




7 ΙΑΝΟΥΑΡΙΟΥ 2025

BIG DATA MANAGEMENT
1Η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
UNIVERSITY OF PIRAEUS

MICHALIS KOVAIOS

MPKED2322

mixalis.koveos@gmail.com



Πίνακας περιεχομένων

Ερώτημα 1: Εισαγωγή Δεδομένων	2
Ερώτημα 2: Ανάκτηση Δεδομένων	6
Υποερώτημα 2.1	6
Υποερώτημα 2.2	9
Υποερώτημα 2.3	11
Υποερώτημα 2.4	14
Υποερώτημα 2.5	15
Ερώτημα 3: Optimization.....	17
Υποερώτημα 3.1	24
Υποερώτημα 3.2	27
Υποερώτημα 3.3	29
Υποερώτημα 3.4	31
Υποερώτημα 3.5	33

Ερώτημα 1: Εισαγωγή Δεδομένων

Η διαδικασία αφορά στην εισαγωγή του αρχείου δεδομένων `tripadvisor_european_restaurants.json` στη βάση δεδομένων MongoDB. Το αρχείο περιλαμβάνει πληροφορίες για 1.083.397 ευρωπαϊκά εστιατόρια και περιέχει πεδία όπως τοποθεσία, τύποι κουζίνας, ώρες λειτουργίας, μέση βαθμολογία, αριθμός κριτικών, και άλλα.

Βήμα 1: Αντιγραφή Αρχείου στο Container

Για να μεταφέρουμε το αρχείο JSON μέσα στο Docker container, χρησιμοποιήθηκε η εντολή `docker cp` από το PowerShell.

```
docker cp
```

```
"E:\ModifiedDownloads\tripadvisor_european_restaurants\tripadvisor_european_restaurants.json" MONGO_CONTAINER:/tmp/tripadvisor_european_restaurants.json
```

```
PS C:\Users\Feuer_Frei> docker cp "E:\ModifiedDownloads\tripadvisor_european_restaurants\tripadvisor_european_restaurants.json" MONGO_CONTAINER:/tmp/tripadvisor_european_restaurants.json
Successfully copied 1.39GB to MONGO_CONTAINER:/tmp/tripadvisor_european_restaurants.json
```

Το αρχείο αποθηκεύτηκε στη διαδρομή `/tmp` μέσα στο container με όνομα `MONGO_CONTAINER`.

Βήμα 2: Πρόσβαση στο Container

Για να προχωρήσουμε στην εισαγωγή των δεδομένων στη MongoDB, εισήλαμε στο περιβάλλον του container χρησιμοποιώντας την ακόλουθη εντολή:

```
docker exec -it MONGO_CONTAINER bash
```

```
root@ea3ce78beb3f:/#
```

Βήμα 3: Εισαγωγή Δεδομένων στη MongoDB

Μέσα στο container, χρησιμοποιήθηκε η εντολή mongoimport για να εισαχθεί το αρχείο JSON στη βάση δεδομένων MongoDB.

```
mongoimport --db TripAdvisor --collection restaurants --file  
/tmp/tripadvisor_european_restaurants.json --jsonArray
```

- **Βάση δεδομένων:** TripAdvisor
- **Συλλογή:** restaurants
- **Διαδρομή αρχείου:** /tmp/tripadvisor_european_restaurants.json
- **Μορφή αρχείου:** JSON Array

```
.json --jsonArray  
2025-01-07T15:49:37.823+0000 connected to: mongodb://localhost/  
2025-01-07T15:49:40.824+0000 [#.....] TripAdvisor.restaurants59.2MB/1.29GB (4.5%)  
2025-01-07T15:49:43.824+0000 [##.....] TripAdvisor.restaurants135MB/1.29GB (10.2%)  
2025-01-07T15:49:46.824+0000 [###.....] TripAdvisor.restaurants209MB/1.29GB (15.8%)  
2025-01-07T15:49:49.824+0000 [####.....] TripAdvisor.restaurants284MB/1.29GB (21.5%)  
2025-01-07T15:49:52.824+0000 [#####.....] TripAdvisor.restaurants356MB/1.29GB (26.9%)  
2025-01-07T15:49:55.824+0000 [#####.....] TripAdvisor.restaurants431MB/1.29GB (32.6%)  
2025-01-07T15:49:58.824+0000 [#####.....] TripAdvisor.restaurants504MB/1.29GB (38.1%)  
2025-01-07T15:50:01.824+0000 [#####.....] TripAdvisor.restaurants580MB/1.29GB (43.9%)  
2025-01-07T15:50:04.824+0000 [#####.....] TripAdvisor.restaurants656MB/1.29GB (49.6%)  
2025-01-07T15:50:07.822+0000 [#####.....] TripAdvisor.restaurants730MB/1.29GB (55.2%)  
2025-01-07T15:50:10.822+0000 [#####.....] TripAdvisor.restaurants804MB/1.29GB (60.8%)  
2025-01-07T15:50:13.822+0000 [#####.....] TripAdvisor.restaurants872MB/1.29GB (66.0%)  
2025-01-07T15:50:16.822+0000 [#####.....] TripAdvisor.restaurants 946MB/1.29GB (71.6%)  
2025-01-07T15:50:19.822+0000 [#####.....] TripAdvisor.restaurants 1023MB/1.29GB (77.4%)  
2025-01-07T15:50:22.822+0000 [#####.....] TripAdvisor.restaurants 1.07GB/1.29GB (83.2%)  
2025-01-07T15:50:25.822+0000 [#####.....] TripAdvisor.restaurants 1.15GB/1.29GB (88.9%)  
2025-01-07T15:50:28.822+0000 [#####.....] TripAdvisor.restaurants 1.22GB/1.29GB (94.8%)  
2025-01-07T15:50:31.561+0000 [#####.....] TripAdvisor.restaurants 1.29GB/1.29GB (100.0%)  
2025-01-07T15:50:31.561+0000 1083397 document(s) imported successfully. 0 document(s) failed to import.
```

Βήμα 4: Σύνδεση στη MongoDB με mongosh

Για να διαχειριστούμε τη βάση και να εκτελέσουμε queries, συνδεόμαστε στο MongoDB χρησιμοποιώντας την εντολή:

```
mongosh
```

Μετά την επιτυχή εισαγωγή των δεδομένων στη βάση MongoDB, πραγματοποιήθηκε έλεγχος για την επιβεβαίωση της δημιουργίας και της πληρότητας της βάσης. Χρησιμοποιώντας την εντολή show dbs, επαληθεύτηκε η δημιουργία της βάσης

TripAdvisor, η οποία καταλαμβάνει 343.49 MiB. Στη συνέχεια, με την εντολή `show collections` διαπιστώθηκε η ύπαρξη της συλλογής `restaurants`, και τέλος, με την εντολή `db.restaurants.countDocuments()` επιβεβαιώθηκε ότι εισήχθησαν επιτυχώς όλες οι 1,083,397 εγγραφές της συλλογής. Αυτοί οι έλεγχοι διασφαλίζουν ότι η βάση δεδομένων είναι έτοιμη για ανάλυση.

```
test> show dbs
TripAdvisor  343.49 MiB
admin        40.00 KiB
config       108.00 KiB
local        72.00 KiB
test> use TripAdvisor
switched to db TripAdvisor
TripAdvisor> show collections
restaurants
TripAdvisor> db.restaurants.countDocuments()
1083397
```

Βήμα 5: Δείγμα Δεδομένων

Προκειμένου να αποκτήσουμε εικόνα της δομής κάθε στοιχείου της βάσης, χρησιμοποιούμε την επόμενη εντολή ώστε να εμφανίσουμε το πρώτο στοιχείο.

```
db.restaurants.find().limit(1).pretty()
```

```

TripAdvisor> db.restaurants.find().limit(1).pretty()
[
  {
    _id: ObjectId('677d4d1104e1ddf4b783318c'),
    restaurant_link: 'g10002058-d4586832',
    restaurant_name: 'Au Bout du Pont',
    original_location: [
      'Europe',
      'France',
      'Centre-Val de Loire',
      'Berry',
      'Indre',
      'Rivarennes'
    ],
    country: 'France',
    region: 'Centre-Val de Loire',
    province: 'Berry',
    city: 'Rivarennes',
    address: '2 rue des Dames, 36880 Rivarennes France',
    latitude: 46.635895,
    longitude: 1.386133,
    claimed: 'Claimed',
    awards: [],
    popularity_detailed: '#1 of 1 Restaurant in Rivarennes',
    popularity_generic: '#1 of 1 places to eat in Rivarennes',
    top_tags: [ 'Cheap Eats', 'French', 'European' ],
    price_level: '$',
    price_range: null,
    meals: [ 'Dinner', 'Lunch', 'Drinks' ],
    cuisines: [ 'French', 'European' ],
    special_diets: [],
    features: [
      'Reservations',
      'Seating',
      'Table Service',
      'Wheelchair Accessible'
    ],
    vegetarian_friendly: 'N',
    vegan_options: 'N',
    original_open_hours: null,
    open_days_per_week: null,
    open_hours_per_week: null,
    working_shifts_per_week: null,
    avg_rating: 5,
    total_reviews_count: 13,
    default_language: 'English',
    reviews_count_in_default_language: 4,
    excellent: 3,
    very_good: 1,
    average: 0,
  }
]
TripAdvisor>

```

Το πρώτο στοιχείο της συλλογής restaurants παρουσιάζει τη δομή των δεδομένων και τα διαθέσιμα πεδία που μπορούμε να χρησιμοποιήσουμε για περαιτέρω ελέγχους στα

επόμενα ερωτήματα. Αυτή η αρχική επισκόπηση μας επιτρέπει να εντοπίσουμε τα κρίσιμα πεδία και να σχεδιάσουμε τα queries που θα απαντήσουν στα επόμενα ερωτήματα με ακρίβεια.

Η ύπαρξη κενών τιμών υποδεικνύει ότι τα δεδομένα δεν είναι πλήρη για όλα τα εστιατόρια. Αυτό είναι ένα σημαντικό στοιχείο που πρέπει να έχουμε υπόψη, καθώς μπορεί να επηρεάσει τα αποτελέσματα των queries στα επόμενα ερωτήματα.

Ερώτημα 2: Ανάκτηση Δεδομένων

Υποερώτημα 2.1

Προεργασία

Προκειμένου να φτιάξουμε το κατάλληλο query, πρέπει να κάνουμε προεργασία στα δεδομένα μας. Συγκεκριμένα, πρέπει να εξασφαλίσουμε ειδικά για τα πεδία που περιέχουν κατηγορικές τιμές, ότι περιλαμβάνουν τις τιμές βάσει των οποίων θα κάνουμε την αναζήτηση, καθώς και όλες τις παρεμφερείς τιμές (“city” = “Athens, Athina, Atina”), οι οποίες για οποιοδήποτε λόγο (πχ τυπογραφικό λάθος ή παραλλαγή) θα είχαν αποκλειστεί.

```
TripAdvisor> db.restaurants.distinct("city", { city: { $regex: "^at", $options: "i" } })
[
  'Atalaia',          'Atalandi',          'Atamaria',
  'Atcham',          'Athanasios Diakos', 'Athani',
  'Athboy',          'Athee',             'Athee-sur-Cher',
  'Athenry',         'Athens',            'Atheras',
  'Atherstone',     'Atherton',         'Athies sous Laon',
  'Athikia',        'Athinios',         'Athis-Mons',
  'Athis-de-l'Orne', 'Athleague',        'Athlone',
  'Athus',          'Athy',             'Atiya',
  'Atogo',          'Atorp',            'Atougua da Baleia',
  'Atsiki',         'Atsipopoulo',     'Atsitsa',
  'Attainville',    'Attenborough',    'Attendorn',
  'Attenkirchen',  'Attenschwiller',  'Attersee',
  'Atttert',       'Attignat',         'Attignat-Oncin',
  'Attigny',       'Attin',            'Atting',
  'Attleborough',  'Attnang-Puchheim', 'Atvidaberg',
  'Atworth',       'Atzbach',          'Atzelsdorf',
  'Atzenbrugg'
]
TripAdvisor> db.restaurants.distinct("vegan_options")
[ 'N', 'Y' ]
TripAdvisor> db.restaurants.distinct("meals")
[ 'After-hours', 'Breakfast', 'Brunch', 'Dinner', 'Drinks', 'Lunch' ]
```

Παρατηρούμε πως η τιμή 'Athens' είναι η μοναδική που ανταποκρίνεται στα κριτήρια της αναζήτησής μας και πως δεν υπάρχουν άλλες παρεμφερείς τιμές. Επιπλέον, το πεδίο 'vegan_options' παίρνει τιμές Y ή N και το πεδίο 'meals' περιέχει την τιμή Breakfast.

Όσον αφορά στο υποερώτημα 2.1 σχετικά με την κατάταξη των εστιατορίων βάσει δεδομένων κριτηρίων, παρακάτω φαίνεται το query που χρησιμοποιήθηκε καθώς και τα αποτελέσματα.

```
db.restaurants.find({
  city: "Athens",
  vegan_options: "Y",
  meals: "Breakfast",
  total_reviews_count: { $gt: 100 }
}, {
  restaurant_name: 1,
  address: 1,
  avg_rating: 1,
  total_reviews_count: 1,
  _id: 0
}).sort({
  avg_rating: -1,
  total_reviews_count: -1
}).limit(5)
```

Ο παραπάνω κώδικας αναζητά τα 5 κορυφαία εστιατόρια στην Αθήνα που προσφέρουν vegan επιλογές, σερβίρουν πρωινό και έχουν περισσότερες από 100 αξιολογήσεις. Επιστρέφει μόνο συγκεκριμένα πεδία, όπως το όνομα, τη διεύθυνση, τη μέση βαθμολογία και τον αριθμό αξιολογήσεων. Τα αποτελέσματα ταξινομούνται κατά

φθίνουσα σειρά με βάση τη μέση βαθμολογία και σε περίπτωση ισοβαθμίας, χρησιμοποιείται ο φθίνων αριθμός αξιολογήσεων, με περιορισμό στα 5 πρώτα αποτελέσματα.

```
[
  {
    restaurant_name: 'Vegan Beat',
    address: 'Perikleous 56 Ground floor, Athens 10560 Greece',
    avg_rating: 5,
    total_reviews_count: 754
  },
  {
    restaurant_name: 'Victory Cafe',
    address: 'Fillelinon 22, Athens 10557 Greece',
    avg_rating: 5,
    total_reviews_count: 562
  },
  {
    restaurant_name: 'Montakioy',
    address: 'Stadiou 30 Stoa Korai, Athens 10564 Greece',
    avg_rating: 5,
    total_reviews_count: 208
  },
  {
    restaurant_name: 'Coffee Joint',
    address: 'Vourvachi 5-9 Iosif ton Rogon, Athens 117 43 Greece',
    avg_rating: 5,
    total_reviews_count: 184
  },
  {
    restaurant_name: 'Candy Cat',
    address: 'Persefonis 59, Athens 11854 Greece',
    avg_rating: 5,
    total_reviews_count: 113
  }
]
```

Τα αποτελέσματα που προκύπτουν από το παραπάνω query αναδεικνύουν τα 5 κορυφαία εστιατόρια στην Αθήνα που πληρούν τα κριτήρια του ερωτήματος. Τα εστιατόρια αυτά είναι το **Vegan Beat**, το **Victory Cafe**, το **Montakioy**, το **Coffee Joint**, και το **Candy Cat**. Όλα τους έχουν μέγιστη μέση βαθμολογία (5), ενώ διαφοροποιούνται στον αριθμό συνολικών αξιολογήσεων, που κυμαίνεται από 754

έως 113. Εμφανίζονται το όνομα του εστιατορίου, η διεύθυνση, η μέση βαθμολογία και ο συνολικός αριθμός αξιολογήσεων.

Υποερώτημα 2.2

Για την απάντηση στο υποερώτημα 2.2, δημιουργήθηκε ένα query που αναλύει τους 10 πιο δημοφιλείς τύπους κουζίνας (πεδίο cuisines) στην Αθήνα από εστιατόρια που προσφέρουν vegan επιλογές. Αρχικά, τα δεδομένα φιλτράρονται για να περιλαμβάνουν μόνο εστιατόρια της Αθήνας που υποστηρίζουν vegan επιλογές. Στη συνέχεια, χρησιμοποιείται το στάδιο \$unwind για να διαχωριστούν οι τύποι κουζίνας από τον πίνακα cuisines. Τα δεδομένα ομαδοποιούνται βάσει του κάθε τύπου κουζίνας, υπολογίζοντας τον συνολικό αριθμό εστιατορίων που τον προσφέρουν και τον μέσο όρο του αριθμού αξιολογήσεων αυτών των εστιατορίων. Τέλος, τα αποτελέσματα ταξινομούνται κατά φθίνουσα σειρά βάσει του αριθμού εστιατορίων και περιορίζονται στους 10 κορυφαίους τύπους κουζίνας.

```
db.restaurants.aggregate([
```

```
{
  $match: {
    city: "Athens",
    vegan_options: "Y",
  }
},
{
  $unwind: "$cuisines"
},
{
  $group: {
    _id: "$cuisines",
```

```

    total_restaurants: { $sum: 1 },
    avg_reviews: { $avg: "$total_reviews_count" }
  }
},
{
  $sort: { total_restaurants: -1 }
},
{
  $limit: 10
}
])

```

```

{
  _id: 'Greek',
  total_restaurants: 361,
  avg_reviews: 433.1246537396122
},
{
  _id: 'Mediterranean',
  total_restaurants: 349,
  avg_reviews: 407.1919770773639
},
{
  _id: 'European',
  total_restaurants: 186,
  avg_reviews: 416.4139784946237
},
{
  _id: 'CaFe',
  total_restaurants: 96,
  avg_reviews: 204.51041666666666
},
{
  _id: 'Healthy',
  total_restaurants: 59,
  avg_reviews: 674.4406779661017
},
{
  _id: 'Asian',
  total_restaurants: 43,
  avg_reviews: 278.8837209302326
},
{
  _id: 'Italian',
  total_restaurants: 41,
  avg_reviews: 197.73170731707316
},
{
  _id: 'Fast Food',
  total_restaurants: 37,
  avg_reviews: 173.8108108108108
},
{
  _id: 'Bar',
  total_restaurants: 36,
  avg_reviews: 288.05555555555554
},
{
  _id: 'Seafood',
  total_restaurants: 34,
  avg_reviews: 431.9117647058824
}

```

Από τα αποτελέσματα του query προκύπτει ότι οι πιο δημοφιλείς τύποι κουζίνας στην Αθήνα που προσφέρονται από εστιατόρια με vegan επιλογές είναι η **ελληνική κουζίνα** (Greek) με 361 εστιατόρια και μέσο όρο 433 αξιολογήσεων, ακολουθούμενη από την **μεσογειακή κουζίνα** (Mediterranean) και την **ευρωπαϊκή κουζίνα** (European). Άλλοι σημαντικοί τύποι περιλαμβάνουν την **υγιεινή διατροφή** (Healthy) με υψηλό μέσο όρο αξιολογήσεων (674), την **ασιατική κουζίνα** (Asian) και την **ιταλική κουζίνα** (Italian). Τα αποτελέσματα αυτά δείχνουν την ποικιλία και τη δημοτικότητα των κουζινών που καλύπτουν τις vegan ανάγκες στην Αθήνα, αναδεικνύοντας την ελληνική κουζίνα ως τον πιο κυρίαρχο τύπο.

Υποερώτημα 2.3

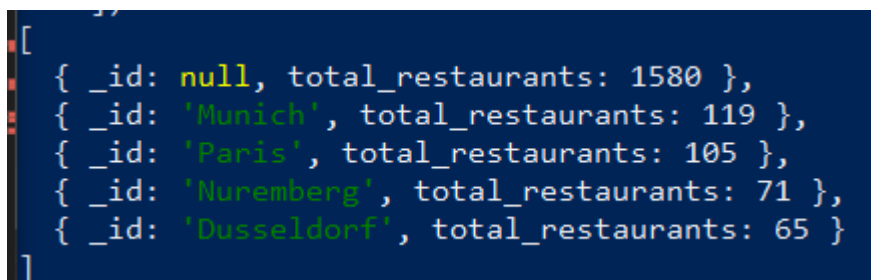
Σε αυτό το query αναζητούνται οι 5 πόλεις εκτός Ελλάδας με τον μεγαλύτερο αριθμό εστιατορίων που προσφέρουν ελληνική κουζίνα. Αρχικά, το φίλτρο \$match περιορίζει τα δεδομένα σε εστιατόρια που βρίσκονται εκτός Ελλάδας (country: { \$ne: "Greece" }) και προσφέρουν ελληνική κουζίνα (cuisines: "Greek"). Στη συνέχεια, τα δεδομένα ομαδοποιούνται ανά πόλη μέσω του \$group, όπου υπολογίζεται ο συνολικός αριθμός των εστιατορίων για κάθε πόλη (total_restaurants: { \$sum: 1 }). Τα αποτελέσματα ταξινομούνται κατά φθίνουσα σειρά με το \$sort και περιορίζονται στις 5 πρώτες πόλεις μέσω του \$limit. Ωστόσο, καθώς δεν υπάρχει φίλτρο για τιμές null στο πεδίο city, το query εμφανίζει επίσης εγγραφές όπου η τιμή της πόλης είναι null, αντιπροσωπεύοντας δεδομένα χωρίς προσδιορισμένη γεωγραφική τοποθεσία. Αυτό υπογραμμίζει την ανάγκη για περαιτέρω καθαρισμό δεδομένων.

```
db.restaurants.aggregate([
{
  $match: {
    country: { $ne: "Greece" },
    cuisines: "Greek",
  }
},
```

```

{
  $group: {
    _id: "$city",
    total_restaurants: { $sum: 1 }
  }
},
{
  $sort: { total_restaurants: -1 }
},
{
  $limit: 5
}
])

```



```

[
  { _id: null, total_restaurants: 1580 },
  { _id: 'Munich', total_restaurants: 119 },
  { _id: 'Paris', total_restaurants: 105 },
  { _id: 'Nuremberg', total_restaurants: 71 },
  { _id: 'Dusseldorf', total_restaurants: 65 }
]

```

Επειδή το query όμως εμφανίζει επίσης εγγραφές όπου η τιμή της πόλης είναι null, προστέθηκε φίλτρο για την εξαίρεση εγγραφών όπου το πεδίο city είναι null (city: { \$ne: null }).

```

db.restaurants.aggregate([
  {
    $match: {

```

```

country: { $ne: "Greece" },
cuisines: "Greek",
city: { $ne: null }
}
},
{
  $group: {
    _id: "$city",
    total_restaurants: { $sum: 1 }
  }
},
{
  $sort: { total_restaurants: -1 }
},
{
  $limit: 5
}
])

```

```

[
  { _id: 'Munich', total_restaurants: 119 },
  { _id: 'Paris', total_restaurants: 105 },
  { _id: 'Nuremberg', total_restaurants: 71 },
  { _id: 'Dusseldorf', total_restaurants: 65 },
  { _id: 'Vienna', total_restaurants: 64 }
]

```

Τα αποτελέσματα του query αναδεικνύουν τις 5 πόλεις εκτός Ελλάδας με τον μεγαλύτερο αριθμό εστιατορίων που προσφέρουν ελληνική κουζίνα. Συγκεκριμένα, η

πόλη **Munich** κατέχει την πρώτη θέση με 119 εστιατόρια, ακολουθούμενη από το **Paris** με 105 εστιατόρια. Στην τρίτη θέση βρίσκεται το **Nuremberg** με 71 εστιατόρια, ενώ ακολουθούν το **Dusseldorf** με 65 και η **Vienna** με 64 εστιατόρια.

Υποερώτημα 2.4

Το συγκεκριμένο query εντοπίζει την πόλη (ανεξαρτήτως χώρας) με τον μεγαλύτερο αριθμό εστιατορίων που πληρούν δύο βασικά κριτήρια: έχουν μέση αξιολόγηση (avg_rating) μεγαλύτερη από 4.5 και περισσότερες από 1000 συνολικές αξιολογήσεις (total_reviews_count). Αρχικά, το φίλτρο \$match φιλτράρει τα δεδομένα για να διασφαλίσει ότι εξετάζονται μόνο εστιατόρια με τις απαιτούμενες αξιολογήσεις και που ανήκουν σε πόλεις όπου το πεδίο city δεν είναι κενό (city: { \$ne: null }). Στη συνέχεια, τα δεδομένα ομαδοποιούνται μέσω του \$group ανά πόλη (_id: "\$city") και υπολογίζεται ο συνολικός αριθμός των εστιατορίων για κάθε πόλη. Τα αποτελέσματα ταξινομούνται κατά φθίνουσα σειρά βάσει του αριθμού των εστιατορίων (\$sort), και περιορίζονται στην κορυφαία πόλη (\$limit: 1).

```
db.restaurants.aggregate([
{
  $match: {
    avg_rating: { $gt: 4.5 },
    total_reviews_count: { $gt: 1000 },
    city: { $ne: null }
  },
{
  $group: {
    _id: "$city",
    total_restaurants: { $sum: 1 }
  }
}
```

```

    },
    {
      $sort: { total_restaurants: -1 }
    },
    {
      $limit: 1
    }
  ]
})

```

```

... ]
[ { _id: 'Rome', total_restaurants: 10 } ]
TripAdvisor>

```

Η ανάλυση ανέδειξε τη Ρώμη (Rome) ως την πόλη με τον μεγαλύτερο αριθμό εστιατορίων που πληρούν τα κριτήρια, με 10 εστιατόρια.

Υποερώτημα 2.5

Το παραπάνω query στοχεύει να αναδείξει τις περιφέρειες της Ελλάδας με τον μεγαλύτερο αριθμό εστιατορίων που έχουν μέση αξιολόγηση (avg_rating) μεγαλύτερη από 4.5. Αρχικά, το φίλτρο \$match περιορίζει τα δεδομένα ώστε να περιλαμβάνουν μόνο εστιατόρια που βρίσκονται στην Ελλάδα (country: "Greece") με υψηλή βαθμολογία και έγκυρη τιμή στο πεδίο region (region: { \$ne: null }). Στη συνέχεια, μέσω του \$group, τα δεδομένα ομαδοποιούνται ανά περιφέρεια, και υπολογίζεται ο συνολικός αριθμός των εστιατορίων για κάθε περιφέρεια (total_restaurants: { \$sum: 1 }). Τέλος, τα αποτελέσματα ταξινομούνται κατά φθίνουσα σειρά βάσει του αριθμού των εστιατορίων (\$sort: { total_restaurants: -1 }), προσφέροντας έτσι μια ξεκάθαρη εικόνα των περιφερειών με την υψηλότερη συγκέντρωση εστιατορίων υψηλής βαθμολογίας.


```

db.restaurants.aggregate([
  {
    $match: {
      country: "Greece",
      avg_rating: { $gt: 4.5 },
      region: { $ne: null }
    }
  },
  {
    $group: {
      _id: "$region",
      total_restaurants: { $sum: 1 }
    }
  },
  {
    $sort: { total_restaurants: -1 }
  }
])

```

```

[
  { _id: 'Attica', total_restaurants: 1328 },
  { _id: 'South Aegean', total_restaurants: 1083 },
  { _id: 'Crete', total_restaurants: 1067 },
  { _id: 'Central Macedonia', total_restaurants: 802 },
  { _id: 'Ionian Islands', total_restaurants: 658 },
  { _id: 'Peloponnese', total_restaurants: 476 },
  { _id: 'Northeast Aegean Islands', total_restaurants: 337 },
  { _id: 'Thessaly', total_restaurants: 265 },
  { _id: 'Central Greece', total_restaurants: 240 },
  { _id: 'Epirus', total_restaurants: 231 },
  { _id: 'West Greece', total_restaurants: 224 },
  { _id: 'East Macedonia and Thrace', total_restaurants: 173 },
  { _id: 'West Macedonia', total_restaurants: 93 },
  { _id: 'Sporades', total_restaurants: 53 },
  { _id: null, total_restaurants: 1 }
]

```

```
[
  { _id: 'Attica', total_restaurants: 1328 },
  { _id: 'South Aegean', total_restaurants: 1083 },
  { _id: 'Crete', total_restaurants: 1067 },
  { _id: 'Central Macedonia', total_restaurants: 802 },
  { _id: 'Ionian Islands', total_restaurants: 658 },
  { _id: 'Peloponnese', total_restaurants: 476 },
  { _id: 'Northeast Aegean Islands', total_restaurants: 337 },
  { _id: 'Thessaly', total_restaurants: 265 },
  { _id: 'Central Greece', total_restaurants: 240 },
  { _id: 'Epirus', total_restaurants: 231 },
  { _id: 'West Greece', total_restaurants: 224 },
  { _id: 'East Macedonia and Thrace', total_restaurants: 173 },
  { _id: 'West Macedonia', total_restaurants: 93 },
  { _id: 'Sporades', total_restaurants: 53 }
]
```

Τα αποτελέσματα δείχνουν ότι η **Αττική** καταλαμβάνει την πρώτη θέση με 1.328 εστιατόρια που έχουν μέση βαθμολογία πάνω από 4.5, ακολουθούμενη από το **Νότιο Αιγαίο** με 1.083 εστιατόρια και την **Κρήτη** με 1.067. Σημαντική παρουσία έχουν επίσης η **Κεντρική Μακεδονία** (802) και τα **Ιόνια Νησιά** (658). Οι υπόλοιπες περιφέρειες, όπως η **Πελοπόννησος** (476), τα **Βορειοανατολικά Αιγαίο** (337) και η **Θεσσαλία** (265), διατηρούν επίσης υψηλά πρότυπα.

Ερώτημα 3: Optimization

Για την απάντηση των υποερωτημάτων του ερωτήματος 3 βελτιστοποιούμε τα queries του Ερωτήματος 2, δημιουργώντας τα κατάλληλα ευρετήρια (indexes) στα πεδία που χρησιμοποιούνται πιο συχνά για φιλτράρισμα, ταξινόμηση ή ομαδοποίηση. Τα ευρετήρια θα επιλέγονται με βάση τα πεδία που χρησιμοποιήθηκαν στα queries.

Χρησιμοποιώντας συμπληρωματικά τη μέθοδο `.explain()` στο 2.2 παράγουμε τα παρακάτω.

```
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'TripAdvisor.restaurants',
```

```

parsedQuery: {
  '$and': [
    { city: { '$eq': 'Athens' } },
    { meals: { '$eq': 'Breakfast' } },
    { vegan_options: { '$eq': 'Y' } },
    { total_reviews_count: { '$gt': 100 } }
  ]
},
indexFilterSet: false,
planCacheShapeHash: '80F76314',
planCacheKey: '472AA15C',
optimizationTimeMillis: 0,
maxIndexedOrSolutionsReached: false,
maxIndexedAndSolutionsReached: false,
maxScansToExplodeReached: false,
prunedSimilarIndexes: false,
winningPlan: {
  isCached: false,
  stage: 'SORT',
  sortPattern: { avg_rating: -1, total_reviews_count: -1 },
  memLimit: 104857600,
  limitAmount: 5,
  type: 'simple',
  inputStage: {
    stage: 'COLLSCAN',
    filter: {

```

```

'$and': [
  { city: { '$eq': 'Athens' } },
  { meals: { '$eq': 'Breakfast' } },
  { vegan_options: { '$eq': 'Y' } },
  { total_reviews_count: { '$gt': 100 } }
]
},
direction: 'forward'
}
},
rejectedPlans: []
},
executionStats: {
  executionSuccess: true,
  nReturned: 5,
  executionTimeMillis: 505,
  totalKeysExamined: 0,
  totalDocsExamined: 1083397,
  executionStages: {
    isCached: false,
    stage: 'SORT',
    nReturned: 5,
    executionTimeMillisEstimate: 478,
    works: 1083404,
    advanced: 5,
    needTime: 1083398,

```

needYield: 0,
saveState: 25,
restoreState: 25,
isEOF: 1,
sortPattern: { avg_rating: -1, total_reviews_count: -1 },
memLimit: 104857600,
limitAmount: 5,
type: 'simple',
totalDataSizeSorted: 8712,
usedDisk: false,
spills: 0,
spilledDataStorageSize: 0,
inputStage: {
 stage: 'COLLSCAN',
 filter: {
 '\$and': [
 { city: { '\$eq': 'Athens' } },
 { meals: { '\$eq': 'Breakfast' } },
 { vegan_options: { '\$eq': 'Y' } },
 { total_reviews_count: { '\$gt': 100 } }
]
 },
 nReturned: 58,
 executionTimeMillisEstimate: 456,
 works: 1083398,
 advanced: 58,

```

    needTime: 1083339,
    needYield: 0,
    saveState: 25,
    restoreState: 25,
    isEOF: 1,
    direction: 'forward',
    docsExamined: 1083397
  }
}
},
queryShapeHash:
'7E1ADDCA46666AD116B58F392721BBE52C6B70622EA9C664B6ACBE6A1556E
C1D',
command: {
  find: 'restaurants',
  filter: {
    city: 'Athens',
    vegan_options: 'Y',
    meals: 'Breakfast',
    total_reviews_count: { '$gt': 100 }
  },
  sort: { avg_rating: -1, total_reviews_count: -1 },
  limit: 5,
  '$db': 'TripAdvisor'
},
serverInfo: {
  host: 'ea3ce78beb3f',

```

```

port: 27017,
version: '8.0.4',
gitVersion: 'bc35ab4305d9920d9d0491c1c9ef9b72383d31f9'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
  internalQueryFrameworkControl: 'trySbeRestricted',
  internalQueryPlannerIgnoreIndexWithCollationForRegex: 1
},
ok: 1
}

```

Το αποτέλεσμα του `explain()` παρέχει λεπτομέρειες για το πώς η MongoDB σχεδίασε και εκτέλεσε το ερώτημα, καθώς και πληροφορίες για τη στρατηγική εκτέλεσης. Τμηματοποιείται ως εξής:

1. queryPlanner

Το τμήμα `queryPlanner` περιγράφει πώς η MongoDB σχεδίασε την εκτέλεση του query. Περιλαμβάνει το `namespace`, δηλαδή τη συλλογή και τη βάση δεδομένων όπου εκτελέστηκε το query, καθώς και το `parsedQuery`, που δείχνει τα φίλτρα του ερωτήματος (π.χ. `city: "Athens"`, `vegan_options: "Y"`, κ.λπ.). Το πιο σημαντικό στοιχείο

εδώ είναι το `winningPlan`, το οποίο αναφέρει ότι η MongoDB χρησιμοποίησε πλήρη σάρωση συλλογής (COLLSCAN), αφού δεν αξιοποιήθηκαν ευρετήρια.

2. winningPlan

Το τμήμα `winningPlan` περιγράφει το επιλεγμένο εκτελεστικό σχέδιο. Περιλαμβάνει διάφορα στάδια, όπως το COLLSCAN, που δείχνει ότι πραγματοποιήθηκε πλήρης σάρωση της συλλογής για την εύρεση των εγγραφών που πληρούν τα κριτήρια. Επιπλέον, περιγράφεται το στάδιο SORT, όπου τα δεδομένα ταξινομήθηκαν με βάση το `avg_rating` και το `total_reviews_count`. Το τελικό στάδιο είναι το PROJECTION_SIMPLE, όπου επιστράφηκαν μόνο τα ζητούμενα πεδία. Το `winningPlan` μας βοηθά να αξιολογήσουμε την αποδοτικότητα του σχεδίου και να διαπιστώσουμε αν η χρήση ευρετηρίων θα μπορούσε να βελτιώσει την απόδοση.

3. command

Το τμήμα `command` παρέχει τις ακριβείς παραμέτρους του query που εκτελέστηκε. Περιλαμβάνει το φίλτρο (filter), που καθορίζει τις συνθήκες αναζήτησης (π.χ. `city: "Athens"`, `total_reviews_count: { $gt: 100 }`), την ταξινόμηση (sort) που εφαρμόστηκε βάσει των πεδίων `avg_rating` και `total_reviews_count`, καθώς και την προβολή (projection) που καθορίζει ποια πεδία εμφανίζονται στο αποτέλεσμα (π.χ. `restaurant_name`, `address`).

4. serverInfo

Το τμήμα `serverInfo` παρέχει πληροφορίες για τον διακομιστή MongoDB που εκτέλεσε το query. Δείχνει τη διεύθυνση (host), τη θύρα (port) και την έκδοση της MongoDB (version: 8.0.4).

5. serverParameters

Το τμήμα `serverParameters` περιγράφει εσωτερικές παραμέτρους που καθορίζουν πώς η MongoDB διαχειρίζεται τη μνήμη και άλλους πόρους κατά την εκτέλεση του query. Για παράδειγμα, το `internalQueryMaxBlockingSortMemoryUsageBytes` δείχνει

το μέγιστο μέγεθος μνήμης που μπορεί να χρησιμοποιηθεί για ταξινομήσεις. Αυτό το τμήμα είναι κυρίως τεχνικό και χρήσιμο για τη βελτιστοποίηση του συστήματος σε περιπτώσεις όπου απαιτούνται μεγάλες ταξινομήσεις ή υπάρχουν περιορισμοί στη μνήμη.

Υποερώτημα 3.1

```
executionStats: {
  executionSuccess: true,
  nReturned: 5,
  executionTimeMillis: 505,
  totalKeysExamined: 0,
  totalDocsExamined: 1083397,
  executionStages: {
    isCached: false,
    stage: 'SORT',
    nReturned: 5,
    executionTimeMillisEstimate: 478,
    works: 1083404,
    advanced: 5,
    needTime: 1083398,
    needYield: 0,
    saveState: 25,
    restoreState: 25,
    isEOF: 1,
    sortPattern: { avg_rating: -1, total_reviews_count: -1 },
    memLimit: 104857600,
    limitAmount: 5,
    type: 'simple',
    totalDataSizeSorted: 8712,
    usedDisk: false,
    spills: 0,
    spilledDataStorageSize: 0,
    inputStage: {
      stage: 'COLLSCAN',
      filter: {
        '$and': [
          { city: { '$eq': 'Athens' } },
          { meals: { '$eq': 'Breakfast' } },
          { vegan_options: { '$eq': 'Y' } },
          { total_reviews_count: { '$gt': 100 } }
        ]
      }
    }
  },
}
```

Το συγκεκριμένο query παρουσιάζει υψηλό κόστος εκτέλεσης, καθώς χρησιμοποιεί πλήρη σάρωση της συλλογής (COLLSCAN) για την αναζήτηση δεδομένων, εξετάζοντας συνολικά 1.083.397 έγγραφα, ενώ επιστρέφει μόνο 5 αποτελέσματα. Ο συνολικός χρόνος εκτέλεσης είναι 505ms, με το στάδιο ταξινόμησης (SORT) να καταναλώνει περίπου 478ms, υποδεικνύοντας ότι η απουσία ευρετηρίων στα πεδία city, meals, vegan_options, και total_reviews_count οδηγεί σε αναποτελεσματική αναζήτηση. Παρότι το query πραγματοποιεί ταξινόμηση και φιλτράρισμα, η έλλειψη ευρετηρίων προκαλεί υψηλό φόρτο στη MongoDB, καθώς αναγκάζεται να εξετάσει κάθε έγγραφο. Για τη βελτιστοποίηση, είναι απαραίτητο να δημιουργηθούν κατάλληλα ευρετήρια στα πεδία φιλτραρίσματος και ταξινόμησης, μειώνοντας τον αριθμό των εξετασθέντων εγγράφων και τον χρόνο εκτέλεσης.

Ευρετήρια

Για τη βελτιστοποίηση του συγκεκριμένου query, προτείνεται η δημιουργία ευρετηρίων στα πεδία city, vegan_options, meals, total_reviews_count, και avg_rating. Συγκεκριμένα, τα πεδία city, vegan_options, και meals χρησιμοποιούνται για φιλτράρισμα με ακρίβεια (\$eq), και ένα ευρετήριο σε αυτά τα πεδία θα μειώσει σημαντικά τον αριθμό εγγράφων που χρειάζεται να εξεταστούν. Το πεδίο total_reviews_count χρησιμοποιείται τόσο για φιλτράρισμα με σύγκριση (\$gt) όσο και για ταξινόμηση, καθιστώντας το απαραίτητο για τη βελτιστοποίηση. Τέλος, το πεδίο avg_rating χρησιμοποιείται για ταξινόμηση, οπότε η συμπερίληψή του στο ευρετήριο θα μειώσει τον χρόνο που απαιτείται για την ταξινόμηση. Με τη δημιουργία ευρετηρίων σε αυτά τα πεδία, η MongoDB θα μπορεί να εκτελεί το query πιο αποτελεσματικά, αποφεύγοντας τη σάρωση ολόκληρης της συλλογής (COLLSCAN) και χρησιμοποιώντας το πιο αποδοτικό στάδιο IXSCAN.

```
db.restaurants.createIndex({ city: 1, vegan_options: 1, meals: 1, total_reviews_count: 1 })
```

```
db.restaurants.createIndex({  
  avg_rating: -1,  
  total_reviews_count: -1
```

```
}}
```

```
db.restaurants.getIndexes()
```

Εκτελώντας ξανά το query του υποερωτήματος 2.1 προσθέτοντας `.explain("executionStats")` εστιάζουμε στα παρακάτω τμήματα του αποτελέσματος που παράγεται.

```
executionStats: {
  executionSuccess: true,
  nReturned: 5,
  executionTimeMillis: 8,
  totalKeysExamined: 58,
  totalDocsExamined: 58,
  executionStages: {
    isCached: false,
    stage: 'PROJECTION_SIMPLE',
    nReturned: 5,
    executionTimeMillisEstimate: 0,
    works: 65,
    advanced: 5,
```

```
inputStage: {
  stage: 'IXSCAN',
  keyPattern: { avg_rating: -1, total_reviews_count: -1 },
  indexName: 'avg_rating_-1_total_reviews_count_-1',
  isMultiKey: false,
  multiKeyPaths: { avg_rating: [], total_reviews_count: [] },
  isUnique: false,
  isSparse: false,
  isPartial: false,
  indexVersion: 2,
  direction: 'forward',
  indexBounds: {
    avg_rating: [ '[MaxKey, MinKey]' ],
    total_reviews_count: [ '[MaxKey, MinKey]' ]
  }
}
```

Μετά την εφαρμογή ευρετηρίων, η εκτέλεση του ερωτήματος βελτιώθηκε σε πολύ μεγάλο βαθμό. Ο χρόνος εκτέλεσης (`executionTimeMillis`) μειώθηκε από **505 ms** σε **8 ms**, καθώς πλέον χρησιμοποιείται σάρωση ευρετηρίου (IXSCAN) αντί για πλήρη

σάρωση συλλογής (COLLSCAN). Επιπλέον, ο αριθμός των εγγράφων που εξετάστηκαν (totalDocsExamined) περιορίστηκε σε **58**, ενώ ο συνολικός αριθμός κλειδιών που εξετάστηκαν (totalKeysExamined) είναι επίσης **58**. Το ευρετήριο που χρησιμοποιήθηκε για αυτή τη βελτιστοποίηση ήταν το **city_1_vegan_options_1_meals_1_total_reviews_count_1**, το οποίο επιτρέπει το γρήγορο φιλτράρισμα των εγγράφων με βάση τα πεδία city, vegan_options, meals και total_reviews_count.

Για εξοικονόμηση χώρου και χρόνου χρησιμοποιούμε την ίδια διαδικασία στα επόμενα υποερωτήματα αναφέροντας τους δείκτες που δημιουργούνται κάθε φορά και μία σύγκριση στα στατιστικά πριν και μετά τη χρήση τους. Η αναλυτική περιγραφή της διαδικασίας παραλείπεται. Επιπλέον κάθε φορά χρησιμοποιείται η εντολή db.restaurants.dropIndexes() ώστε να διαγραφούν τα προηγούμενα ευρετήρια.

Υποερώτημα 3.2

```
db.restaurants.createIndex({ city: 1, vegan_options: 1, cuisines: 1 })
```

Οι εικόνες που ακολουθούν αποτυπώνουν τα αποτελέσματα πριν και μετά τη δημιουργία ευρετηρίων αντίστοιχα.

```
transformby: { cuisines: 1, total_reviews_count: 1 },
inputStage: {
  stage: 'COLLSCAN',
  filter: {
    '$and': [
      { city: { '$eq': 'Athens' } },
      { vegan_options: { '$eq': 'Y' } },
      { cuisines: { '$exists': true } },
      { cuisines: { '$not': { '$eq': [] } } }
    ]
  },
  direction: 'forward'
},
rejectedPlans: []
},
executionStats: {
  executionSuccess: true,
  nReturned: 541,
  executionTimeMillis: 469,
  totalKeysExamined: 0,
  totalDocsExamined: 1083397
}
```


Υποερώτημα 3.3

db.restaurants.createIndex({ country: 1, cuisines: 1, city: 1 })

```
    stage: 'GROUP',
    planNodeId: 3,
    inputStage: {
      stage: 'COLLSCAN',
      planNodeId: 1,
      filter: {
        '$and': [
          { cuisines: { '$eq': 'Greek' } },
          { country: { '$not': { '$eq': 'Greece' } } }
        ]
      },
      direction: 'forward'
    }
  },
  slotBasedPlan: {
    slots: '$$RESULT=s12 env: { s6 = "Greek", s7 = "Greece" }',
    stages: '[3] project [s12 = newObj("_id", s9, "total_restaurants", s12)] \n' +
      '[3] project [s11 = (convert ( s10, int32 )?: s10)] \n' +
      '[3] group [s9] [s10 = count()] spillsSlots[s8] mergingExprs[sum(s8)] \n' +
      '[3] project [s9 = (s1?: null)] \n' +
      '[1] filter {(traverseF(s3, lambda(l3.0) { ((move(l3.0) == s6)?: false) & ((move(l4.0) == s7)?: false) }, false))} \n' +
      '[1] scan s4 s5 none none none none none lowPriority [s1 = city a629-4f8f-8a12-7dc48e8c8466" true false '
  }
},
rejectedPlans: []
},
executionStats: {
  executionSuccess: true,
  nReturned: 2996,
  executionTimeMillis: 870,
  totalKeysExamined: 0,
  totalDocsExamined: 1083397,
  executionStages: {
    stage: 'project',
    planNodeId: 3,
    nReturned: 2996,
    executionTimeMillisEstimate: 860,
    opens: 1,
    closes: 1,
    saveState: 51,
    restoreState: 51,
    isEOF: 1,
    projections: { '12': 'newObj("_id", s9, "total_restaurants", s11) ' },
    inputStage: {
      stage: 'project',
    }
  }
}
```

```

},
executionStats: {
  executionSuccess: true,
  nReturned: 2995,
  executionTimeMillis: 26,
  totalKeysExamined: 7063,
  totalDocsExamined: 0,
  executionStages: {
    stage: 'project',
    planNodeId: 3,
    nReturned: 2995,
    executionTimeMillisEstimate: 16,
    opens: 1,
    closes: 1,
    saveState: 8,
    restoreState: 8,
    isEOF: 1,
    projections: { '18': 'newObj("_id", s15

```

```

isCached: false,
queryPlan: {
  stage: 'GROUP',
  planNodeId: 3,
  inputStage: {
    stage: 'PROJECTION_COVERED',
    planNodeId: 2,
    transformBy: { city: true, _id: false },
    inputStage: {
      stage: 'IXSCAN',
      planNodeId: 1,
      keyPattern: { country: 1, cuisines: 1, city: 1 },
      indexName: 'country_1_cuisines_1_city_1',
      isMultiKey: true,
      multiKeyPaths: { country: [], cuisines: [ 'cuisines' ], city: [] },
      isUnique: false,
      isSparse: false,
      isPartial: false,
      indexVersion: 2,
      direction: 'forward'
    }
  }
}

```

Πριν την υλοποίηση ευρετηρίων, το σύστημα πραγματοποιούσε πλήρη σάρωση συλλογής (COLLSCAN), με αποτέλεσμα να εξετάζονται 1.083.397 έγγραφα, ενώ ο χρόνος εκτέλεσης ήταν 870ms. Μετά τη χρήση ευρετηρίων, ο χρόνος εκτέλεσης μειώθηκε στα 26ms, καθώς χρησιμοποιήθηκε ευρετήριο με το όνομα country_1_cuisines_1_city_1. Το ευρετήριο περιόρισε τα έγγραφα που εξετάστηκαν, μειώνοντας κατά πολύ τον φόρτο αναζήτησης σε 7.063 κλειδιά.

Υποερώτημα 3.4

db.restaurants.createIndex({ avg_rating: 1, total_reviews_count: 1, city: 1 })

```
planNodeId: 3,  
inputStage: {  
  stage: 'COLLSCAN',  
  planNodeId: 1,  
  filter: {  
    '$and': [  
      { avg_rating: { '$gt': 4.5 } },  
      { total_reviews_count: { '$gt': 1000 } },  
      { city: { '$not': { '$eq': null } } }  
    ]  
  },  
  direction: 'forward'  
},  
slotBasedPlan: {  
  slots: '$$RESULT=s12 env: { s6 = 4.5, s7 = 1000 }'  
  stages: '[3] project [s12 = newObj("_id", s9, "tot  
[3] project [s11 = (convert ( s10, int32) ? : s1  
[3] group [s9] [s10 = count()] spillSlots[s8] m  
[3] project [s9 = (s1 ? : null)] \n' +  
[1] filter {(traverseF(s2, lambda(14.0) { ((mov  
0) { ((move(15.0) > s7) ? : false) }, false) && !(traverseF(s  
'    if (typeMatch(16.0, 1024) ? : true) \n' +  
'    then null \n' +  
'    else move(16.0) \n' +  
'== null) ? : false) }, false))))) \n' +  
[1] scan s4 s5 none none none none none none lo  
t] @"cb2fbba6-a629-4f8f-8a12-7dc48e8c8466" true false '  
}  
,  
rejectedPlans: []  
,  
executionStats: {  
  executionSuccess: true,  
  nReturned: 138,  
  executionTimeMillis: 836,  
  totalKeysExamined: 0,  
  totalDocsExamined: 1083397,
```



```

InputStage: {
  stage: 'IXSCAN',
  planNodeId: 1,
  keyPattern: { avg_rating: 1, total_reviews_count: 1, city: 1 },
  indexName: 'avg_rating_1_total_reviews_count_1_city_1',
  isMultiKey: false,
  multiKeyPaths: { avg_rating: [], total_reviews_count: [], city: [] },
  isUnique: false,
  isSparse: false,
  isPartial: false,
  indexVersion: 2,
  direction: 'forward',
  indexBounds: {
    avg_rating: [ '(4.5, inf.0]' ],
    total_reviews_count: [ '(1000, inf.0]' ],
    city: [ '[MinKey, null)', '(null, MaxKey]' ]
  }
},
slotBasedPlan: {
  slots: '$$RESULT=$s18 env: { s5 = IndexBounds("field #0['avg_rating']': (4.5, inf.0], field #2['city']': [MinKey, null), (null, MaxKey]"), s9 = Nothing
  stages: '[3] project [s18 = newObj("_id", s15, "total_restaurants", s17)]
           '[3] project [s17 = (convert ( s16, int32) ? : s16)] \n' +
           '[3] group [s15] [s16 = count()] spillsSlots[s14] mergingExprs[sum(s14)]
           '[3] project [s15 = (s1 ? : null)] \n' +
           '[1] branch {s13} [s1, s12] \n' +
           '[s2, s4] [1] ixscan_generic s5 none s4 none none lowPriority [s2 = 2] @
avg_rating_1_total_reviews_count_1_city_1" true \n' +
           '[s3, s6] [1] nlj inner [] [s7, s8] \n' +
           '  left \n' +
           '    [1] project [s7 = getField(s10, "l"), s8 = getField(s10, "h")]
           '    [1] unwind s10 s11 s9 false \n' +
           '    [1] limit 111 \n' +
           '    [1] coscan \n' +
           '  right \n' +
           '    [1] ixseek s7 s8 none s6 none none [s3 = 2] @"cb2fbb6a6-a629-4fd
reviews_count_1_city_1" true \n'
        }
      },
      rejectedPlans: []
    },
    executionStats: {
      executionSuccess: true,
      nReturned: 138,
      executionTimeMillis: 2,
      totalKeysExamined: 339,
      totalDocsExamined: 0,
      executionStages: {

```

Μετά τη δημιουργία του ευρετηρίου `avg_rating_1_total_reviews_count_1_city_1`, παρατηρούμε σημαντική βελτίωση στην απόδοση της αναζήτησης. Πριν τη χρήση ευρετηρίου, το ερώτημα εκτελέσθηκε με COLLSCAN, με χρόνο εκτέλεσης **836ms** και

εξετάζοντας **1,083,397 έγγραφα**. Αντίθετα, μετά τη χρήση του ευρετηρίου, το στάδιο εκτέλεσης είναι IXSCAN, μειώνοντας τον χρόνο εκτέλεσης σε μόλις **2ms** και εξετάζοντας **339 κλειδιά** χωρίς την ανάγκη εξέτασης εγγράφων.

Υποερώτημα 3.5

db.restaurants.createIndex({ avg_rating: 1, total_reviews_count: 1, city: 1 })

```
inputStage: {
  stage: 'COLLSCAN',
  planNodeId: 1,
  filter: {
    '$and': [
      { country: { '$eq': 'Greece' } },
      { avg_rating: { '$gt': 4.5 } },
      { region: { '$not': { '$eq': null } } }
    ]
  },
  direction: 'forward'
},
slotBasedPlan: {
  slots: '$$RESULT=s12 env: { s6 = "Greece", s7
  stages: '[3] project [s12 = newObj("_id", s9,
    '[3] project [s11 = (convert ( s10, int32) ?
    '[3] group [s9] [s10 = count()] spillsSlots[s
    '[3] project [s9 = (s1 ?; null)] \n' +
    '[1] filter {(traverseF(s3, lambda(14.0) { (
    { ((move(15.0) > s7) ?; false) }, false) && !(traver
    '    if (typeMatch(16.0, 1024) ?; true) \n'
    '    then null \n' +
    '    else move(16.0) \n' +
    '== null) ?; false) }, false))))) \n' +
    '[1] scan s4 s5 none none none none none non
-a629-4f8f-8a12-7dc48e8c8466" true false '
  }
},
rejectedPlans: []
},
executionStats: {
  executionSuccess: true,
  nReturned: 14,
  executionTimeMillis: 794,
  totalKeysExamined: 0,
  totalDocsExamined: 1083397,
  executionStages: {
```

```

    inputStage: {
      stage: "IXSCAN",
      planNodeId: 1,
      keyPattern: { avg_rating: 1, total_reviews_count: 1, city: 1 },
      indexName: 'avg_rating_1 total_reviews_count_1 city_1',
      isMultiKey: false,
      multiKeyPaths: { avg_rating: [], total_reviews_count: [], city: [] },
      isUnique: false,
      isSparse: false,
      isPartial: false,
      indexVersion: 2,
      direction: 'forward',
      indexBounds: {
        avg_rating: [ '(4.5, inf.0)' ],
        total_reviews_count: [ '[MinKey, MaxKey]' ],
        city: [ '[MinKey, MaxKey]' ]
      }
    }
  },
  slotBasedPlan: {
    slots: '$$RESULT=s17 env: { s1 = KS(2B098000000000000000F0F0FE04), s2 = KS(
      "avg_rating" : 1, "total_reviews_count" : 1, "city" : 1}, s12 = "Greece" }',
    stages: '[4] project [s17 = newObj("_id", s14, "total_restaurants", s16)]
      [4] project [s16 = (convert ( s15, int32) ?: s15)] \n' +
      [4] group [s14] [s15 = count()] spillSlots[s13] mergingExprs[sum(s13)]
      [4] project [s14 = (s10 ?: null)] \n' +
      [2] filter {(! (traverseF(s10, lambda(l3.0) { ((\n' +
        if (typeMatch(l3.0, 1024) ?: true) \n' +
        then null \n' +
        else move(l3.0) \n' +
        == null) ?: false) }, false)) && traverseF(s11, lambda(l4.0) { ((move(
      [2] nlj inner [] [s3, s4, s5, s6, s7] \n' +
      left \n' +
      [1] cfilter {(exists(s1) && exists(s2))} \n' +
      [1] ixseek s1 s2 s6 s3 s4 s5 [] @"cb2fbba6-a629-4f8f-8a12-7dc4
      unt_1_city_1" true \n' +
      right \n' +
      [2] limit 111 \n' +
      [2] seek s3 s8 s9 s4 s5 s6 s7 none none [s10 = region, s11 = c
      c8466" true false \n'
    }
  },
  rejectedPlans: []
},
  executionStats: {
    executionSuccess: true,
    nReturned: 14,
    executionTimeMillis: 288,
    totalKeysExamined: 128107,
    totalDocsExamined: 128107,
  }
}

```

Η εκτέλεση του query με **COLLSCAN** εξετάζει όλα τα έγγραφα της συλλογής, με αποτέλεσμα μεγάλο χρόνο εκτέλεσης (794ms) και εξέταση 1.083.397 εγγράφων. Αντίθετα, με τη χρήση του ευρετηρίου **avg_rating_1_total_reviews_count_1_city_1** και **IXSCAN**, ο χρόνος εκτέλεσης μειώνεται στα 288ms, ενώ εξετάζονται μόνο 128.107 κλειδιά αντί για ολόκληρη τη συλλογή.

Συμπερασματικά, φαίνεται ότι η χρήση των πεδίων ταύτισης, ομαδοποίησης και ταξινόμησης είναι ένας αποδοτικός κανόνας δημιουργίας ευρετηρίων, τα οποία επιταχύνουν σε πολύ μεγάλο βαθμό τους χρόνους εκτέλεσης queries εφόσον δεν απαιτείται προσπέλαση όλων των στοιχείων της βάσης δεδομένων.