1.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Koristeći kolekciju 'dvdrent' izradite popis naslova filmova prema kategoriji kojoj pripadaju. Uz naslov filma prikažite kratki opis filma i godinu (`release_year`). Filmove koji se nalaze u više od jedne kategorije izlistajte u oba popisa.

Filmove unutar kategorije poredajte abecedno prema njihovom nazivu te izbjegnite višestruko pojavljivanje naziva istog filma unutar jedne kategorije.

Ispisati samo filmove čije je trajanje (length) duže od 180 minuta.

Nazivi atributa i format se vide iz primjera rezultata:

```
[
  {
    "_id": {
      "category": "Action"
    },
    "value": {
      "films": [
        {
          "title": "Darn Forrester",
          "year": 2006,
          "description": "A Fateful Story of a A Shark And a Explorer who must Succumb a Technical Writer in A Jet Boat"
        },
        {
          "title": "Worst Banger",
          "year": 2006,
          "description": "A Thrilling Drama of a Madman And a Dentist who must Conquer a Boy in The Outback"
        }
      ]
    }
  },

  ...
]
```

Create a list of film titles by their respective category using the 'dvdrent' collection. Display the film description and release year alongside the title. Films belonging to more than one category should be displayed in both lists.

Films inside a category are to be sorted by their title. Also, avoid multiples of the same film in a category.

Display this information only for films that have a duration of over 180 minutes.

Names of attributes and the format are presented in the result example above.

```
db.dvdrent.mapReduce(
    function () {
        if (this.film) {
            const film = this.film
            if (film.length > 180) {
                const filmFormat = {
                    title: film.title,
                    year: film.release_year,
                    description: film.description
                }
                film.categories.forEach(category => {
                    emit({ category: category.name }, { ...filmFormat })
                })
            }
        }
    },
    function (key, values) {
        const filmsFiltered = values.filter((value, index, self) => {
            return self.findIndex(film => film.title === value.title) === index
        })
        filmsFiltered.sort((film1, film2) => {
            return film1.title > film2.title
        })
        return { films: filmsFiltered }
    },
    {
        out: {
            inline: 1
        }
    }
)
```

Run

Save

☺ Correct! Well done!

2.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Pomoću kolekcije `dvdRent`, doznajte:

- sveukupan zarađeni iznos po zaposleniku
- broj naplaćenih posudbi i zarađeni iznos koje su pojedini zaposlenici naplatili **po pojedinim kategorijama** (`payment` mora postojati).
  U slučaju da film pripada više kategorija (npr. "love" i "comedy"), zarada se pripisuje i kategoriji "love" i kategoriji "comedy".

Prikažite ove podatke samo za zaposlenike koji žive u zemlji `United Kingdom`.

Nakon izračunavanja, zaokružite sve decimalne iznose na tri decimale.

Primjer rezultata:

```
[
  {
    "_id": {
      "name": "Diego",
      "lastname": "Diaz"
    },
    "value": {
      "Cartoons": {
        "count": 10,
        "amount": 32.9
      },
      "Drama": {
        "count": 10,
        "amount": 48.901
      },
      "Action": {
        "count": 8,
        "amount": 38.92
      },
      ...
      , "totalAmount": 7592.431
   },
  ...
]
```

Using the `dvdRent` collection, find out:

- the overall charged amount charged by an employee
- the number of charges and charged amounts **by movie categories** charged by an employee (`payment` must exist).
  In case a movie belongs to two categories (eg. "love" and "comedy"), charged amount is attributed both to "love" category and to "comedy" category.

Display this information only for the employees who reside in country `United Kingdom`.

Round all the decimal number to three places.

Run | Save

```
db.dvdrent.mapReduce(
    function () {
        if (this.film && this.staff && this.payment) {
            if (this.staff.address.country === 'United Kingdom') {
                const categories = this.film.categories
                const staff = this.staff
                const payment = this.payment
                categories.forEach(category => {
                    emit(
                        {
                            name: staff.first_name,
                            lastname: staff.last_name
                        },
                        {
                            category: category.name,
                            amount: payment.amount
                        }
                    )
                })
            }
        }
    },
    function (key, values) {
        const categoryAmounts = {
            totalAmount: 0
        }
        const totalAmount = 0
        values.forEach(value => {
            if (!categoryAmounts[value.category]) {
                categoryAmounts[value.category] = {
                    count: 1,
                    amount: value.amount
                }
            } else {
                categoryAmounts[value.category].count++
                categoryAmounts[value.category].amount += value.amount
            }
            categoryAmounts.totalAmount += value.amount
        })
        for (const category in categoryAmounts) {
            if (category !== 'totalAmount') {
                categoryAmounts[category].amount =
                    Math.round(categoryAmounts[category].amount * 1000) / 1000
            }
        }
        categoryAmounts.totalAmount =
            Math.round(categoryAmounts.totalAmount * 1000) / 1000
        return { ...categoryAmounts }
    },
    {
        out: {
            inline: 1
        }
    }
)

```

☺ Correct! Well done!

Result:

3.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Koristeći kolekciju `dvdrent` za svakog zaposlenika (`staff`) i najmove koje je naplatio za filmove u kategoriji "Comedy" u godinama 2014 i 2015 (`payment_date`) odredite:

- ukupan naplaćeni iznos,
- ukupan broj naplaćenih najmova i
- prosječan iznos najma. Zaokružite ga na 3 decimale.

Sve iznose formatirajte na tri decimale.

Nazivi atributa i format se vide iz primjera rezultata:

```
[
...
  {
    "_id": {
      "staff": "Ashley Rivera"
    },
    "value": {
      "totAmount": "19.950",
      "totCnt": 6,
      "avgAmount": "3.325"
    }
  }
  ...
]
```

Using the `dvdrent` collection for each stuff member and rentals he charged in years 2014 and 2015 (**payment_date**) for films in category "Comedy" calculate:

- total paid amount
- total number of charged rentals
- average amount of rent. Round the average to 3 decimal places.

All numbers should be formated to three places.

Attribute names and the overall result format can be seen from the example above.

Run

Save

```
db.dvdrent.mapReduce(
    function () {
        if (this.staff && this.film && this.payment) {
            const categories = this.film.categories
            const payment = this.payment
            const staff = this.staff
            const paymentYear = new Date(payment.payment_date).getFullYear()
            if (paymentYear === 2014 || paymentYear === 2015) {
                categories.forEach(category => {
                    if (category.name === 'Comedy') {
                        emit(
                            {
                                staff: staff.first_name + ' ' + staff.last_name
                            },
                            {
                                amount: payment.amount
                            }
                        )
                    }
                })
            }
        }
    },
    function (key, values) {
        let totAmount = 0
        const totCnt = values.length
        values.forEach(value => {
            totAmount += value.amount
        })
        totAmount = totAmount.toFixed(3)
        const avgAmount = (totAmount / totCnt).toFixed(3)
        return {
            totAmount,
            totCnt,
            avgAmount
        }
    },
    {
        out: {
            inline: 1
        }
    }
)
```

☺ Correct! Well done!

1.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

U kolekciji `cards` želimo prebrojiti sva pojavljivanja riječi (tj. podnizova) :

- beginning
- card
- creature

u atributu `text`(ne moraju biti odijeljeni, npr. `"creatureBeginning"` se broji kao pojavljivanje i `creature` i `beginning`).

Za svaki od navedenih ponizova vratiti ukupan broj pojavljivanja **totCnt** (uspoređujemo *case insenstive*) i sve varijacije podniza koje se pojavljuju (s obzirom na veliko i malo slovo) zajedno s njihovim brojem pojavljivanja **cnt**.

Varijacije **poredati uzlazno po abecedi** (velika slova prije malih).

Nazivi atributa i format se vide iz primjera rezultata:

```
[
...
 {
   "_id": "creature",
   "value": {
     "totCnt": 17577,
     "variations": [
       {
         "variation": "Creature",
         "cnt": 483
       },
       {
         "variation": "creature",
         "cnt": 17094
       }
     ]
   }
 }
 ...
]
```

In the `cards` collection we want to count all the occurrences of words (substrings):

- beginning
- card * creature

in the attribute `text` (they do not have to be separated, eg in `"creatureBeginning"` both `creature` and `beginning` occur and count).
For each of the given substrings, return the total number of occurences **totCnt** (use *case insensitive* comparison) and all substring variations that occur along with corresponding count **cnt** (with respect to letter case).
**Sort variations alphabetically ** (upper case before lower case).
Attribute names and the overall result format can be seen from the example above.

Run

Save

```
db.cards.mapReduce(
    function () {
        if (this.text) {
            const text = this.text
            if (text.toLowerCase().includes('beginning')) {
                const regex = /beginning/gi
                const variations = text.match(regex)
                emit('beginning', { variations })
            }
            if (text.toLowerCase().includes('card')) {
                const regex = /card/gi
                const variations = text.match(regex)
                emit('card', { variations })
            }
            if (text.toLowerCase().includes('creature')) {
                const regex = /creature/gi
                const variations = text.match(regex)
                emit('creature', { variations })
            }
        }
    },
    function (key, values) {
        let totCnt = 0
        const variationsObj = {}
        const variations = []
        values.forEach(value => {
            value.variations.forEach(variation => {
                totCnt += 1
                if (!variationsObj[variation]) {
                    variationsObj[variation] = 1
                } else {
                    variationsObj[variation]++
                }
            })
        })
        for (const variation in variationsObj) {
            variations.push({
                variation,
                cnt: variationsObj[variation]
            })
        }
        variations.sort((var1, var2) => {
            return var1.variation.localeCompare(var2.variation)
        })
        return {
            totCnt,
            variations
        }
    },
    {
        out: {
            inline: 1
        }
    }
)
```

☺ Correct! Well done!

2.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Write an M/R query that will 'invert' the collection `nobelprizes` in such a way that for Nobel prizes:

- in the interval [1900, 1915]
- which are **not** of category "peace"

for each laureate, return the list of his prizes **sorted by years in ascending order**. Only the year and category should be returned for the prize.

Example result:


```
[
  {
    "_id": {
      "firstname": "Albert Abraham",
      "surname": "Michelson"
    },
    "value": {
      "prizes": [
        {
          "year": "1907",
          "category": "physics"
        }
      ]
    }
  },
  ...
  {
    "_id": {
      "firstname": "Marie",
      "surname": "Curie, née Sklodowska"
    },
    "value": {
      "prizes": [
        {
          "year": "1903",
          "category": "physics"
        },
        {
          "year": "1911",
          "category": "chemistry"
        }
      ]
    }
  },
  ...
]
```


Run

Save

```
db.nobelprizes.mapReduce(
    function () {
        if (this.year && this.category && this.laureates) {
            const year = this.year
            const category = this.category
            const laureates = this.laureates
            if (
                Number(year) >= 1900 &&
                Number(year) <= 1915 &&
                category !== 'peace'
            ) {
                laureates.forEach(laureate => {
                    emit(
                        {
                            firstname: laureate.firstname,
                            surname: laureate.surname
                        },
                        {
                            year,
                            category
                        }
                    )
                })
            }
        }
    },
    function (key, values) {
        const prizes = values
        prizes.sort((prize1, prize2) => {
            return Number(prize1.year) > Number(prize2.year)
        })
        return {
            prizes
        }
    },
    {
        out: {
            inline: 1
        }
    }
)
```

☺ Correct! Well done!

3.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Koristeći kolekciju `nobelprizes` izradite popis svih laureata za pojedinu kategoriju nobelove nagrade. Popis treba sadržavati ime, prezime i godinu dodjele nagrade za svakog pojedinog laureata.

Popis laureta za svaku kategoriju neka bude:

- poredan kronološki prema godini dodjele nagrade uzlaznim poretkom.
- u slučaju da više laureta ima istu godinu dodjele, poredajte ih abecednim redoslijedom prema prezimenu.
- ako su i prezimena ista - poredati po imenu (postoje zapisi koji nemaju prezime - riječ je o organizacijama koje su dobile npr. nagradu za mir)

... ili, mogli bi reći neformalno u SQL dijalektu - "ORDER BY year, lastname, firstname".

Nazivi atributa i format se vide iz primjera rezultata:

Primjer rezultata:

```
[
  {
    "_id": {
      "category": "chemistry"
    },
    "value": {
      "laureates": [
      {
          "year": "1901",
          "firstname": "Jacobus Henricus",
          "surname": "van 't Hoff"
        },
        {
          "year": "1902",
          "firstname": "Hermann Emil",
          "surname": "Fischer"
        },
        {
          "year": "1903",
          "firstname": "Svante August",
          "surname": "Arrhenius"
        },

            ...

        {
          "year": "1931",
          "firstname": "Friedrich",
          "surname": "Bergius"
        },
        {
          "year": "1931",
          "firstname": "Carl",
          "surname": "Bosch"
        },


           ...
]
```

Create a list of all the laureates for each category of the Nobel prize using the '`nobelprizes`' collection. The list has to contain the first name, last name and year of laureation for each laureate.

The list should be sorted:

- ascending by year within each prize category.
- in the case when more laurettes have the same year, order them in alphabetical order by surname.
- in the case when the surnames are the same - order them in alphabetical order by firstname

...or, in SQL dialect, we could say "ORDER BY year, lastname, firstname".

The names of attributes and the format can be seen in the result example above.

Run

Save

```
db.nobelprizes.mapReduce(
    function () {
        if (this.year && this.category && this.laureates) {
            const year = this.year
            const category = this.category
            const laureates = this.laureates
            const mappedLaureates = laureates.map(laureate => {
                return {
                    year,
                    firstname: laureate.firstname,
                    surname: laureate.surname
                }
            })
            emit({ category }, { laureates: mappedLaureates })
        }
    },
    function (key, values) {
        const laureates = []
        values.forEach(value => {
            value.laureates.forEach(laureate => {
                laureates.push(laureate)
            })
        })
        laureates.sort((laureate1, laureate2) => {
            return (
                Number(laureate1.year) - Number(laureate2.year) ||
                laureate1.surname.localeCompare(laureate2.surname) ||
                laureate1.firstname.localeCompare(laureate2.firstname)
            )
        })
        return {
            laureates
        }
    },
    {
        out: {
            inline: 1
        }
    }
)

```

☺ Correct! Well done!

2.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Pomoću kolekcije `nobelprizes` doznajte broj laureata i različitih kategorija po godinama, te omjer broja laureata i kategorija po godinama.
Npr. omjer 2018. je iznosio 5/4 = 1.25 (stvarni podatak se može razlikovati).
Zaokružite omjer na *dvije* decimale i nazovite ga `laureateCategoryRatio`.

Primjer rezultata:

```
[
  {
    "_id": "2018",
    "value": {
      "laureates": 5,
      "categories": 4,
      "laureateCategoryRatio": 1.25
    }
  },
  ...
]
```

Using the `nobelprizes` collection find out the number of laureates and different categories per years, and the ratio of number of laureates and categories per years.
Eg. The ratio in year 2018 was 5/4 = 1.25 (the real information may be different).
Round up the ratio to **two** digits and name it `laureateCategoryRatio`.

Example of the result set is shown above.

Run

Save

```
db.nobelprizes.mapReduce(
    function () {
        if (this.year && this.category && this.laureates) {
            const year = this.year
            const category = this.category
            const laureates = this.laureates
            emit(year, { category, laureates })
        }
    },
    function (key, values) {
        const categories = []
        const laureates = []
        values.forEach(value => {
            if (!categories.includes(value.category)) {
                categories.push(value.category)
            }
            value.laureates.forEach(laureate => {
                laureates.push(laureate)
            })
        })
        return {
            laureates: laureates.length,
            categories: categories.length,
            laureateCategoryRatio:
                Math.round((laureates.length / categories.length) * 100) / 100
        }
    },
    {
        out: {
            inline: 1
        }
    }
)
```

☺ Correct! Well done!

3.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Koristeći kolekciju `dvdrent` za svaku glumca ispisati:

- **različite** kategorije filmova u kojima je glumio/la
- **različite** filmove u kojima je glumio/la

U obzir uzeti samo filmove koji imaju više od jedne kategorije.
Npr. ako u kolekciji imamo sljedeće posudbe filmova:

```
Film1, [Action], [Actor1, Actor2]
Film2, [SF, Horror], [Actor1]
Film3, [Drama, SF], [Actor1, Actor2]
```

onda:

- Film1 se zanemaruje jer ima samo jednu kategoriju
- Actor1 ima kategorije [Drama, Horor, SF] i filmove [Film2, Film3]
- Actor2 ima kategorije [Drama, SF] i filmove [Film3]

Filmove i kategorije u rezultatu poredati abecedno. Za glumca ispisat id i puno ime.
Nazivi atributa i format se vide iz primjera rezultata:

```
[
  {
    "_id": {
      "id": 1,
      "fullname": "Penelope Guiness"
    },
    "value": {
      "categories": [
        "Children",
        "Classics",
        "Comedy",
        "Family",
        "Horror",
        "Sports"
      ],
      "films": [
        "King Evolution",
        "Language Cowboy",
        "Rules Human",
        "Vertigo Northwest"
      ]
    }
  },
  ...
]
```

Using the `dvdrent` collection, give for each actor: * ** different ** movie categories in which (s)he acted * ** different ** movies in which (s)he acted

** Consider only movies that have more than one category **, e.g. if we have the following movie rentals in the collection:

```
Movie1, [Action], [Actor1, Actor2]
Film2, [SF, Horror], [Actor1]
Movie3, [Drama, SF], [Actor1, Actor2]
```

then: * Film1 is neglected because it has only one category * Actor1 has categories [Drama, Horror, SF] and movies [Movie2, Movie3] * Actor2 has categories [Drama, SF] and Movies [Film3] ** The movies and categories in the result should be listed in alphabetical order.** For the actor, print id and full name. Attribute names and the overall result format can be seen from the example: (see above)

Run
Save

```
db.dvdrent.mapReduce(
    function () {
        if (this.film) {
            const title = this.film.title
            const categories = this.film.categories
            const actors = this.film.actors
            if (categories.length > 1) {
                actors.forEach(actor => {
                    emit(
                        {
                            id: actor.actor_id,
                            fullname: actor.first_name + ' ' + actor.last_name
                        },
                        {
                            film: title,
                            categories
                        }
                    )
                })
            }
        }
    },
    function (key, values) {
        const films = []
        const categories = []
        values.forEach(value => {
            value.categories.forEach(category => {
                if (!categories.includes(category.name)) {
                    categories.push(category.name)
                }
            })
            if (!films.includes(value.film)) {
                films.push(value.film)
            }
        })
        films.sort()
        categories.sort()
        return {
            categories,
            films
        }
    },
    {
        out: {
            inline: 1
        }
    }
)

```

☺ Correct! Well done!

1.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Korištenjem kolekcije 'dvdrent' i M/R upita potrebno je odrediti i prikazati koliko su puta stanovnici pojedine zemlje iznajmili film te koliko je na njima zaradila iznajmljivačka kuća. U obzir treba uzeti samo filmove koji imaju zapis o uplati, traju duže od 120 minuta te njihov "rental_rate" iznosi više od 3.
Ukupan iznos (zaradu - amount) zaokružiti na tri decimale.

Nazivi atributa i format se vide iz primjera rezultata:

```
[
  {
    "_id": {
      "country": "Anguilla"
    },
    "value": {
      "count": 1,
      "amount": 4.99
    }
  },
  {
    "_id": {
      "country": "Argentina"
    },
    "value": {
      "count": 2,
      "amount": 9.98
    }
  },
  {
    "_id": {
      "country": "Brazil"
    },
    "value": {
      "count": 5,
      "amount": 37.95
    }
  },
     ...
]
```

By using the dvdrent collection and M/R querying determine and display how many times did the citizens of one country rent a film and how much profit did the rental company get from those rents. Consider only those films that have payment info, are over 120 minutes long and have a „rental_rate“ over 3. **Round the amount to three decimal places.** Names of attributes and the required format are presented in the result example above.

Run

Save

```
db.dvdrent.mapReduce(
    function () {
        if (this.film && this.customer && this.payment) {
            const film = this.film
            const customer = this.customer
            const payment = this.payment
            if (film.length > 120 && film.rental_rate > 3) {
                emit(
                    {
                        country: customer.address.country
                    },
                    {
                        payment: payment.amount
                    }
                )
            }
        }
    },
    function (key, values) {
        let amount = 0
        const count = values.length
        values.forEach(value => {
            amount += value.payment
        })
        return {
            count,
            amount: Math.round(amount * 1000) / 1000
        }
    },
    {
        out: {
            inline: 1
        }
    }
)

```

☺ Correct! Well done!

3.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Koristeći kolekciju `dvdrent` izračunajte prosječan iznos plaćanja najma u 2016 godini i koliko je postotno različit od prosjeka svih prethodnih godina zajedno. Prosjek zaokružiti na 4 decimale, a postotnu razliku na dvije i formatirati na dvije.

Nazivi atributa i format se vide iz primjera rezultata:

```
[
  {
    "_id": {
      "year": 2016
    },
    "value": {
      "avg": 4.1234,
      "diff": "-1.20%"
    }
  }
]
```

Using the `dvdrent` collection calculate the average amount of rent payment in 2016 and difference in percentages from the average of all the previous years together. Round the average to 4 decimal places, and the percentage difference to 2 decimal places (also format to two places).

Attribute names and the overall result format can be seen from the example: (see above).

Run

Save

```
db.dvdrent.mapReduce(
    function () {
        if (this.rental_date && this.payment) {
            if (this.payment.payment_date) {
                const payment = this.payment
                const payment_date = payment.payment_date
                const year = new Date(payment_date).getFullYear()
                if (year <= 2016) {
                    emit(
                        {
                            year: 2016
                        },
                        {
                            year,
                            payment: payment.amount
                        }
                    )
                }
            }
        }
    },
    function (key, values) {
        let amount2016 = 0
        let count2016 = 0
        let amountPrev = 0
        let countPrev = 0
        values.forEach(value => {
            if (value.year === 2016) {
                amount2016 += value.payment
                count2016++
            } else {
                amountPrev += value.payment
                countPrev++
            }
        })
        let diff = 0
        const avg2016 = amount2016 / count2016
        const avgPrev = amountPrev / countPrev
        if (avgPrev > 0) {
            diff = ((avg2016 - avgPrev) / avgPrev) * 100
        }
        return {
            avg: Math.round(avg2016 * 10000) / 10000,
            diff: diff.toFixed(2) + '%'
        }
    },
    {
        out: {
            inline: 1
        }
    }
)
```

☺ Correct! Well done!

1.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Koristeći kolekciju `dvdrent` izradite popis naslova filmova za svakog glumca. U obzir uzmite samo filmove s rejtingom "R".
Filmove poredajte abecedno i izbjegnite višestruko pojavljivanje istog filma.
Nazivi atributa i format se vide iz primjera rezultata:

```
{
  "_id": {
    "actorFName": "Fay",
    "actorLName": "Kilmer"
  },
  "value": {
    "films": [
      "Bird Independence",
      "Harry Idaho",
      "Nemo Campus",
      "Racer Egg",
      "Secret Groundhog"
    ]
  }
}
```

Using the `dvdrent` collection create a list of film titles for each actor. Take into acount only movies with the rating "R".
Sort the movies alphabetically and avoid repeating the title of the movie.

Attribute names and the overall result format can be seen from the example above.

Run

Save

```
db.dvdrent.mapReduce(
    function () {
        if (this.film && this.film.actors) {
            const film = this.film
            if (film.rating === 'R')
                film.actors.forEach(actor => {
                    emit(
                        {
                            actorFName: actor.first_name,
                            actorLName: actor.last_name
                        },
                        {
                            film: film.title
                        }
                    )
                })
        }
    },
    function (key, values) {
        const films = []
        values.forEach(value => {
            if (!films.includes(value.film)) {
                films.push(value.film)
            }
        })
        films.sort()
        return {
            films
        }
    },
    {
        out: { inline: 1 }
    }
)

```

☺ Correct! Well done!

2.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

U kolekciji `cards` želimo prebrojiti sva pojavljivanja riječi (tj. podnizova):

- battlefield
- creature
- death
- victory

u atributu `text`.
(ne moraju biti odijeljeni, npr. `"battlefieldcreature"` se broji kao pojavljivanje i `battlefield` i `creature`).

Za svaki od navedenih ponizova vratiti broj pojavljivanja kada uspoređujemo *case senstive* i kad uspoređujemo *case insenstive*, te postotak *case senstive* pojavljivanja. Postotak zaokružiti na dvije decimale i formatirati kao postotak s dvije decimale(%). Nazivi atributa se vide iz primjera rezultata:

```
[
  {
    "_id": "battlefield",
    "value": {
      "cntCI": 5094,
      "cntCS": 5084,
      "avg": "99.80%"
    }
  },
  {
    "_id": "creature",
    "value": {
      "cntCI": 17577,
      "cntCS": 17094,
      "avg": "97.25%"
    }
  },
  ...
]
```

Count all the occurrences of words (substrings):

 * battlefield * creatures * death * victory

in the `cards` collection, attribute `text`.
(they do not have to be separated, for example, in "battlefieldcreature" both "battlefield" and "creature" occur).

For each of these substrings, return the number of occurences when comparing *case sensitive* and *case insensitive*, and the percentage *case senstive* occurences.
Round the percentage to two decimal places and format it as a percentage with two decimals (%).
Attribute names and the overall result format can be seen from the example: (see above)

Run
Save

```
db.cards.mapReduce(
    function () {
        if (this.text) {
            const text = this.text
            if (text.toLowerCase().includes('battlefield')) {
                const regexInsensitive = new RegExp('battlefield', 'gi')
                const regexSensitive = new RegExp('battlefield', 'g')
                const matchedInsensitive = text.match(regexInsensitive)
                    ? text.match(regexInsensitive).length
                    : 0
                const matchedSensitive = text.match(regexSensitive)
                    ? text.match(regexSensitive).length
                    : 0
                emit('battlefield', {
                    insensitive: matchedInsensitive,
                    sensitive: matchedSensitive
                })
            }
            if (text.toLowerCase().includes('creature')) {
                const regexInsensitive = new RegExp('creature', 'gi')
                const regexSensitive = new RegExp('creature', 'g')
                const matchedInsensitive = text.match(regexInsensitive)
                    ? text.match(regexInsensitive).length
                    : 0
                const matchedSensitive = text.match(regexSensitive)
                    ? text.match(regexSensitive).length
                    : 0
                emit('creature', {
                    insensitive: matchedInsensitive,
                    sensitive: matchedSensitive
                })
            }
            if (text.toLowerCase().includes('death')) {
                const regexInsensitive = new RegExp('death', 'gi')
                const regexSensitive = new RegExp('death', 'g')
                const matchedInsensitive = text.match(regexInsensitive)
                    ? text.match(regexInsensitive).length
                    : 0
                const matchedSensitive = text.match(regexSensitive)
                    ? text.match(regexSensitive).length
                    : 0
                emit('death', {
                    insensitive: matchedInsensitive,
                    sensitive: matchedSensitive
                })
            }
            if (text.toLowerCase().includes('victory')) {
                const regexInsensitive = new RegExp('victory', 'gi')
                const regexSensitive = new RegExp('victory', 'g')
                const matchedInsensitive = text.match(regexInsensitive)
                    ? text.match(regexInsensitive).length
                    : 0
                const matchedSensitive = text.match(regexSensitive)
                    ? text.match(regexSensitive).length
                    : 0
                emit('victory', {
                    insensitive: matchedInsensitive,
                    sensitive: matchedSensitive
                })
            }
        }
    },
    function (key, values) {
        let cntCI = 0
        let cntCS = 0
        values.forEach(value => {
            cntCI += value.insensitive
            cntCS += value.sensitive
        })
        const avg = ((cntCS / cntCI) * 100).toFixed(2) + '%'
        return {
            cntCI,
            cntCS,
            avg
        }
    },
    {
        out: { inline: 1 }
    }
)
```

☺ Correct! Well done!

3.

1pts Correct | 0 pts Unanswered | 0 pts Incorrect

Pomoću kolekcije `nobelprizes`, za zadnjih 5 godina, dohvatite prezimena laureata i njihove kategorije.

Laureate poredati po prezimenu.

(trenutnu godinu možete odrediti s npr. `(new Date()).getFullYear()`)

Primjer rezultata:

```
[
  (...),
  {
    "_id": "2016",
    "value": {
      "laureates": [
        {
          "surname": "Dylan",
          "category": "literature"
        },
        {
          "surname": "Feringa",
          "category": "chemistry"
        },
        {
          "surname": "Haldane",
          "category": "physics"
        },
        (...)
        {
          "surname": "Stoddart",
          "category": "chemistry"
        },
        {
          "surname": "Thouless",
          "category": "physics"
        }
      ]
    }
  },
  (...)
]
```

```
db.nobelprizes.mapReduce(
    function () {
        if (this.category && this.year && this.laureates) {
            const currYear = new Date().getFullYear()
            const year = this.year
            const category = this.category
            const laureates = this.laureates
            if (currYear - Number(year) <= 5) {
                laureates.forEach(laureate => {
                    emit(year, {
                        surname: laureate.surname,
                        category
                    })
                })
            }
        }
    },
    function (key, values) {
        const laureates = values
        laureates.sort((l1, l2) => {
            return l1.surname.localeCompare(l2.surname)
        })
        return {
            laureates
        }
    },
    {
        out: {
            inline: 1
        }
    }
)
```

Run

Save

☺ Correct! Well done!