Отчёт по лабораторной работе №7 по

управлению данными

*Гайфиев Иван Алексеевич*

*Булычев Андрей Сергеевич*

*Староверов Роман Александрович*

*Дементьев Леонид Васильевич*

*Николаев Никита Дмитриевич*

*Агафонов Алексей Львович*

*студенты 23КНТ-1*

*НИУ ВШЭ НН*

Формулировки заданий и решения:

## 4. Создать запросы для вывода:

**a) всех различных размеров налогов;**

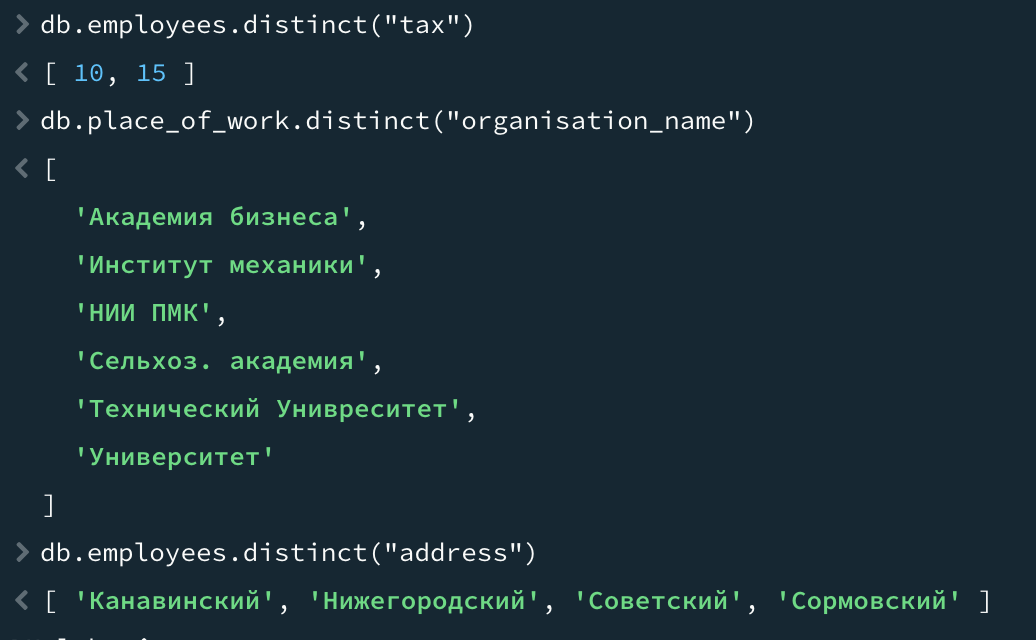
**db.employees.distinct("tax")**

**b) всех различных мест работы;**

**db.place\_of\_work.distinct("organisation\_name")**

**c) всех различных районов проживания сотрудников.**

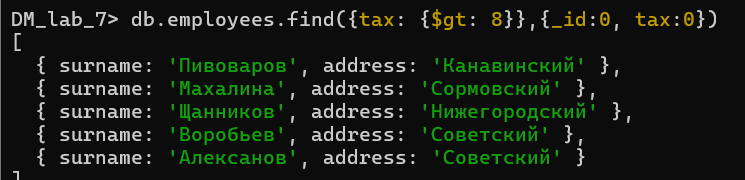
**db.employees.distinct("address")**



## Создав запрос получить следующую информацию:

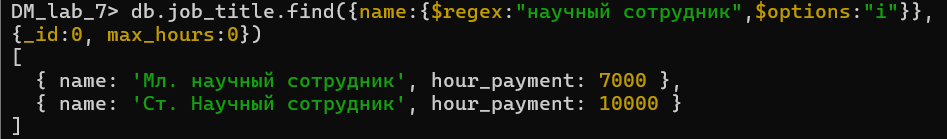
* 1. фамилии и адреса сотрудников, имеющих налог более 8%;

**db.employees.find({tax: {$gt: 8}}, {\_id:0, tax:0})**

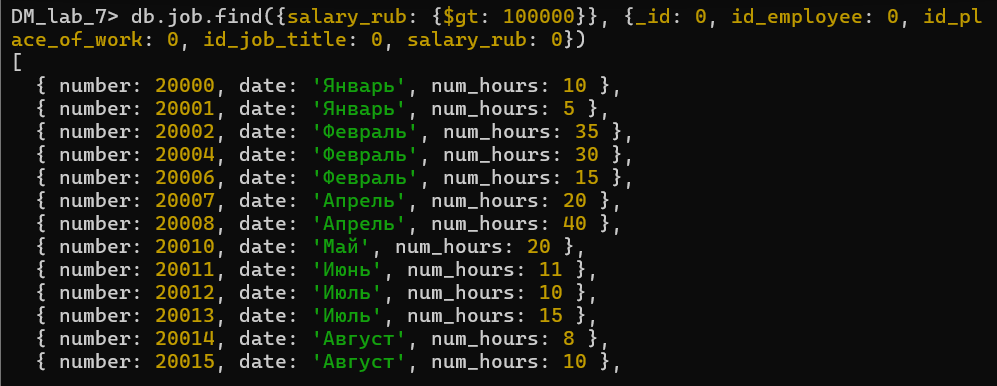


* 1. почасовую оплату и название для должностей, в названии которых встречаются слова “научный сотрудник”;

**db.job\_title.find({name: {$regex: "научный сотрудник", $options: "i"}}, {\_id: 0, max\_hours: 0})**



* 1. номер, дату и количество часов для тех записей о работе, где плата превышала 100000руб.

**db.job.find({salary\_rub: {$gt: 100000}}, {\_id: 0, id\_employee: 0, id\_place\_of\_work: 0, id\_job\_title: 0, salary\_rub: 0})**

## На основании данных о работе вывести все данные в таком формате:

* 1. номер, фамилия сотрудника, дата, количество часов. Отсортировать по количеству часов;

**db.job.aggregate([**

**{**

**$lookup: {**

**from: "employees", localField: "id\_employee", foreignField: "\_id", as: "employee"**

**}**

**},**

**{ $unwind: "$employee" },**

**{**

**$project: {**

**\_id: 0, number: 1, surname: "$employee.surname", date: 1, num\_hours: 1**

**}**

**},**

**{ $sort: { num\_hours: 1 } }**

**])**

**.forEach(doc => {**

**printjson({**

**number: doc.number, surname: doc.surname, date: doc.date, num\_hours: doc.num\_hours })})**

A screen shot of a computer code

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

* 1. название работы, должность, дата, плата.

**db.job.aggregate([**

**{**

**$lookup: {**

**from: "place\_of\_work", localField: "id\_place\_of\_work", foreignField: "\_id", as: "pow"**

**}**

**},**

**{ $unwind: "$pow" },**

**{**

**$lookup: {**

**from: "job\_title", localField: "id\_job\_title", foreignField: "\_id", as: "jt"**

**}**

**},**

**{ $unwind: "$jt" },**

**{**

**$project: {**

**\_id: 0, organisation\_name: "$pow.organisation\_name", job\_title\_name: "$jt.name", date: 1, salary\_rub: 1**

**}**

**}**

**])**

**.forEach(doc => {**

**printjson({**

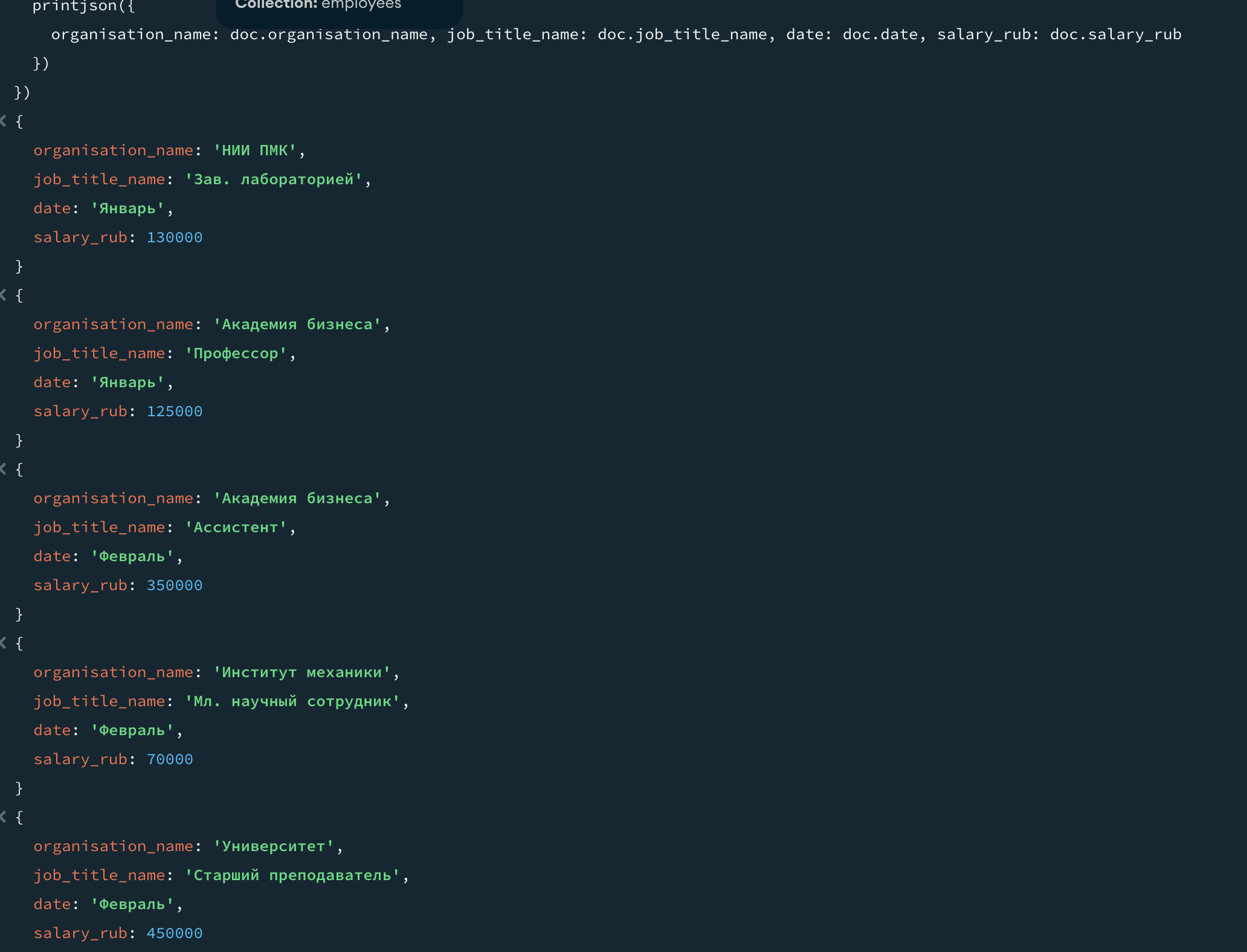
**organisation\_name: doc.organisation\_name, job\_title\_name: doc.job\_title\_name, date: doc.date, salary\_rub: doc.salary\_rub**

**})**

**})**

A computer screen shot of white text

AI-generated content may be incorrect.







A screenshot of a computer program

AI-generated content may be incorrect.

## Вывести:

* 1. названия организаций, где работали доценты или служащие того же района;

**db.job.aggregate([{**

**$lookup: {from: "place\_of\_work",**

**localField: "id\_place\_of\_work",**

**foreignField: "\_id",**

**as: "place\_of\_work\_info"}},**

**{$unwind: "$place\_of\_work\_info"},**

**{$lookup: {from: "employees",**

**localField: "id\_employee",**

**foreignField: "\_id",**

**as: "employee\_info" }},**

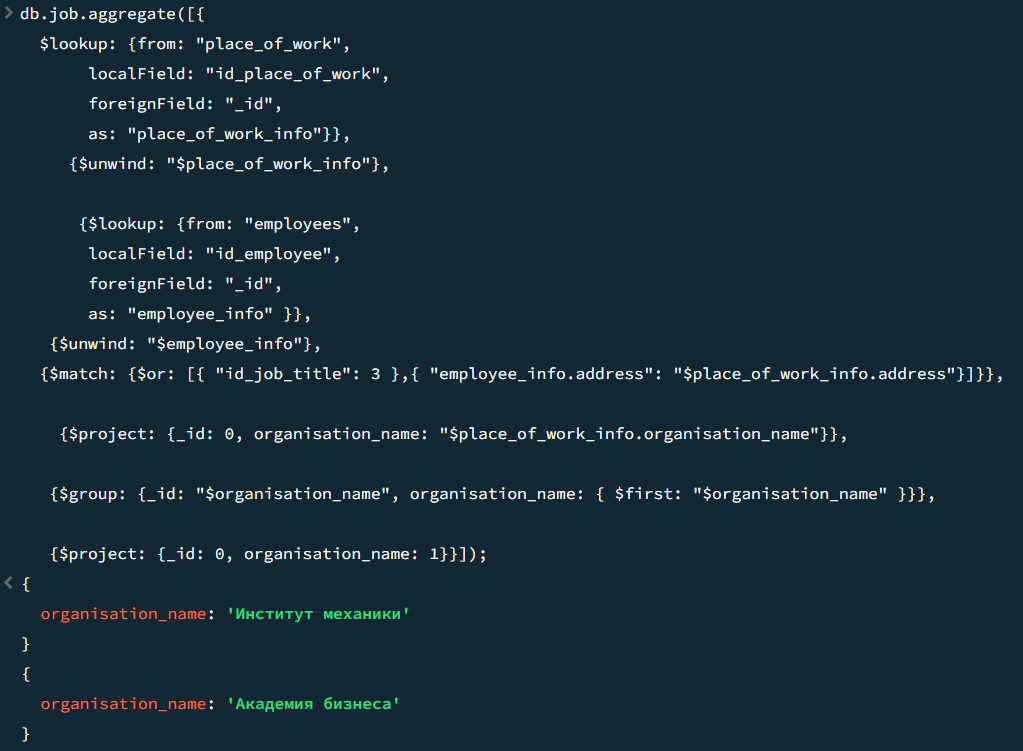
**{$unwind: "$employee\_info"},**

**{$match: {$or: [{ "id\_job\_title": 3 },{ "employee\_info.address": "$place\_of\_work\_info.address"}]}},**

**{$project: {\_id: 0, organisation\_name: "$place\_of\_work\_info.organisation\_name"}},**

**{$group: {\_id: "$organisation\_name", organisation\_name: { $first: "$organisation\_name" }}},**

**{$project: {\_id: 0, organisation\_name: 1}}]);**



* 1. фамилии и размер налога для тех работников, которые имели работу с почасовой оплатой менее 15000руб. не ранее января;

**db.job.aggregate([**

**{$lookup:**

**{from: "employees",**

**localField: "id\_employee",**

**foreignField: "\_id",**

**as: "employees\_info"}},**

**{$unwind: "$employees\_info"},**

**{$lookup:**

**{from: "job\_title",**

**localField: "id\_job\_title",**

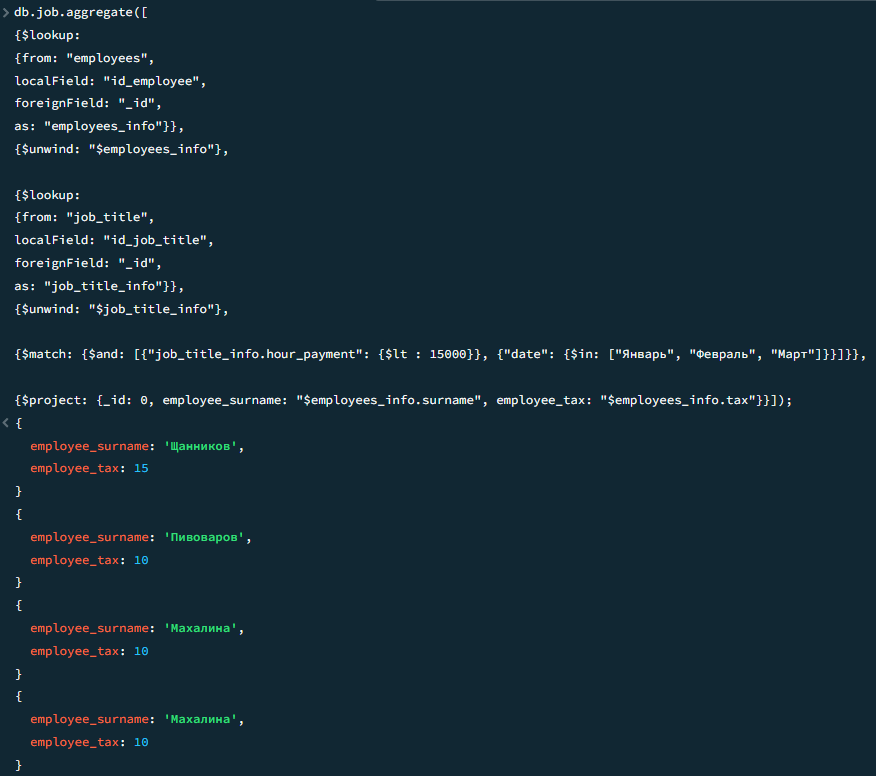
**foreignField: "\_id",**

**as: "job\_title\_info"}},**

**{$unwind: "$job\_title\_info"},**

**{$match: {$and: [{"job\_title\_info.hour\_payment": {$lt : 15000}}, {"date": {$in: ["Январь", "Февраль", "Март"]}}]}},**

**{$project: {\_id: 0, employee\_surname: "$employees\_info.surname", employee\_tax: "$employees\_info.tax"}}]);**

****

* 1. название и размер отчислений для организаций, где работал Александров более одного раза.

**db.job.aggregate([**

**{$lookup:**

**{from: "employees",**

**localField: "id\_employee",**

**foreignField: "\_id",**

**as: "employee\_info"}},**

**{$unwind: "$employee\_info" },**

**{$match: {"employee\_info.surname": "Александров"}},**

**{$lookup:**

**{from: "place\_of\_work",**

**localField: "id\_place\_of\_work",**

**foreignField: "\_id",**

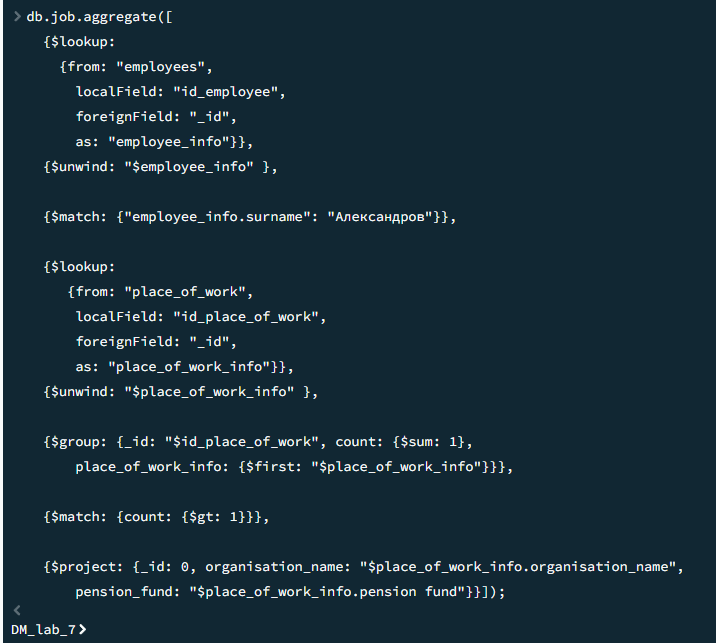
**as: "place\_of\_work\_info"}},**

**{$unwind: "$place\_of\_work\_info" },**

**{$group: {\_id: "$id\_place\_of\_work", count: {$sum: 1},**

**place\_of\_work\_info: {$first: "$place\_of\_work\_info"}}},**

**{$match: {count: {$gt: 1}}},**



* 1. номер работы, название организации, где работали работники из Советского района. Добавить в вывод фамилии таких работников и отсортировать по названию организации.

**db.job.aggregate([**

**{$lookup:**

**{from: "employees",**

**localField: "id\_employee",**

**foreignField: "\_id",**

**as: "employees\_info"}},**

**{$unwind: "$employees\_info"},**

**{$lookup:**

**{from: "place\_of\_work",**

**localField: "id\_place\_of\_work",**

**foreignField: "\_id",**

**as: "place\_of\_work\_info"}},**

**{$unwind: "$place\_of\_work\_info"},**

**{$match: {"employees\_info.address": "Советский"}},**

**{$project: {\_id: 0, number: 1, organisation\_name: "$place\_of\_work\_info.organisation\_name", worker\_surname: "$employees\_info.surname"}},**

**{$sort: {organisation\_name: 1}}]);**



## 10. Используя операцию IN (NOT IN) реализовать следующие запросы:

* 1. найти такие места работы, где не трудились сотрудники из Приокского района;

**db.job.aggregate([**

**{$lookup:**

**{from: "employees",**

**localField: "id\_employee",**

**foreignField: "\_id",**

**as: "employee\_info"}},**

**{$unwind: "$employee\_info"},**

**{$lookup:**

**{from: "place\_of\_work",**

**localField: "id\_place\_of\_work",**

**foreignField: "\_id",**

**as: "place\_info"}},**

**{$unwind: "$place\_info"},**

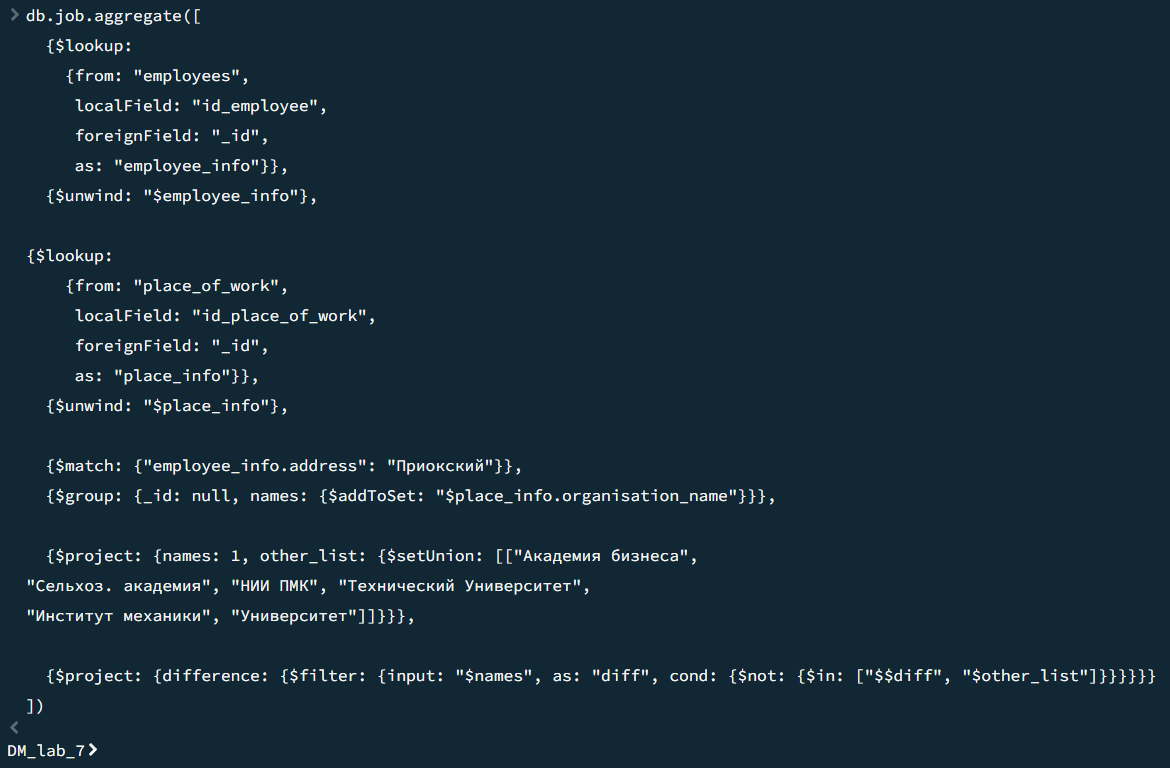
**{$match: {"employee\_info.address": "Приокский"}},**

**{$group: {\_id: null, names: {$addToSet: "$place\_info.organisation\_name"}}},**

**{$project: {names: 1, other\_list: {$setUnion: [["Академия бизнеса",**

**"Сельхоз. академия", "НИИ ПМК", "Технический Университет",**

**"Институт механики", "Университет"]]}}},**

**{$project: {difference: {$filter: {input: "$names", as: "diff", cond: {$not: {$in: ["$$diff", "$other\_list"]}}}}}}])**

* 1. запросы задания 7.а и 7.d;

**db.job.aggregate([{**

**$lookup: {from: "place\_of\_work",**

**localField: "id\_place\_of\_work",**

**foreignField: "\_id",**

**as: "place\_of\_work\_info"}},**

**{$unwind: "$place\_of\_work\_info"},**

**{$lookup: {from: "employees",**

**localField: "id\_employee",**

**foreignField: "\_id",**

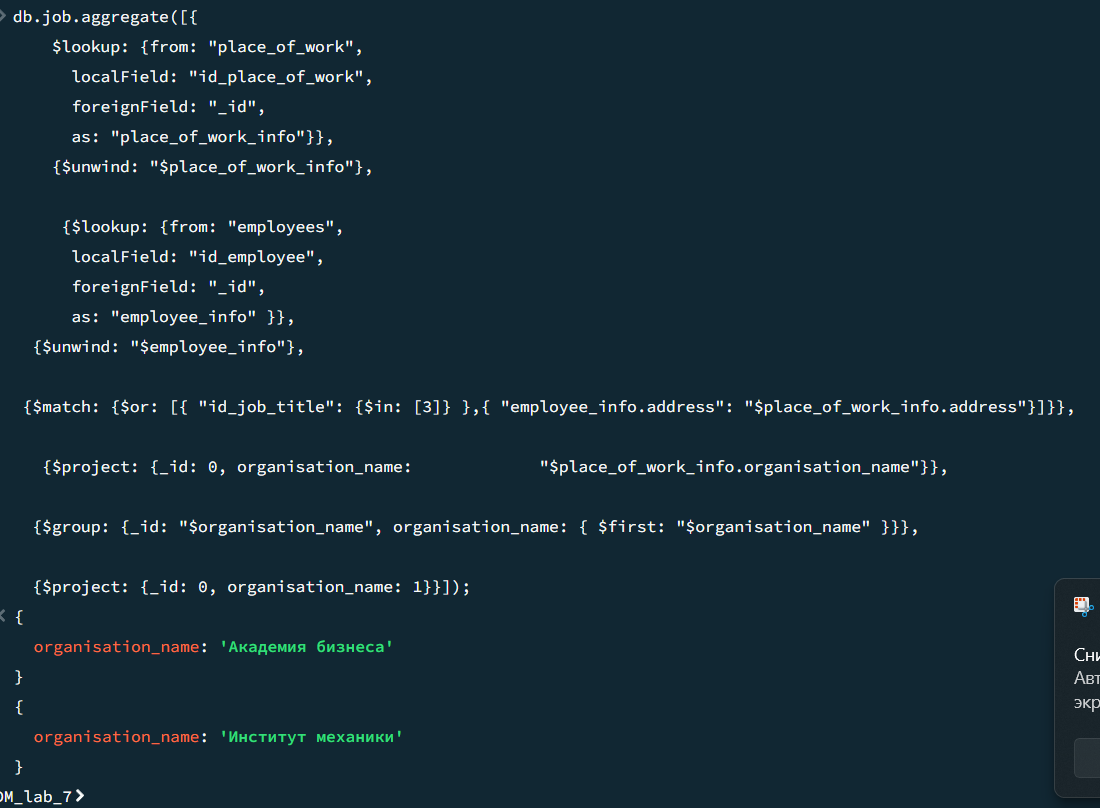
**as: "employee\_info" }},**

**{$unwind: "$employee\_info"},**

**{$match: {$or: [{ "id\_job\_title": {$in: [3]} },{ "employee\_info.address": "$place\_of\_work\_info.address"}]}},**

**{$project: {\_id: 0, organisation\_name: "$place\_of\_work\_info.organisation\_name"}},**

**{$group: {\_id: "$organisation\_name", organisation\_name: { $first: "$organisation\_name" }}},**

**{$project: {\_id: 0, organisation\_name: 1}}]);**  
  


**db.job.aggregate([**

**{$lookup:**

**{from: "employees",**

**localField: "id\_employee",**

**foreignField: "\_id",**

**as: "employees\_info"}},**

**{$unwind: "$employees\_info"},**

**{$lookup:**

**{from: "place\_of\_work",**

**localField: "id\_place\_of\_work",**

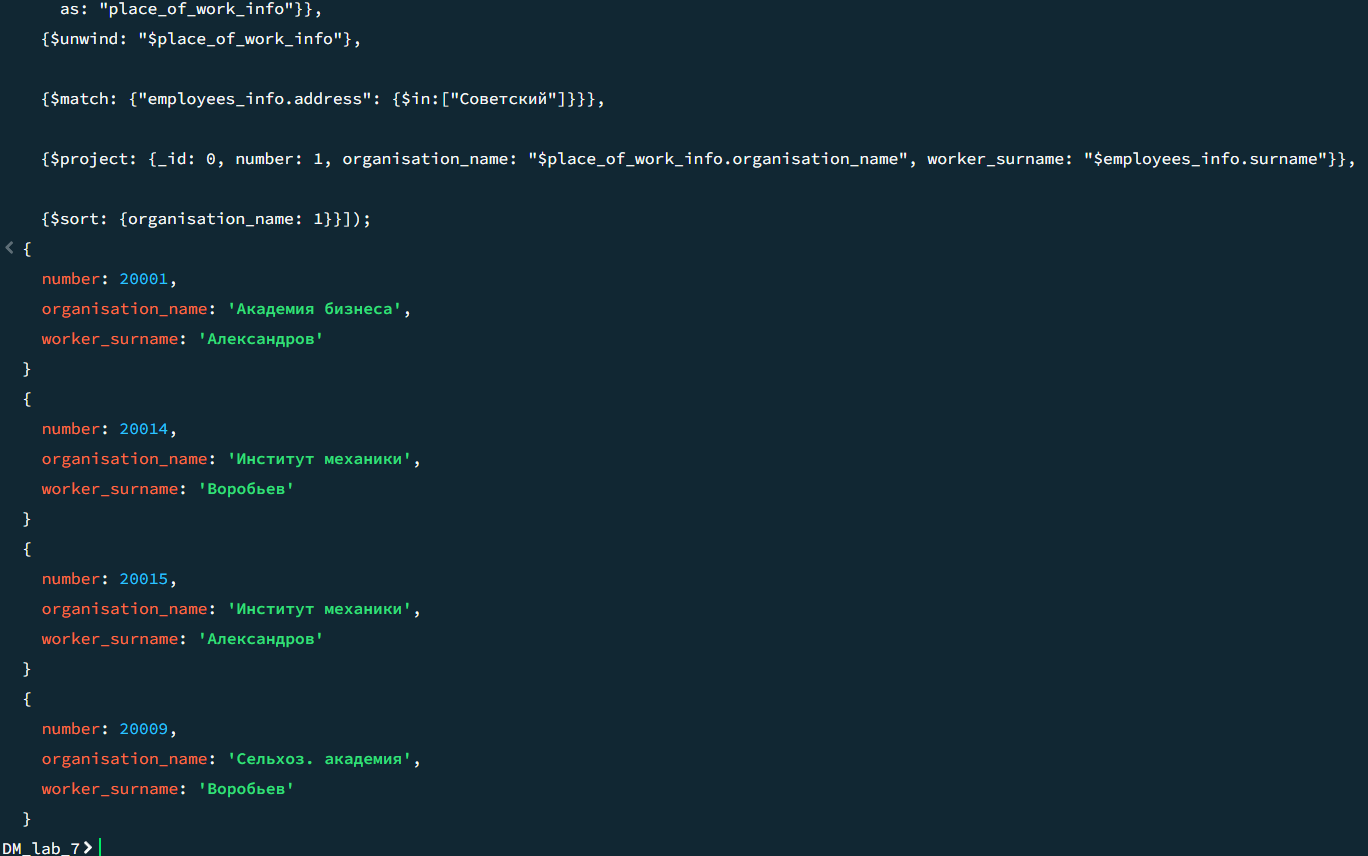
**foreignField: "\_id",**

**as: "place\_of\_work\_info"}},**

**{$unwind: "$place\_of\_work\_info"},**

**{$match: {"employees\_info.address": {$in:["Советский"]}}},**

**{$project: {\_id: 0, number: 1, organisation\_name: "$place\_of****\_work\_info.organisation\_name", worker\_surname: "$employees\_info.surname"}},**

**{$sort: {organisation\_name: 1}}]);db.place\_of\_work.d**  
  


* 1. определить должности, на которых работал Пивоваров более раза.

**db.job.aggregate([**

**{$lookup:**

**{from: "employees",**

**localField: "id\_employee",**

**foreignField: "\_id",**

**as: "employees\_info"}},**

**{$unwind: "$employees\_info"},**

**{$lookup:**

**{from: "job\_title",**

**localField: "id\_job\_title",**

**foreignField: "\_id",**

**as: "title\_info"}},**

**{$unwind: "$title\_info"},**

**{$match: {"employees\_info.surname": {$in: ["Пивоваров"]}}},**

**{$group: {\_id: "$title\_info.name", count: {$sum: 1}}},**

**{$match: {count: {$gt: 1}}},**

**{$project: {\_id: 0, title\_name: "$\_id"}}]);**



## 11. Используя операции ALL-ANY реализовать следующие запросы:

* 1. найти должность с самой высокой почасовой оплатой;

**var maxPayment = db.job\_title.aggregate([**

**{ $group: { \_id: null, maxPayment: { $max: "$hour\_payment" } } }**

**]).toArray()[0].maxPayment;**

**db.job\_title.find({ hour\_payment: maxPayment }).forEach(doc => printjson(doc));**

A screen shot of a computer code

AI-generated content may be incorrect.

* 1. найти место работы, где на самой низкооплачиваемой должности работал Воробьев;

**db.job.aggregate([**

**{ $match: { id\_employee: 4 } },**

**{ $lookup: {**

**from: "job\_title", localField: "id\_job\_title", foreignField: "\_id", as: "jt"**

**} },**

**{ $unwind: "$jt" },**

**{ $sort: { "jt.hour\_payment": 1 } },**

**{ $limit: 1 },**

**{ $lookup: {**

**from: "place\_of\_work", localField: "id\_place\_of\_work", foreignField: "\_id", as: "pow"**

**} },**

**{ $unwind: "$pow" },**

**{ $project: {**

**\_id: 0, organisation\_name: "$pow.organisation\_name", address: "$pow.address" }}]).forEach(doc => printjson(doc));**

A computer screen with white text

AI-generated content may be incorrect.

* 1. среди работавших в университете найти сотрудника с самым низким налогом;

**db.job.aggregate([**

**{ $lookup: {**

**from: "place\_of\_work", localField: "id\_place\_of\_work", foreignField: "\_id", as: "pow"**

**} },**

**{ $unwind: "$pow" },**

**{ $match: { "pow.organisation\_name": "Университет" } },**

**{ $lookup: {**

**from: "employees", localField: "id\_employee", foreignField: "\_id", as: "emp"**

**} },**

**{ $unwind: "$emp" },**

**{ $group: {**

**\_id: "$emp.\_id", surname: { $first: "$emp.surname" }, tax: { $first: "$emp.tax" }**

**} },**

**{ $sort: { tax: 1 } },**

**{ $limit: 1 }**

**])**

**.forEach(doc => printjson(doc));**

A computer screen shot of a program code

AI-generated content may be incorrect.

* 1. название и размер отчислений для организаций, где работал Александров более одного раза.

**var emp = db.employees.findOne({ surname: "Алексанов" });**

**if (emp) {**

**db.job.aggregate([**

**{ $match: { id\_employee: emp.\_id } },**

**{ $group: {**

**\_id: "$id\_place\_of\_work", jobCount: { $sum: 1 }**

**} },**

**{ $match: { jobCount: { $gt: 1 } } },**

**{ $lookup: {**

**from: "place\_of\_work", localField: "\_id", foreignField: "\_id", as: "pow"**

**} },**

**{ $unwind: "$pow" },**

**{ $project: {**

**\_id: 0, organisation\_name: "$pow.organisation\_name", pension\_fund: "$pow['pension fund']"} } ]) .forEach(doc => printjson(doc));}**

A computer screen shot of white text

AI-generated content may be incorrect.

## 12. Используя операцию UNION получить адреса проживания сотрудников и места расположения организаций.

**db.employees.aggregate([**

**{**

**$project: {**

**\_id: 0,**

**address: 1,**

**belongsTo: "$surname",**

**type: { $literal: "employee" } } },**

**{**

**$unionWith: {**

**coll: "place\_of\_work",**

**pipeline: [**

**{**

**$project: {**

**\_id: 0,**

**address: 1,**

**belongsTo: "$organisation\_name",**

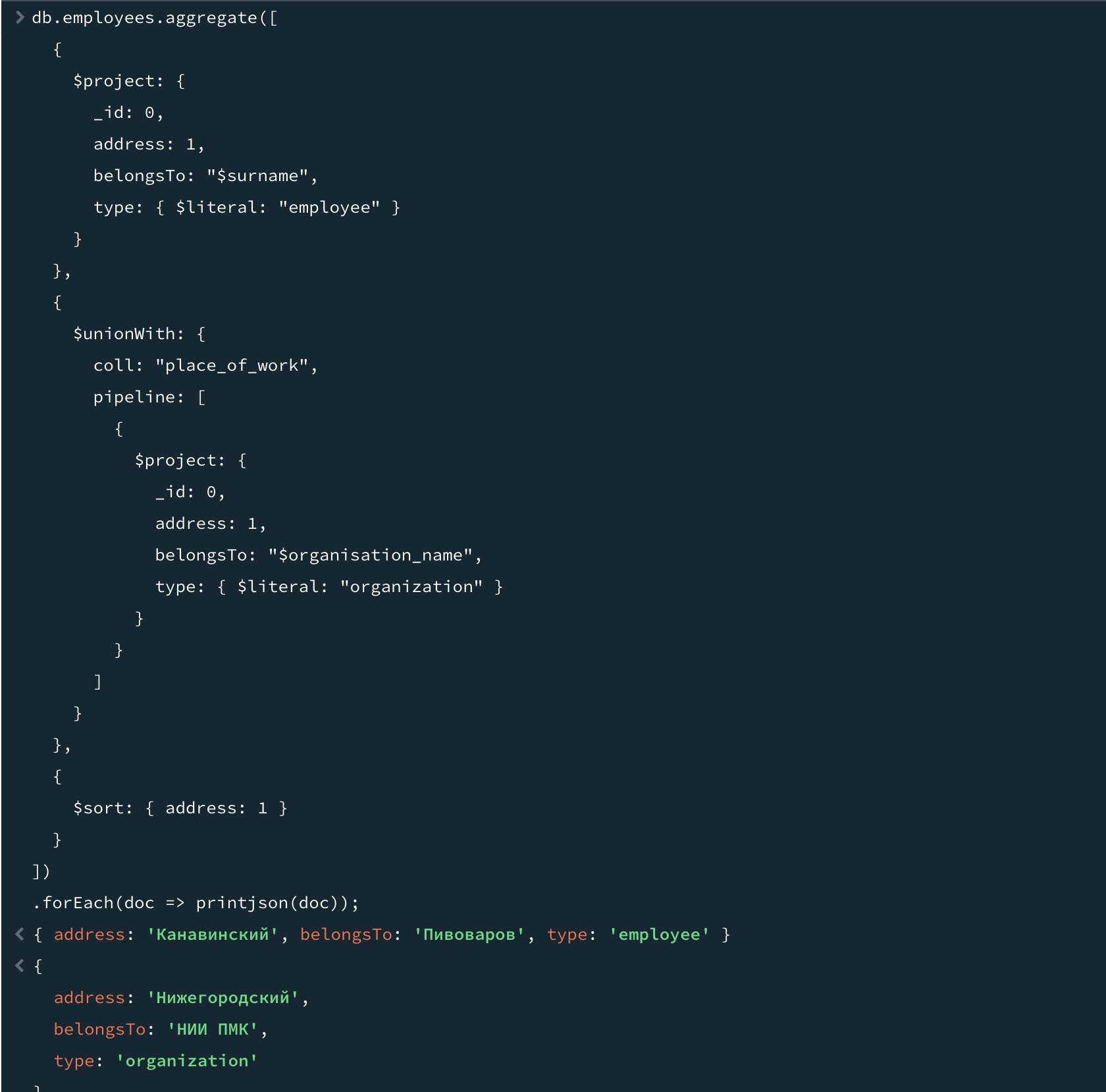
**type: { $literal: "organization" }}}] }, {**

**$sort: { address: 1 }**

**}**

**])**

**.forEach(doc => printjson(doc));**



A screen shot of a computer program

AI-generated content may be incorrect.

## 13. Используя операцию EXISTS ( NOT EXISTS ) реализовать нижеследующие запросы. В случае, если для текущего состояния БД запрос будет выдавать пустое множество строк, требуется указать какие добавления в БД необходимо провести.

a. найти сотрудников, работавших на должности доцента во всех организациях;  
**var totalOrgs = db.place\_of\_work.countDocuments({});**

**db.employees.aggregate([**

**{**

**$lookup: {**

**from: "job",**

**let: { empId: "$\_id" },**

**pipeline: [**

**{**

**$match: { $expr: { $and: [**

**{ $eq: ["$id\_employee", "$$empId"] }, { $eq: ["$id\_job\_title", 3] }]} } },**

**{ $group: {**

**\_id: "$id\_place\_of\_work" } } ],**

**as: "dozentOrgs" } },**

**{ $addFields: {**

**dozentCount: { $size: "$dozentOrgs" } } },**

**{**

**$match: { dozentCount: totalOrgs } },**

**{ $project: { \_id: 0, surname: 1, dozentCount: 1 } }]).forEach(doc => printjson(doc));**A screenshot of a computer code

AI-generated content may be incorrect.b. найти места работы на которых работали все сотрудники из Приокского или Сормовского районов;

**db.place\_of\_work.aggregate([**

**{**

**$lookup: {**

**from: "employees",**

**let: { placeId: "$\_id" },**

**pipeline: [**

**{**

**$match: {**

**$or: [**

**{ address: { $regex: /Приокский/i } }, { address: { $regex: /Сормовский/i } }]}**

**},**

**{ $project: { \_id: 1 } }**

**],**

**as: "allPotentialEmp"**

**}**

**},**

**{**

**$lookup: {**

**from: "job", localField: "\_id", foreignField: "id\_place\_of\_work", as: "jobs"**

**}**

**},**

**{**

**$addFields: {**

**potentialEmp: {**

**$map: {**

**input: "$allPotentialEmp.\_id", as: "peId", in: { $toString: "$$peId" }**

**}**

**},**

**actualEmp: {**

**$map: {**

**input: "$jobs.id\_employee", as: "aeId", in: { $toString: "$$aeId" }**

**}}}**

**},**

**{**

**$match: {**

**$expr: {**

**$setIsSubset: [**

**"$potentialEmp",**

**"$actualEmp"**

**]**

**}**

**}**

**},**

**{**

**$project: {**

**\_id: 0, organisation\_name: 1, address: 1**

**}**

**}**

**])**

**.forEach(doc => printjson(doc));**A screen shot of a computer program

AI-generated content may be incorrect.

c. определить должности на которых не работали сотрудники в организациях чужих районов;

**db.job.aggregate([**

**{**

**$lookup: {**

**from: "employees", localField: "id\_employee", foreignField: "\_id", as: "emp"**

**}**

**},**

**{ $unwind: "$emp" },**

**{**

**$lookup: {**

**from: "place\_of\_work", localField: "id\_place\_of\_work", foreignField: "\_id", as: "pow"**

**}**

**},**

**{ $unwind: "$pow" },**

**{**

**$project: {**

**id\_job\_title: 1, diff: { $ne: ["$emp.address", "$pow.address"] } } },**

**{ $group: {**

**\_id: "$id\_job\_title", allDiff: { $min: "$diff" }**

**}},**

**{ $match: { allDiff: true } },**

**{ $lookup: { from: "job\_title", localField: "\_id", foreignField: "\_id", as: "jt"} },**

**{ $unwind: "$jt" },**

**{ $project: {**

**\_id: 0,**

**job\_title: "$jt.name"**

**}}**

**]).forEach(doc => printjson(doc));**A computer screen shot of a computer code

AI-generated content may be incorrect.

d. найти сотрудника и должность, в которой этот сотрудник проработал во всех организациях Нижегородского или Сормовского районов.

**{ $addFields: { id\_place\_of\_work\_str: { $toString: "$id\_place\_of\_work" } } },**

**{ $match: { id\_place\_of\_work\_str: { $in: targetOrgs } } },**

**{**

**$group: {**

**\_id: { employee: "$id\_employee", jobTitle: "$id\_job\_title" },**

**orgs: { $addToSet: "$id\_place\_of\_work\_str" } }},**

**{ $match: {**

**$expr: { $eq: [ { $size: "$orgs" }, totalTargetOrgs }},**

**{ $lookup: { from: "employees", localField: "\_id.employee", foreignField: "\_id", as: "emp" } },**

**{ $unwind: "$emp" },**

**{ $lookup: {**

**from: "job\_title", localField: "\_id.jobTitle", foreignField: "\_id", as: "jt" } },**

**{ $unwind: "$jt" },**

**{ $project: {**

**\_id: 0, surname: "$emp.surname", job\_title: $jt.name" }}**

**]).forEach(doc => printjson(doc));**

A screenshot of a computer program

AI-generated content may be incorrect.

## 14. Реализовать запросы с использованием аггрегатных функций:

а) найти число различных работников, трудившихся в должности ассистента до ноября;

**db.job.aggregate([**

**{$match: {**

**date: {**

**$in: [**

**"Январь", "Февраль", "Март", "Апрель",**

**"Май", "Июнь", "Июль", "Август",**

**"Сентябрь", "Октябрь" ]}}},**

**{$lookup: {**

**from: "job\_title", localField: "id\_job\_title", foreignField: "\_id", as: "jt"}},**

**{ $unwind: "$jt" },**

**{$match: {"jt.name": "Ассистент"}},**

**{$group: {\_id: "$id\_employee"}},**

**{ $count: "count\_of\_assistants\_before\_november"}]);**

A screenshot of a computer code

AI-generated content may be incorrect.

1. определить среднюю величину отчислений для тех организаций, где трудились сотрудники с налогом менее 15%;

**db.job.aggregate([**

**{**

**$lookup: {**

**from: "employees", localField: "id\_employee", foreignField: "\_id", as: "emp"**

**}**

**},**

**{ $unwind: "$emp" },**

**{**

**$match: {**

**"emp.tax": { $lt: 15 }**

**}**

**},**

**{**

**$lookup: {**

**from: "place\_of\_work", localField: "id\_place\_of\_work", foreignField: "\_id", as: "pow"**

**}**

**},**

**{ $unwind: "$pow" },**

**{**

**$group: {**

**\_id: { placeOfWorkId: "$pow.\_id", organisation\_name: "$pow.organisation\_name"**

**},**

**average: {**

**$avg: {**

**$multiply: [ "$salary\_rub",{ $divide: ["$pow.pension fund", 100] }]**

**} } }},**

**{**

**$project: {**

**\_id: 0,**

**organisation\_name: "$\_id.organisation\_name",**

**average: 1**

**}**

**}])**

**.forEach(doc => printjson(doc));**

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a code

AI-generated content may be incorrect.

1. найти число сотрудников, работавших до марта в университете;

**db.job.aggregate([**

**{**

**$match: {**

**date: { $in: ["Январь", "Февраль"] }**

**}**

**},**

**{**

**$lookup: {**

**from: "place\_of\_work",**

**localField: "id\_place\_of\_work",**

**foreignField: "\_id",**

**as: "pow"**

**}**

**},**

**{ $unwind: "$pow" },**

**{**

**$match: {**

**"pow.organisation\_name": "Университет" } },**

**{ $group: {**

**\_id: "$id\_employee" } },**

**{ $count: "count\_of\_employees\_in\_university\_before\_march" }])**

**.forEach(doc => printjson(doc));**

A screen shot of a computer program

AI-generated content may be incorrect.

1. какие работники получали плату выше средней.

**db.job.aggregate([**

**{**

**$lookup: {**

**from: "employees", localField: "id\_employee", foreignField: "\_id", as: "emp"**

**}**

**},**

**{ $unwind: "$emp" },**

**{**

**$set: {**

**netPay: {**

**$multiply: [**

**"$salary\_rub", {$subtract: [1, { $divide: ["$emp.tax", 100] }] }]}**

**}**

**},**

**{**

**$setWindowFields: {**

**partitionBy: null,**

**output: {**

**avgNetPay: { $avg: "$netPay" } } }**

**},**

**{**

**$match: {**

**$expr: { $gt: ["$netPay", "$avgNetPay"] }**

**}**

**},**

**{**

**$group: {**

**\_id: "$emp.\_id", surname: { $first: "$emp.surname" }, netPayAboveAvg: { $addToSet: "$netPay" }**

**} },**

**{**

**$project: {**

**\_id: 0, surname: 1, netPayAboveAvg: 1 } }])**

**.forEach(doc => printjson(doc));**

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a code

AI-generated content may be incorrect.

## 15. Используя средства группировки реализовать следующие запросы:

* 1. найти должности, на которых работало более трех человек;

**db.job.aggregate([**

**{$lookup:**

**{from: "job\_title",**

**localField: "id\_job\_title",**

**foreignField: "\_id",**

**as: "title\_info"}},**

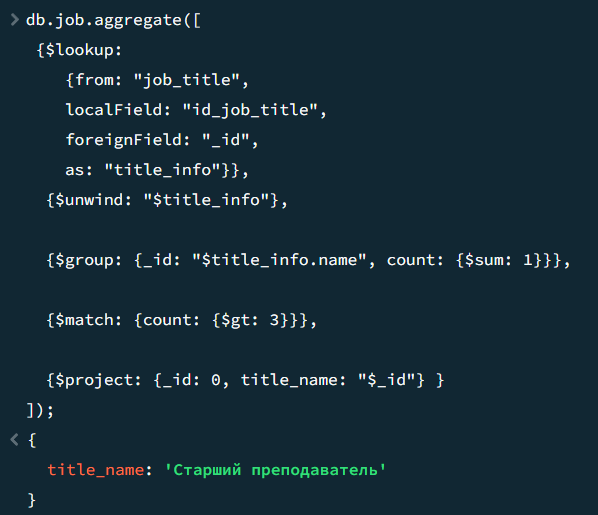
**{$unwind: "$title\_info"},**

**{$group: {\_id: "$title\_info.name", count: {$sum: 1}}},**

**{$match: {count: {$gt: 3}}},**

**{$project: {\_id: 0, title\_name: "$\_id"} }**

**]);**



* 1. найти для каждого сотрудника максимальную получаемую плату;

**db.job.aggregate([**

**{$lookup:**

**{from: "employees",**

**localField: "id\_employee",**

**foreignField: "\_id",**

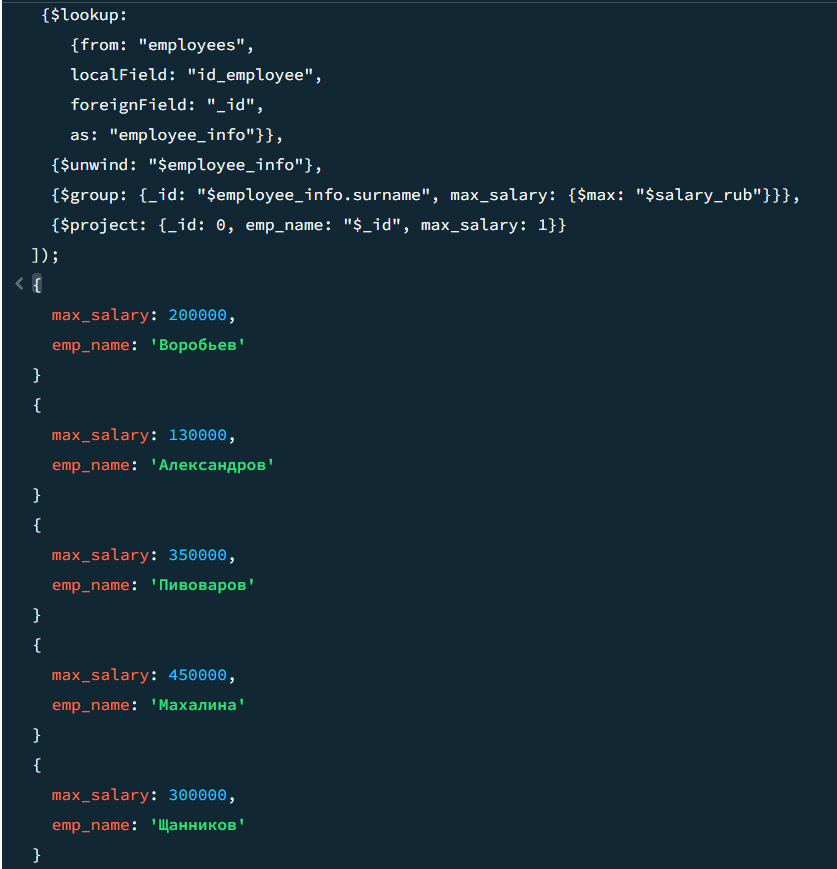
**as: "employee\_info"}},**

**{$unwind: "$employee\_info"},**

**{$group: {\_id: "$employee\_info.surname", max\_salary: {$max: "$salary\_rub"}}},**

**{$project: {\_id: 0, emp\_name: "$\_id", max\_salary: 1}}**

**]);**



* 1. какие организации за месяц тратили на зарплату более 1000000;

**db.job.aggregate([**

**{$lookup:**

**{from: "place\_of\_work",**

**localField: "id\_place\_of\_work",**

**foreignField: "\_id",**

**as: "place\_info"}},**

**{$unwind: "$place\_info"},**

**{$group: {\_id: {month: "$date", place\_name: "$place\_info.name"}, sum\_salary: {$sum: "$salary\_rub"}}},**

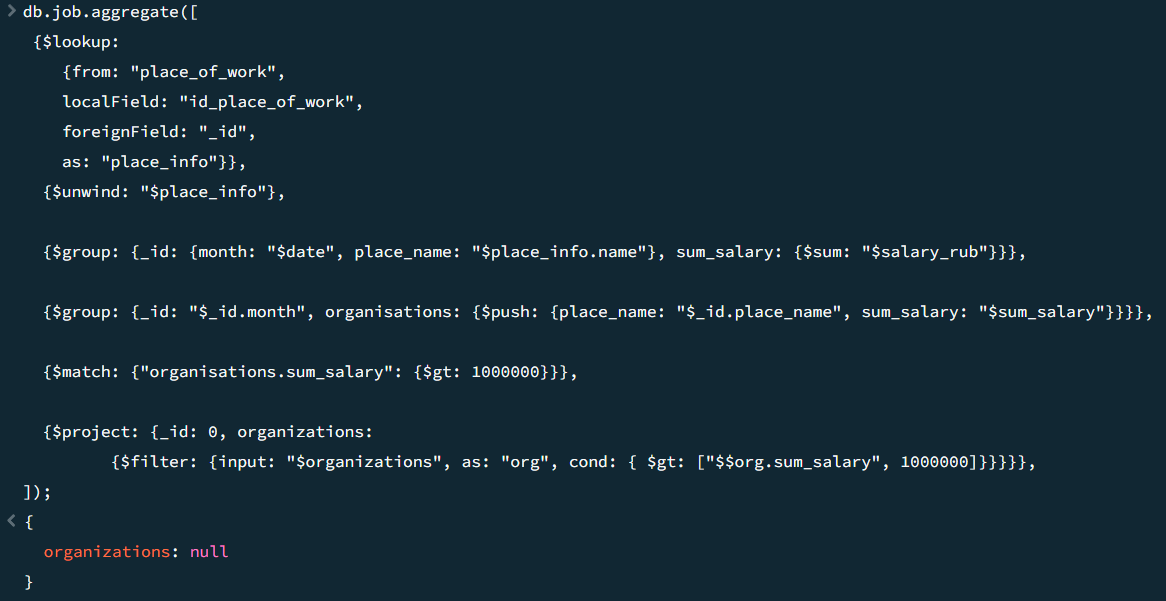
**{$group: {\_id: "$\_id.month", organisations: {$push: {place\_name: "$\_id.place\_name", sum\_salary: "$sum\_salary"}}}},**

**{$match: {"organisations.sum\_salary": {$gt: 1000000}}},**

**{$project: {\_id: 0, organizations:**

**{$filter: {input: "$organizations", as: "org", cond: { $gt: ["$$org.sum\_salary", 1000000]}}}}},**

**]);**



* 1. получить для каждой организации из Нижегородского или Сормовского района среднее число часов, которое нарабатывали сотрудники.

**db.job.aggregate([**

**{$lookup:**

**{from: "place\_of\_work",**

**localField: "id\_place\_of\_work",**

**foreignField: "\_id",**

**as: "place\_info"}},**

**{$unwind: "$place\_info"},**

**{$match: {"place\_info.address" : {$in: ["Нижегородский", "Сормовский"]}}},**

**{$group: {\_id: "$place\_info.organisation\_name", average\_hours: {$avg: "$num\_hours"}}},**

**{$project: {\_id: 0, organisation\_name: "$\_id", average\_hours: 1}}])**