УРОК 19.1. ПРАКТИЧЕСКАЯ ОТРАБОТКА MONGO DB

ПРАКТИЧЕСКАЯ ОТРАБОТКА

$bucket - это этап агрегации, используемый для категоризации документов по группам

1. Из коллекции sample_airbnb.listingsAndReviews отсортировать по цене за ночь недвижимость Барселоны:
    - 0-50
    - 50-100
    - 100-1000
    - Дороже 1000

```
Unset
[
  {
   $match: {
    "address.country_code": "ES"}
   },
  {
   $bucket: {
    groupBy: "$price",
    boundaries: [0, 50, 100, 1000], // Задаем диапазоны
    default: "EXPENSIVE", // для документов вне указанных диапазонов
    output: {
     count: { $sum: 1 }, // Считаем количество в каждом диапазоне
     property: { $push: "$name" } // Собираем названия
    }
   }
  }
]
```

2. Из коллекции sample_airbnb.listingsAndReviews создать новую коллекцию reviews и отправить в нее все отзывы из исходной коллекции. 1 документ - 1 отзыв.

Unset

```
[
 {
  $match:
   /**
    * query: The query in MQL.
    */
   {
    reviews: {
     $ne: [],
    },
   },
 },
 {
  $project:
   /**
    * specifications: The fields to
    *  include or exclude.
    */
   {
    _id: 0,
    reviews: 1,
   },
 },
 {
  $unwind: {
   path: "$reviews",
  },
 },
 {
  $set: {
   date: "$reviews.date",
   listing_id: "$reviews.listing_id",
   reviewer_id: "$reviews.reviewer_id",
   reviewer_name: "$reviews.reviewer_name",
   comments: "$reviews.comments",
  },
 },
 {
  $unset:
```

```
    /**
     * Provide the field name to exclude.
     * To exclude multiple fields, pass the field names in an array.
     */
    "reviews",
  },
  {
    $out: "reviews",
  },
]
```

$match:

- Этот этап фильтрует документы, оставляя только те, у которых поле "reviews" не является пустым массивом ($ne: []).
- Все документы, которые не соответствуют этому условию, будут исключены из последующих этапов агрегации.

$project:

- Этот этап проецирует (выбирает) только определенные поля для вывода.
- В данном случае, выбираются все поля из "reviews" массива и включаются в вывод, при этом _id исключается из результата.

$unwind:

- Этот этап разворачивает массив "reviews", создавая дубликат документа для каждого элемента массива.
- Это делается для того, чтобы в дальнейшем можно было работать с каждым отзывом как с отдельным документом.

$set:

- Этот этап используется для создания новых полей на основе значений внутри объекта "reviews".
- Новые поля (date, listing_id, reviewer_id, reviewer_name, comments) присваиваются значениям из соответствующих полей внутри "reviews".

$unset:

- Этот этап удаляет поле "reviews" из документа.
- Теперь все значения отзыва стали отдельными полями, и сам объект "reviews" больше не нужен.

$out:

- Этот этап записывает результат агрегации в новую коллекцию "reviews".
- Вся обработка и преобразование приводят к созданию новой коллекции, где каждый отзыв представлен отдельным документом.

3. Из коллекции sample_restaurants.restaurants: выяснить, в каких диапазонах средняя оценка этих ресторанов, расположив их по группам [0, 20, 40, 60, 80, 100]

   Таким образом мы можем выяснить группу лучших и худших ресторанов.

```
Unset
db.restaurants.aggregate([
 {
  $match: {
   "grades": { $exists: true, $ne: [] } // Фильтруем документы без оценок
  }
 },
 {
  $project: {
   name: 1,
   score: { $avg: "$grades.score" } // Вычисляем средний балл
  }
 },
 {
  $bucket: {
   groupBy: "$score",
   boundaries: [0, 20, 40, 60, 80, 100], // Задаем диапазоны баллов
   default: "unknown", // для документов вне указанных диапазонов
   output: {
    count: { $sum: 1 }, // Считаем количество ресторанов в каждом диапазоне
    restaurants: { $push: "$name" } // Собираем названия ресторанов в каждом
   }
  }
 }
]);
```

- $match - чтобы удостовериться, что мы работаем с документами, имеющими информацию об оценках
- $project вычисляет средний балл для каждого ресторана.
- $bucket категоризирует рестораны в различные диапазоны и для удобства показывает количество ресторанов в данном диапазоне

4. Добавить в коллекцию sample_restaurants.restaurants поля: среднее значение, оценку, из отзывов и добавить количество отзывов для удобства.

   Для этого понадобятся стадии:
   - $set для создания новых полей
   - $merge позволяет вам обновлять существующие документы. whenMatched: "merge" указывает, что существующие документы будут обновлены, а новые документы будут проигнорированы. Необходимо использовать $merge вместо $out х так как мы хотим обновить существующие документы в коллекции и не добавлять новые документы.

```
Unset
[
 {
  $set:
   /**
    * field: The field name
    * expression: The expression.
    */
   {
    avg_score: {
     $avg: "$grades.score",
    },
    number_of_reviews: {
     $size: "$grades",
    },
   },
```

```
 },
 {
  $merge: {
   into: "restaurants",
   // Provide the name of the output collection.
   whenMatched: "merge", // Specify the merge behavior for existing documents.
  },
 },
]
```

5. Из коллекции Sample_supplies.sales: добавить новое поле - total - итоговая сумма покупки, которое состоит из всех элементов items ,в котором мы умножаем количество на стоимость одной единицы.

- $map используется для итерации по массиву items.
- Для каждого элемента $multiply используется для вычисления произведения price и quantity.
- $sum используется для суммирования всех вычисленных произведений, предоставляя общее значение.

Unset

```
[
 {
  $addFields: {
   total: {
    $sum: {
     $map: {
      input: "$items",
      as: "item",
      in: {
       $multiply: [
        "$$item.price",
```


```
        "$$item.quantity",
       ],
      },
     },
    },
   },
  },
 },
 {
  $merge: {
   $merge: {
    into: "sales",
    whenMatched: "merge",
   },
  },
 },
]
```