

УРОК 16.1. СОЗДАНИЕ И НАПОЛНЕНИЕ КОЛЛЕКЦИЙ. АГРЕГАЦИЯ

АГРЕГАЦИЯ И ГРУППИРОВКА	2
ФУНКЦИОНАЛ АГРЕГАЦИЙ	3
AGGREGATION PIPELINE	4
ПОПУЛЯРНЫЕ СТАДИИ АГРЕГАЦИИ	6
ФУНКЦИИ, ИСПОЛЬЗУЕМЫЕ ПРИ АГРЕГАЦИИ	7
ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ	8



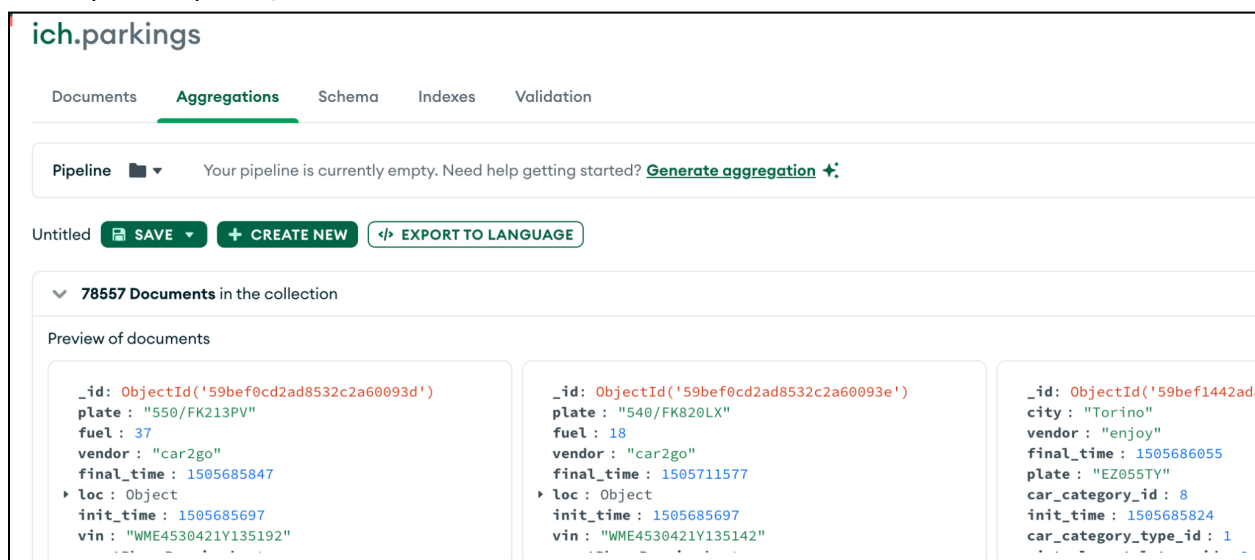
АГРЕГАЦИЯ И ГРУППИРОВКА

Агрегация и группировка состоит из стадий и функций.
Вот примерное соответствие SQL и NoSQL запросов :

SQL	MongoDB
WHERE	\$match
GROUP BY	\$group
HAVING	\$match
SELECT	\$project
ORDER BY	\$sort
//LIMIT	\$limit
SUM	\$sum
COUNT	\$sum

Для группировки агрегации используются следующие основные стадии или этапы.
Сам процесс передачи данных от одной стадии к другой называется Aggregation Pipeline.

В compass агрегации находятся в отдельной вкладке :



The screenshot shows the MongoDB Compass interface with the 'Aggregations' tab selected. The top navigation bar includes 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. Below the navigation bar, there is a 'Pipeline' section with a dropdown menu and a message: 'Your pipeline is currently empty. Need help getting started? [Generate aggregation](#)'. Below this, there are buttons for 'SAVE', 'CREATE NEW', and 'EXPORT TO LANGUAGE'. The main area shows '78557 Documents in the collection' and a 'Preview of documents' section with three document snippets:

```

_id: ObjectId('59bef0cd2ad8532c2a60093d')
plate: "550/FK213PV"
fuel: 37
vendor: "car2go"
final_time: 1505685847
loc: Object
init_time: 1505685697
vin: "WME4530421Y135192"

_id: ObjectId('59bef0cd2ad8532c2a60093e')
plate: "540/FK820LX"
fuel: 18
vendor: "car2go"
final_time: 1505711577
loc: Object
init_time: 1505685697
vin: "WME4530421Y135142"

_id: ObjectId('59bef1442ad8532c2a60093f')
city: "Torino"
vendor: "enjoy"
final_time: 1505686055
plate: "EZ055TY"
car_category_id: 8
init_time: 1505685824
car_category_type_id: 1

```



ФУНКЦИОНАЛ АГРЕГАЦИЙ

Разберем функционал агрегаций на примере.

Вот агрегация для демонстрации всех особенностей UI. Функции и стадии разберем позже.

Коллекция `ich_edit.movies`

Unset

```
[
  {
    $match:
    {
      year: {
        $gte: 1950,
        $lte: 1970,
      },
    },
  },
  {
    $group:
    {
      _id: "$year",
      total: {
        $sum: 1,
      },
    },
  },
  {
    $sort:
    {
      _id: 1,
    },
  },
]
```



AGGREGATION PIPELINE

Строим агрегацию, добавляя стадии (шаги) агрегации и описываем фильтры для каждого оператора:

Pipeline

\$match
\$group
\$sort

1950-1970 sort ...

SAVE
CREATE NEW
EXPORT TO LANGUAGE

45938 Documents in the collection

Preview of documents

```

_id: ObjectId('573a1390f29313caabcd4132')
title: "Carmencita"
year: 1894
runtime: 1
cast: Array (1)
poster: "https://m.media-
amazon.com/images/M/MV5BMjAzNDEwMzk3O...
plot: "Performing on what looks like a small
wooden stage wearing a dress and "

```

```

_id: ObjectId('573a1390f29313caabcd4132')
title: "Blacksmith"
year: 1893
runtime: 1
released: 1893-04-01
cast: Array (2)
plot: "Three men
bottle of b

```

Stage 1 \$match

A sample of the aggregate

Stage 2 \$group

A sample of the aggregate

Stage 3 \$sort

\$addFields
Adds new field(s) to a document with a cor
reassigns an existing field(s) with a comput

\$bucket
Categorizes incoming documents into gro
based on specified boundaries.

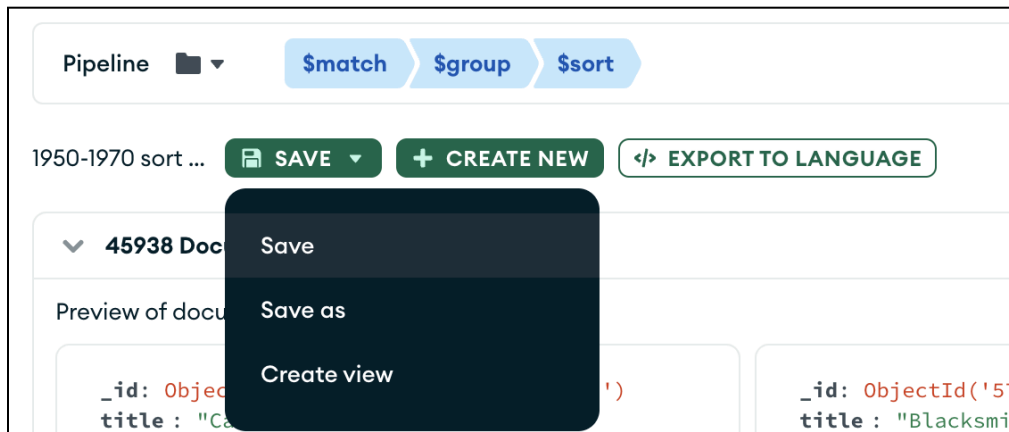
\$bucketAuto
Automatically categorizes documents into
buckets, attempting even distribution if po

\$collStats
Returns statistics regarding a collection or

Returns a count of the number of documen



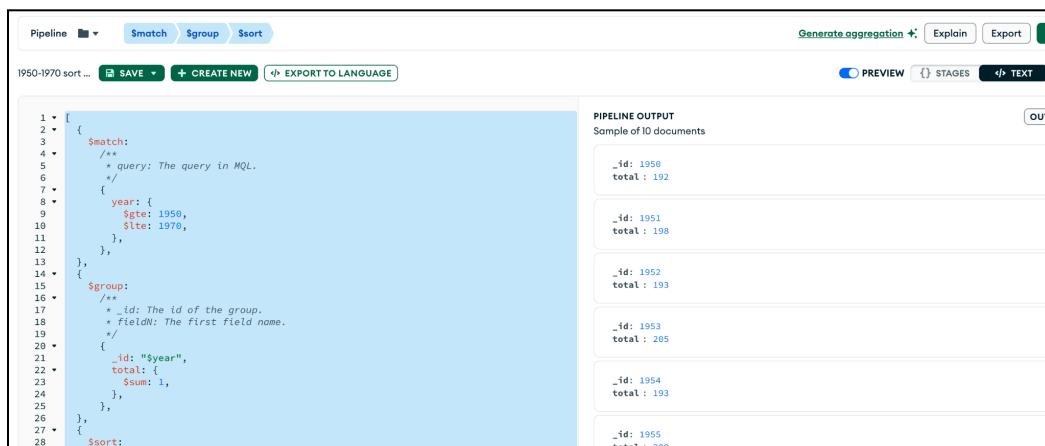
Мы можем сохранить агрегацию в избранных запросах, нажав кнопку save



Переключатель отображения стадий агрегации для удобного отображения стадий и для удобного копирования всей агрегации.



Так, например, все стадии будут описаны в текстовом варианте, нам остается лишь скопировать то, что в [] - квадратных скобках:





ПОПУЛЯРНЫЕ СТАДИИ АГРЕГАЦИИ

\$match - Фильтрация документов на основе заданных критериев. Равнозначен filter из первой вкладки mongo compass

\$group - Группировка документов по определенному полю. Работает с уникальным параметром _id (\$название поля, по которому группируем), Может включать операции, такие как суммирование, подсчет, нахождение среднего и другие.

\$sort - сортирует все документы в указанном порядке. Позволяет упорядочивать результаты в заданном порядке (возрастающем или убывающем) на основе одного или нескольких полей.

\$skip - пропускает несколько результатов

\$limit - Ограничивает количество документов для вывода на количество, переданное методу, начиная, с текущей позиции.

\$project - позволяет показать или скрыть поля в документе, используется для выбора некоторых специальных полей из коллекции.



ФУНКЦИИ, ИСПОЛЬЗУЕМЫЕ ПРИ АГРЕГАЦИИ

\$sum - Суммирует указанные значения всех документов в коллекции

\$avg - Рассчитывает среднее значение указанного поля для всех документов коллекции.

\$max - Получает максимальное значение указанного поля документа в коллекции

\$min - Получает минимальное значение указанного поля документа в коллекции

\$round - Округляет число до целого числа или до указанного десятичного знака.

\$first - Получает первый документ из сгруппированных. Обычно используется вместе с сортировкой.

\$last - Получает крайний документ из сгруппированных. Обычно используется вместе с сортировкой.

И многие другие: [Aggregation Operators — MongoDB Manual](#)



ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Группируем по вендорам коллекцию ich.bookings:

Unset

```
[
  {
    $group: {
      _id: "$vendor",
      total: {
        $sum: 1,
      },
    },
  },
]
```

Pipeline \$group Generate aggregation Explain Export R

Untitled - modified SAVE + CREATE NEW EXPORT TO LANGUAGE PREVIEW STAGES TEXT

78383 Documents in the collection

Preview of documents

<pre>{ "_id": ObjectId("59bef0cd2ad8532c2a600909"), "init_fuel": 96, "city": "Torino", "walking": Object, "init_address": "Via Chambéry 93, 110, 10142 Torino TO", "vendor": "car2go", "final_time": 1505686342 }</pre>	<pre>{ "_id": ObjectId("59bef0cd2ad8532c2a600925"), "init_fuel": 90, "city": "Torino", "walking": Object, "init_address": "Strada Altessano, 140, 10151 Torino TO", "vendor": "car2go", "final_time": 1505686489 }</pre>	<pre>{ "_id": ObjectId("59bef0cd2ad8532c2a60093a"), "init_fuel": 75, "city": "Torino", "walking": Object, "init_address": "Via Guido Reni, 26A, 10136 Torino TO", "vendor": "car2go", "final_time": 1505686390 }</pre>	<pre>{ "_id": ObjectId("59bef12f..."), "init_fuel": 31, "city": "Torino", "walking": Object, "init_address": "Via Silv... Torino TO", "vendor": "car2go", "final_time": 1505688989 }</pre>
---	--	--	--

Stage 1 \$group

1 {
2 _id: "\$vendor",
3 total: {
4 \$sum: 1,
5 },
6 }

Output after \$group stage (Sample of 2 documents)

<pre>{ "_id": "enjoy", "total": 29926 }</pre>	<pre>{ "_id": "car2go", "total": 48457 }</pre>
---	--

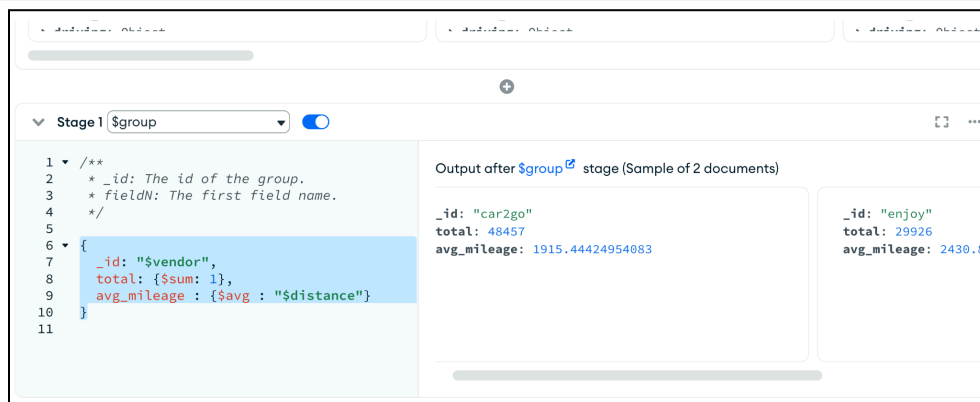
Находим средний показатель километража с разбивкой по вендорам:

Unset

```
[
```



```
{
  $group: {
    _id: "$vendor",
    total: {
      $sum: 1,
    },
    avg_mileage: {
      $avg: "$distance",
    },
  },
},
],
```



The screenshot shows the MongoDB Atlas aggregation pipeline editor. The stage is named "\$group". The pipeline is as follows:

```
1 /**
2  * _id: The id of the group.
3  * fieldN: The first field name.
4  */
5
6 {
7   _id: "$vendor",
8   total: { $sum: 1 },
9   avg_mileage: { $avg: "$distance" }
10 }
11
```

The output after the "\$group" stage (Sample of 2 documents) is shown on the right:

_id	total	avg_mileage
"car2go"	48457	1915.44424954083
"enjoy"	29926	2430.8

2. Найти вендора, у которого средний километраж больше 2000 км.

Unset

```
[
  {
    $group: {
      _id: "$vendor",
      total: {
        $sum: 1,
      },
```



```
    avg_mileage: {
      $avg: "$distance",
    },
  },
},
{
  $match: {
    avg_mileage: {
      $gt: 2000,
    },
  },
},
]
```

Vendor: "car2go"	Vendor: "car2go"	Vendor: "car2go"	Vendor: "car2go"
final_time: 1505686342	final_time: 1505686489	final_time: 1505686390	final_time: 1505688

Stage 1 \$group

```
1 {
2   _id: "$vendor",
3   total: {
4     $sum: 1,
5   },
6   avg_mileage: {
7     $avg: "$distance",
8   },
9 }
```

Output after \$group stage (Sample of 2 documents)

```
_id: "enjoy"
total: 29926
avg_mileage: 2430.8068234979614
```

```
_id: "car2go"
total: 48457
avg_mileage: 1915.44424954083
```


Stage 2 \$match

```
1 {
2   avg_mileage: {
3     $gt: 2000,
4   },
5 }
```

Output after \$match stage (Sample of 1 document)

```
_id: "enjoy"
total: 29926
avg_mileage: 2430.8068234979614
```

3. Коллекция ich.imdb.

Сгруппируем количество фильмов по годам.

Unset

```
[
  {
```

```
$group: {
  _id: "$year",
  total: {
    $sum: 1,
  },
},
},
],
]
```

4. Imdb - сгруппируем количество фильмов по годам между 1980 и 1990. Отсортируем по годам, по возрастанию.

Обращаем внимание, что year в этой коллекции это текст, поэтому мы не можем использовать корректно функции \$gt \$lt и прочие, нам необходимо сравнивать текст:

Unset

```
[
  {
    $match: {
      year: {
        $in: [
          "1980",
          "1981",
          "1982",
          "1983",
          "1984",
          "1985",
          "1986",
          "1987",
          "1988",
          "1989",
        ]
      }
    }
  }
]
```

```

        "1990",
      ],
    },
  },
  {
    $group: {
      _id: "$year",
      total: {
        $sum: 1,
      },
    },
  },
},
{
  $sort:
  /**
   * Provide any number of field/order pairs.
   */
  {
    _id: 1,
  },
},
]

```

5. Imdb - найти 3 самых популярных страны из списка всех фильмов.

Для этого сгруппируем количество фильмов по странам и отсортируем по популярности и ограничим показ 3 результатами:

```

Unset
[
  {
    $group: {
      _id: "$countries",
      total: {
        $sum: 1,
      },
    },
  },
]

```

```
    },  
  },  
  {  
    $sort: {  
      total: -1,  
    },  
  },  
  {  
    $limit:  
      3,  
  },  
]  
]
```