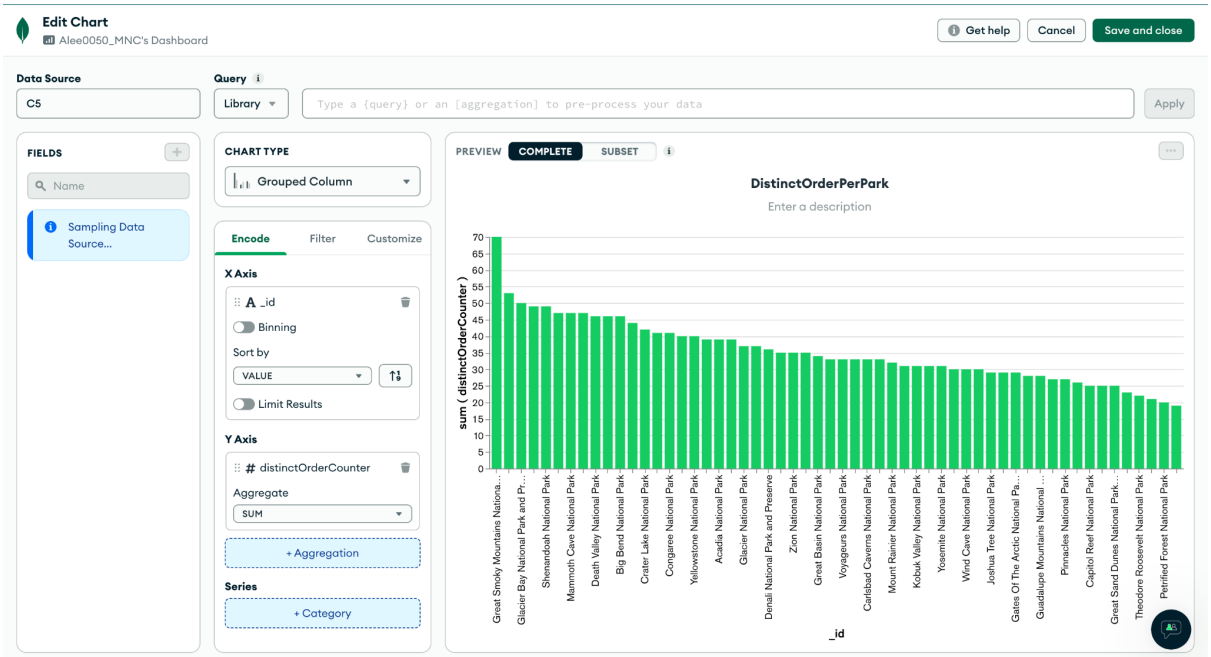
C1.4:



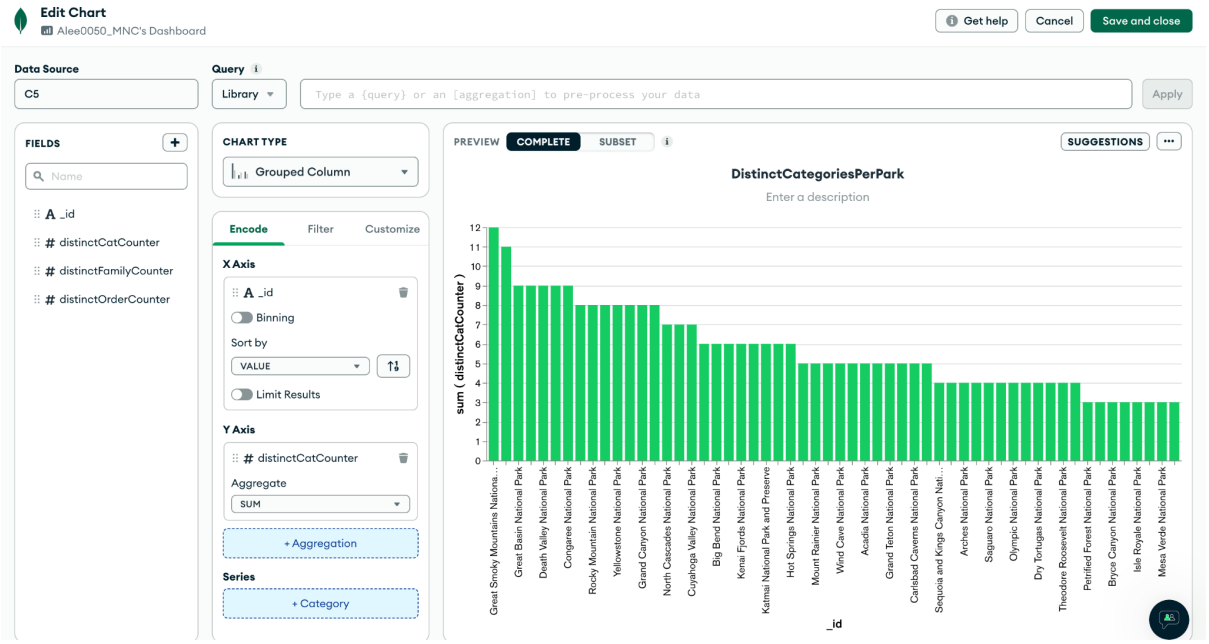
MONASH University
Information Technology

C1.5:





MONASH University
Information Technology



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.