noSQL

Luke Yee

8/13/2020

In this document I will be demonstrating queries from non-relational databases. The database I will use is a sample airbnb database, a collection of documents that represent home listing details and reviews. A sample can be found here: https://docs.atlas.mongodb.com/sample-data/sample-airbnb

A basic query to find listings that are `room_type == "Entire home/apt"` and number of beds >= 3.

```
#m1 is the json object loaded from the database
m1$count(
  '{"room_type" : "Entire home/apt", "beds" : {"$gte" : 3}}'
)
```

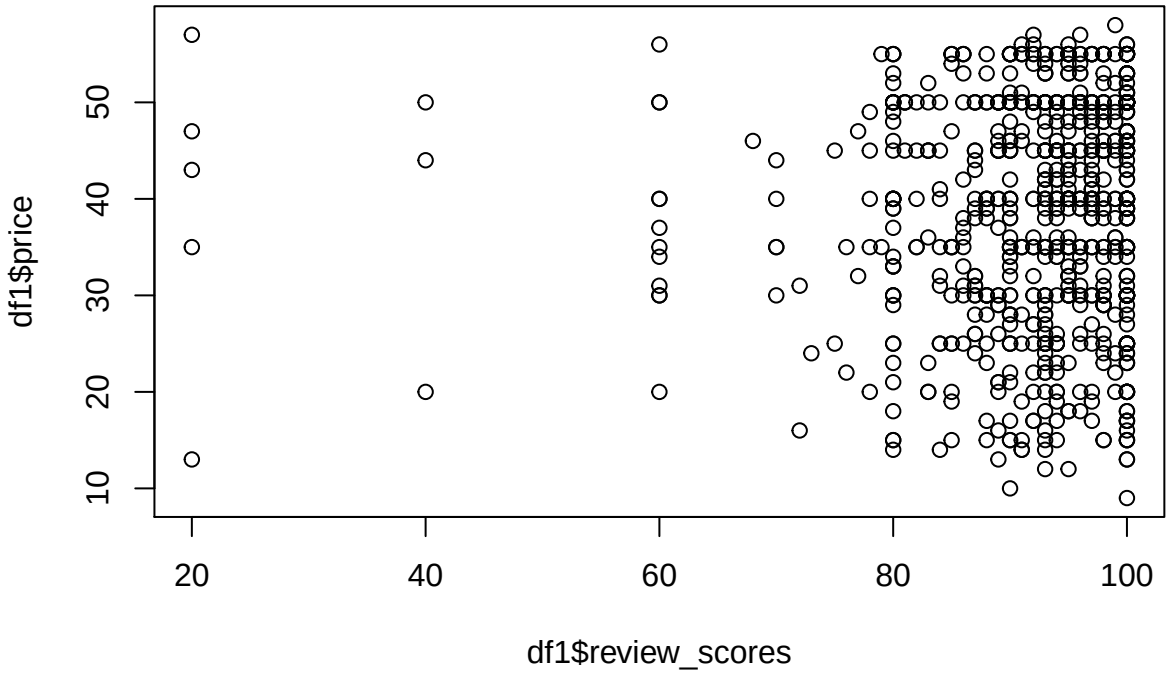
```
## [1] 1288
```

A query of all the experience ratings and prices, and then a plot of `rating` vs `price`

```
m1$find(
  '{}',
  fields = '{"review_scores.review_scores_rating" : true, "price" : true, "name" : true}',
  sort = '{"price" : 1}',
  limit = 10
)
```

```
##         _id                                                name price
## 1  14758068                        Room on spacious appartment     9
## 2  20611485                Cómoda Habitación L'Eixample, Gracia    10
## 3  31305846        Private room with sunny terrace of 200m2. 6     10
## 4  32636126 Near the RAMBLA, the double room at  SEASIDE PORT1     10
## 5  24450419                   Quarto 1 do AL Hostel D.ª Jucunda    12
## 6   8521963                                Habitación privada      12
## 7  19904358                                         Good room      13
## 8  26563602                              Quarto moradia luxo      13
## 9  28493488                                   Shiny Cottage      13
## 10  6583887                              E-House Relógio - 2C     13
##    review_scores_rating
## 1                   100
## 2                    90
## 3                    NA
## 4                    NA
## 5                    93
## 6                    95
## 7                   100
## 8                    20
## 9                   100
## 10                   89
```

```
df1 <- m1$find(
  '{}',
  fields = '{"review_scores.review_scores_rating" : true, "price" : true, "name" : true}',
  sort = '{"price" : 1}',
  limit = 1000
) %>% mutate(review_scores = review_scores$review_scores_rating)
plot(df1$review_scores, df1$price)
```



We can see that price is not the best indicator of a high review score, as both low-price and high-price listings have good reviews

We can also find properties that have certain features. For example, here is a list of properties that have "Washer" and "Kitchen" as amenities

```
m1$find(
  '{"amenities" : {"$in" : ["Washer" , "Kitchen"]}}',
  fields = '{"name" : true}',
  limit = 10
)
```

```
##        _id                                name
## 1  10006546             Ribeira Charming Duplex
## 2  10009999        Horto flat with small garden
## 3   1001265    Ocean View Waikiki Marina w/prkg
## 4  10021707            Private Room in Bushwick
## 5  10030955         Apt Linda Vista Lagoa - Rio
## 6   1003530 New York City - Upper West Side Apt
## 7  10038496        Copacabana Apartment Posto 6
```

```
## 8  10047964      Charming Flat in Downtown Moda
## 9  10051164  Catete's Colonial Big Hause Room B
## 10 10057447      Modern Spacious 1 Bedroom Loft
```

We can also filter listings by certain characteristics and find summary statistics. For example, this query finds all properties with over 100 reviews, then finds the average price for each type of property

```
m1$aggregate('[
  {"$match": {"number_of_reviews": {"$gte": 101}}},
  {"$group": { "_id": "$property_type", "price": { "$avg": "$price" }}},
  {"$sort": { "price": -1}}
]')
```

```
##                   _id     price
## 1              Hostel 447.00000
## 2      Boutique hotel 322.66667
## 3         Condominium 233.24242
## 4           Treehouse 185.00000
## 5           Apartment 175.48754
## 6          Guesthouse 157.66667
## 7               House 156.83636
## 8             Cottage 156.66667
## 9  Serviced apartment 143.00000
## 10           Bungalow 137.50000
## 11  Bed and breakfast 120.66667
## 12          Townhouse 111.38462
## 13         Aparthotel 109.00000
## 14              Cabin  97.00000
## 15        Guest suite  95.46154
## 16              Hotel  87.00000
## 17               Loft  83.41176
## 18              Other  65.00000
```

The next database contains sales data. Each document in the `sales` collection represents a single sale from a store run by the supply company, with relevant information such as the items purchased and customer information. A sample document can be found here: https://docs.atlas.mongodb.com/sample-data/sample-supplies/

Note: to handle items, an `$unwind` must be used in the `aggregate` stage.

The following query finds the number of items per transaction id

```
#m2 is the json object loaded from the database
m2$aggregate('[
{"$unwind" : "$items"},
{"$group": {
    "_id": "$_id",
    "items": {"$sum" : "$items.quantity"}
    }
  },

  {"$limit": 20}
]')
```

```
##                        _id items
## 1  5bd761deae323e45a93ce2e5     9
## 2  5bd761deae323e45a93ce2e4    19
## 3  5bd761deae323e45a93ce2e3    25
```

```
## 4  5bd761deae323e45a93ce2e2    12
## 5  5bd761deae323e45a93ce2e1    37
## 6  5bd761deae323e45a93ce2e0    37
## 7  5bd761deae323e45a93ce2df    21
## 8  5bd761deae323e45a93ce2de     3
## 9  5bd761deae323e45a93ce1ff     4
## 10 5bd761deae323e45a93ce1fe    25
## 11 5bd761deae323e45a93ce1fd    30
## 12 5bd761deae323e45a93ce1fc    20
## 13 5bd761deae323e45a93ce1fb    13
## 14 5bd761deae323e45a93ce1fa     4
## 15 5bd761deae323e45a93ce1f9    26
## 16 5bd761deae323e45a93ce1f8    30
## 17 5bd761deae323e45a93ce1f7    20
## 18 5bd761deae323e45a93ce1f6    12
## 19 5bd761deae323e45a93ce1f5    30
## 20 5bd761deae323e45a93ce1f4    45
```

The following query finds the amount of money spent in each transaction, multiplying corresponding item price and item quantity, then summing the total

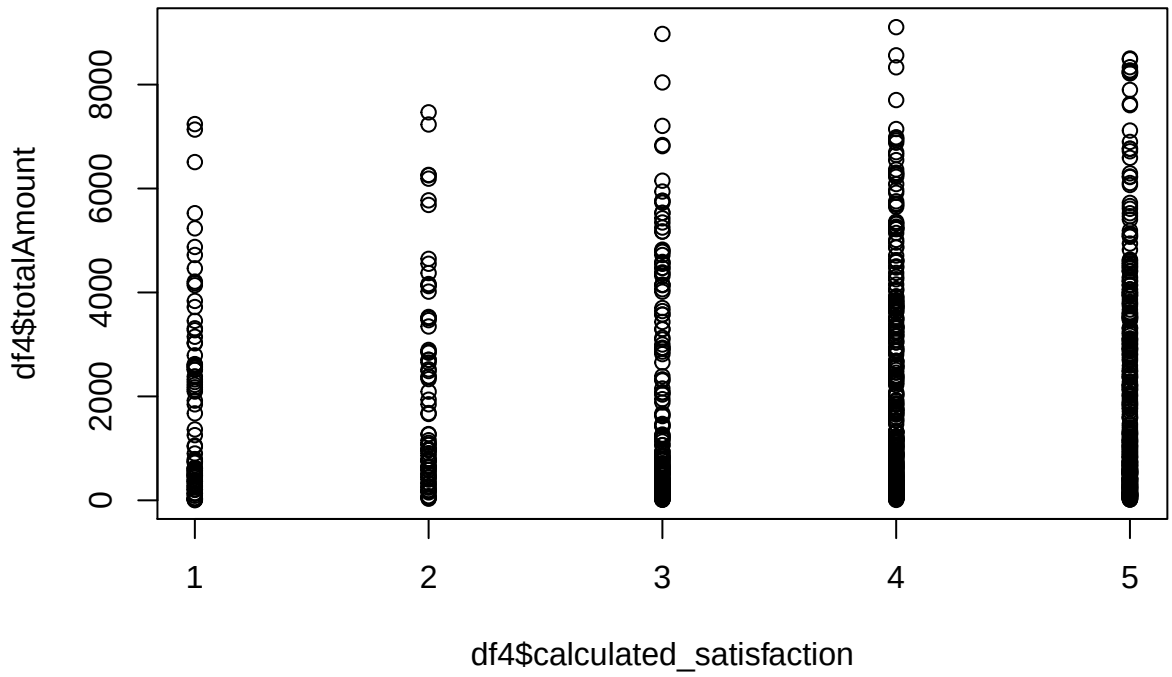
```
m2$aggregate('[
{"$unwind" : "$items"},
{"$group": {
    "_id": "$_id",
    "totalAmount" : { "$sum": { "$multiply": [ "$items.price", "$items.quantity" ] } }
    }
  },
  {"$limit": 20}
]')
```

```
##                         _id totalAmount
## 1  5bd761deae323e45a93ce2e5      317.95
## 2  5bd761deae323e45a93ce2e4     5736.57
## 3  5bd761deae323e45a93ce2e3     5904.47
## 4  5bd761deae323e45a93ce2e2      394.87
## 5  5bd761deae323e45a93ce2e1      791.95
## 6  5bd761deae323e45a93ce2e0     2278.50
## 7  5bd761deae323e45a93ce2df      539.93
## 8  5bd761deae323e45a93ce2de       29.37
## 9  5bd761deae323e45a93ce1ff       44.04
## 10 5bd761deae323e45a93ce1fe      582.94
## 11 5bd761deae323e45a93ce1fd     2135.34
## 12 5bd761deae323e45a93ce1fc      853.60
## 13 5bd761deae323e45a93ce1fb     2457.70
## 14 5bd761deae323e45a93ce1fa       42.64
## 15 5bd761deae323e45a93ce1f9      671.17
## 16 5bd761deae323e45a93ce1f8     3447.21
## 17 5bd761deae323e45a93ce1f7     8199.19
## 18 5bd761deae323e45a93ce1f6     3729.98
## 19 5bd761deae323e45a93ce1f5     1699.17
## 20 5bd761deae323e45a93ce1f4     5338.91
```

We can also plot the customer satisfaction against the price:

```
df2 <- m2$aggregate('[
{"$unwind" : "$items"},
{"$group": {
    "_id": "$_id",
    "totalAmount" : { "$sum": { "$multiply": [ "$items.price", "$items.quantity" ] } }
    }
  },
  {"$limit": 2000}
]')

df3 <- m2$find('{}', fields = '{"customer.satisfaction":1}', limit = 2000)
df3$calculated_satisfaction <- df3$customer$satisfaction
df4 <- merge(df2, df3, by="_id")
plot(df4$calculated_satisfaction, df4$totalAmount)
```



We see the lower priced items do not necessarily mean lower customer satisfaction scores.

Another useful thing we can find is the total sum of transactions by each store to see which stores have the most money flowing through

```
m2$aggregate('[
{"$group": {
    "_id": "$storeLocation",
    "transactions": {"$sum" : 1}
    }
}
]')
```

```
##         _id transactions
## 1 San Diego          346
## 2   Seattle         1134
## 3    Denver         1549
## 4    London          794
## 5  New York          501
## 6    Austin          676
```

We can see that the Denver location has the highest sum of transactions. Finally, we can find details regarding specific items. For example, the total amount of notepads sold across all locations:

```
df5 <- m2$aggregate('[
  {"$unwind": "$items"},
  {"$group": {
    "_id": "$items.name",
    "item_amount": {"$sum" : "$items.quantity"}
    }
  },
  {"$match": {"_id": "notepad"}}
]')
df5
```

```
##       _id item_amount
## 1 notepad       20727
```