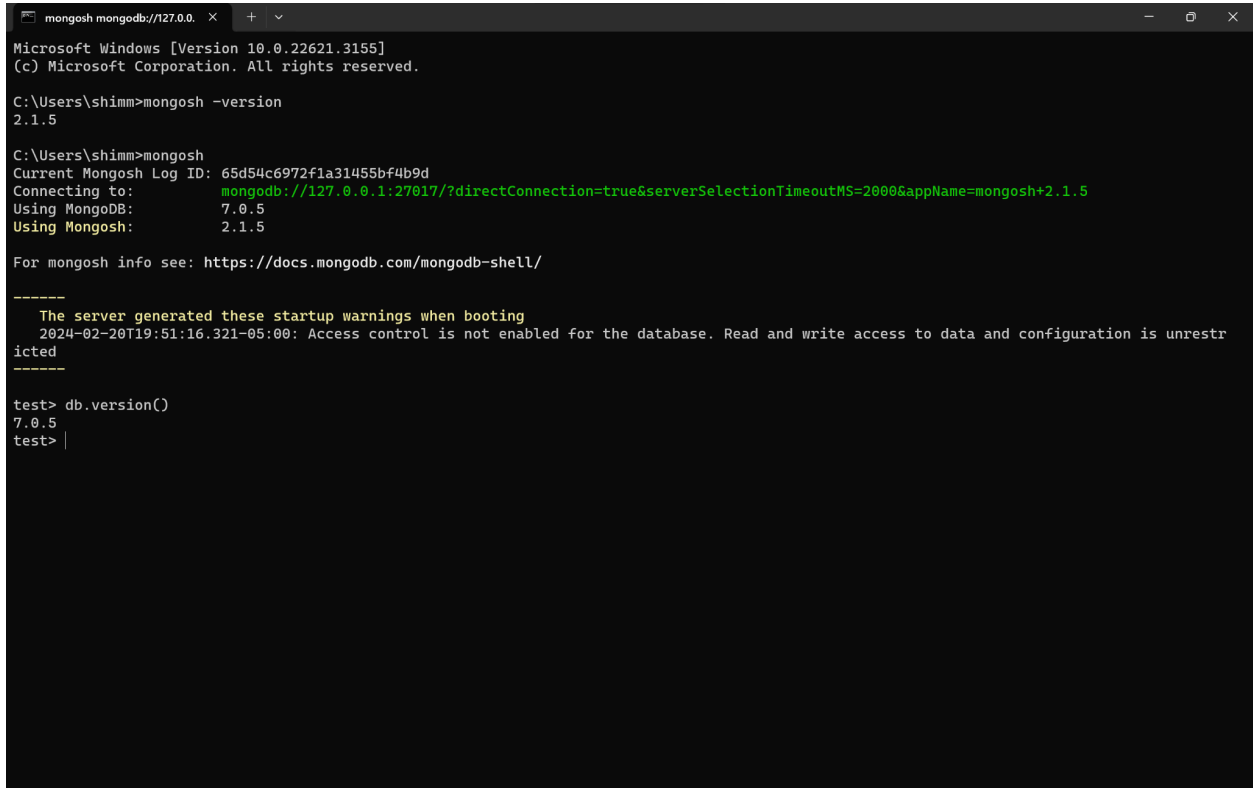


Mongo1 Writeup

Really, this assignment was much easier than you made it imply. MQL has more brackets than are strictly-speaking necessary (since queries are JSON objects), but it's not a bad language.

Screenshot with versions:



```
mongosh mongodb://127.0.0.1:27017/
Microsoft Windows [Version 10.0.22621.3155]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shimm>mongosh -version
2.1.5

C:\Users\shimm>mongosh
Current Mongosh Log ID: 65d54c6972f1a31455bf4b9d
Connecting to:  mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.5
Using MongoDB:  7.0.5
Using Mongosh:  2.1.5

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
The server generated these startup warnings when booting
2024-02-20T19:51:16.321-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestr
icted
-----

test> db.version()
7.0.5
test> |
```

Query 1: Return the document with “worker name” Kimiko

```
db.employees.find({name: "Kimiko"}).pretty()
```

```
mongosh mongodb://127.0.0.1:27017/employees
{
  name: 'Hameke',
  street: 'Park',
  city: 'New York',
  manages: [],
  companies: []
},
{
  _id: ObjectId('65d559716b5687b7a53caf44'),
  name: 'Kimiko',
  street: 'Yonge',
  city: 'Toronto',
  manages: [],
  companies: [
    { name: 'Sony', city: 'Culver City', salary: 89925 },
    { name: 'Sony', city: 'Tokyo', salary: 118000 }
  ]
},
{
  _id: ObjectId('65d559716b5687b7a53caf45'),
  name: 'Olivia',
  street: 'Xidan',
  city: 'Shengzhou',
  manages: [],
  companies: [ { name: 'Criteo', city: 'Dubai', salary: 400000 } ]
}
]
nano_employees> db.employees.find({name: "Kimiko"}).pretty()
[
  {
    _id: ObjectId('65d559716b5687b7a53caf44'),
    name: 'Kimiko',
    street: 'Yonge',
    city: 'Toronto',
    manages: [],
    companies: [
      { name: 'Sony', city: 'Culver City', salary: 89925 },
      { name: 'Sony', city: 'Tokyo', salary: 118000 }
    ]
  }
]
nano_employees> |
```

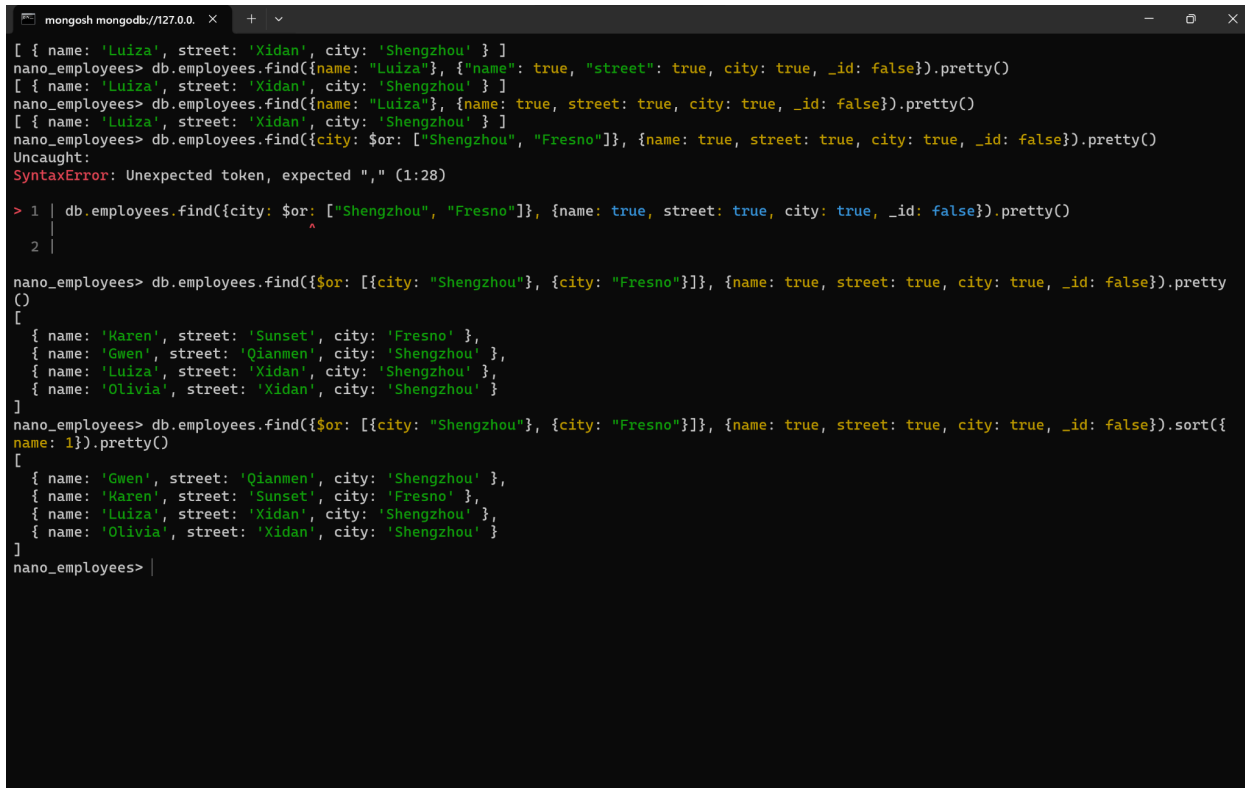
Query 2: Return the document with “worker name” Luiza.

```
db.employees.find({name: "Luiza"}).pretty()
```

```
mongosh mongodb://127.0.0.1:27000 > use nano
nano_employees> db.employees.find({name: "Kimiko"}).pretty()
[
  {
    _id: ObjectId('65d559716b5687b7a53caf44'),
    name: 'Kimiko',
    street: 'Yonge',
    city: 'Toronto',
    manages: [],
    companies: [
      { name: 'Sony', city: 'Culver City', salary: 89925 },
      { name: 'Sony', city: 'Tokyo', salary: 118000 }
    ]
  }
]
nano_employees> db.employees.find({name: "Luiza"}).pretty()
[
  {
    _id: ObjectId('65d559716b5687b7a53caf42'),
    name: 'Luiza',
    street: 'Xidan',
    city: 'Shengzhou',
    manages: [
      ObjectId('65d559716b5687b7a53caf38'),
      ObjectId('65d559716b5687b7a53caf45')
    ],
    companies: [ { name: 'Criteo', city: 'Dubai', salary: 310000 } ]
  }
]
nano_employees> |
```

Query 3: Return all workers living either in Shengzhou or Fresno. Each document in the result must include only the worker name, street, and city properties. The result must be sorted by increasing “name”

```
db.employees.find({$or: [{city: "Shengzhou"}, {city: "Fresno"}]}],
    {name: true, street: true, city: true, _id: false})
.sort({name: 1}).pretty()
```

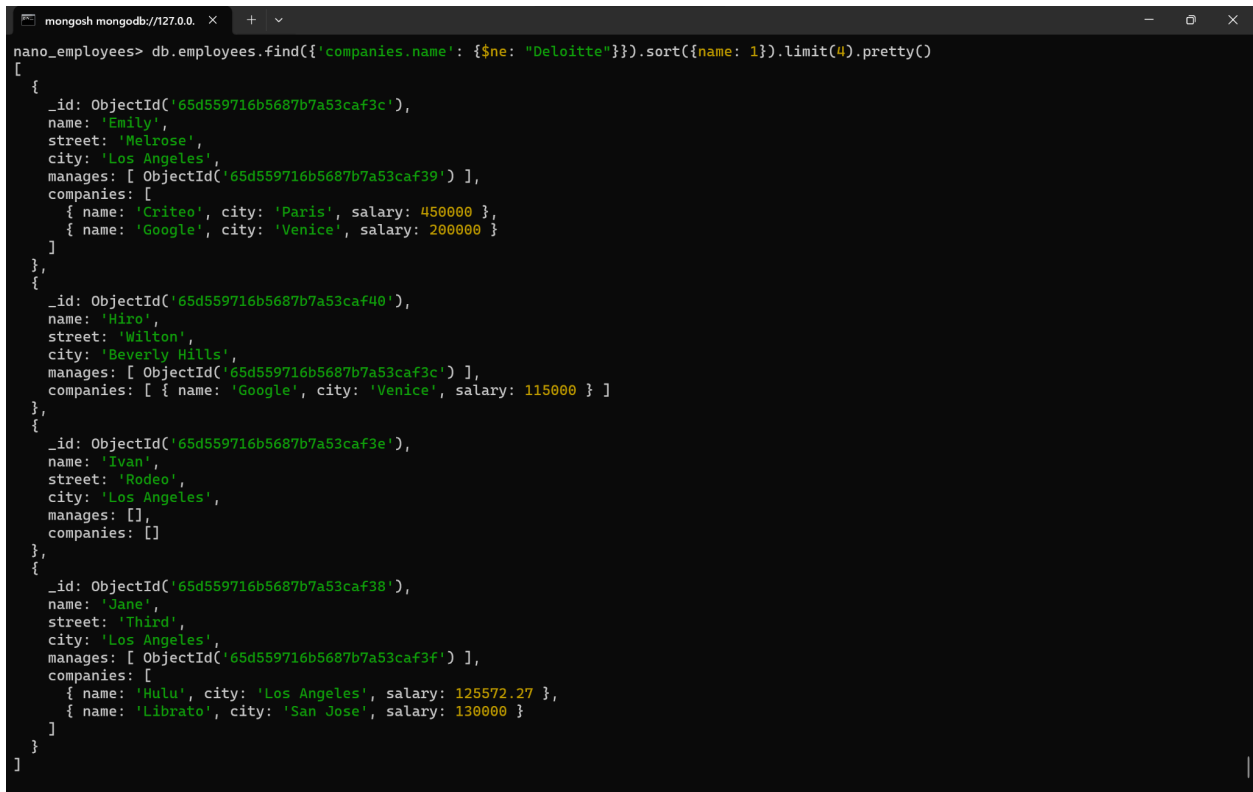


The screenshot shows a MongoDB shell window with the following content:

```
mongosh mongodbr/127.0.0.1 x + v
[ { name: 'Luiza', street: 'Xidan', city: 'Shengzhou' } ]
nano_employees> db.employees.find({name: "Luiza"}, {"name": true, "street": true, city: true, _id: false}).pretty()
[ { name: 'Luiza', street: 'Xidan', city: 'Shengzhou' } ]
nano_employees> db.employees.find({name: "Luiza"}, {name: true, street: true, city: true, _id: false}).pretty()
[ { name: 'Luiza', street: 'Xidan', city: 'Shengzhou' } ]
nano_employees> db.employees.find({city: $or: ["Shengzhou", "Fresno"]}, {name: true, street: true, city: true, _id: false}).pretty()
Uncaught:
SyntaxError: Unexpected token, expected ",", (1:28)
> 1 | db.employees.find({city: $or: ["Shengzhou", "Fresno"]}, {name: true, street: true, city: true, _id: false}).pretty()
    |                                     ^
    2 |
nano_employees> db.employees.find({$or: [{city: "Shengzhou"}, {city: "Fresno"}]}], {name: true, street: true, city: true, _id: false}).pretty()
[
  { name: 'Karen', street: 'Sunset', city: 'Fresno' },
  { name: 'Gwen', street: 'Qianmen', city: 'Shengzhou' },
  { name: 'Luiza', street: 'Xidan', city: 'Shengzhou' },
  { name: 'Olivia', street: 'Xidan', city: 'Shengzhou' }
]
nano_employees> db.employees.find({$or: [{city: "Shengzhou"}, {city: "Fresno"}]}], {name: true, street: true, city: true, _id: false}).sort({
name: 1}).pretty()
[
  { name: 'Gwen', street: 'Qianmen', city: 'Shengzhou' },
  { name: 'Karen', street: 'Sunset', city: 'Fresno' },
  { name: 'Luiza', street: 'Xidan', city: 'Shengzhou' },
  { name: 'Olivia', street: 'Xidan', city: 'Shengzhou' }
]
nano_employees> |
```

Query 4: Return all workers that don't work for Deloitte. The result-set must be sorted by increasing worker name, and display a maximum of four documents.

```
db.employees.find({'companies.name': {'$ne': "Deloitte"}})
.sort({'name': 1}).limit(4).pretty()
```



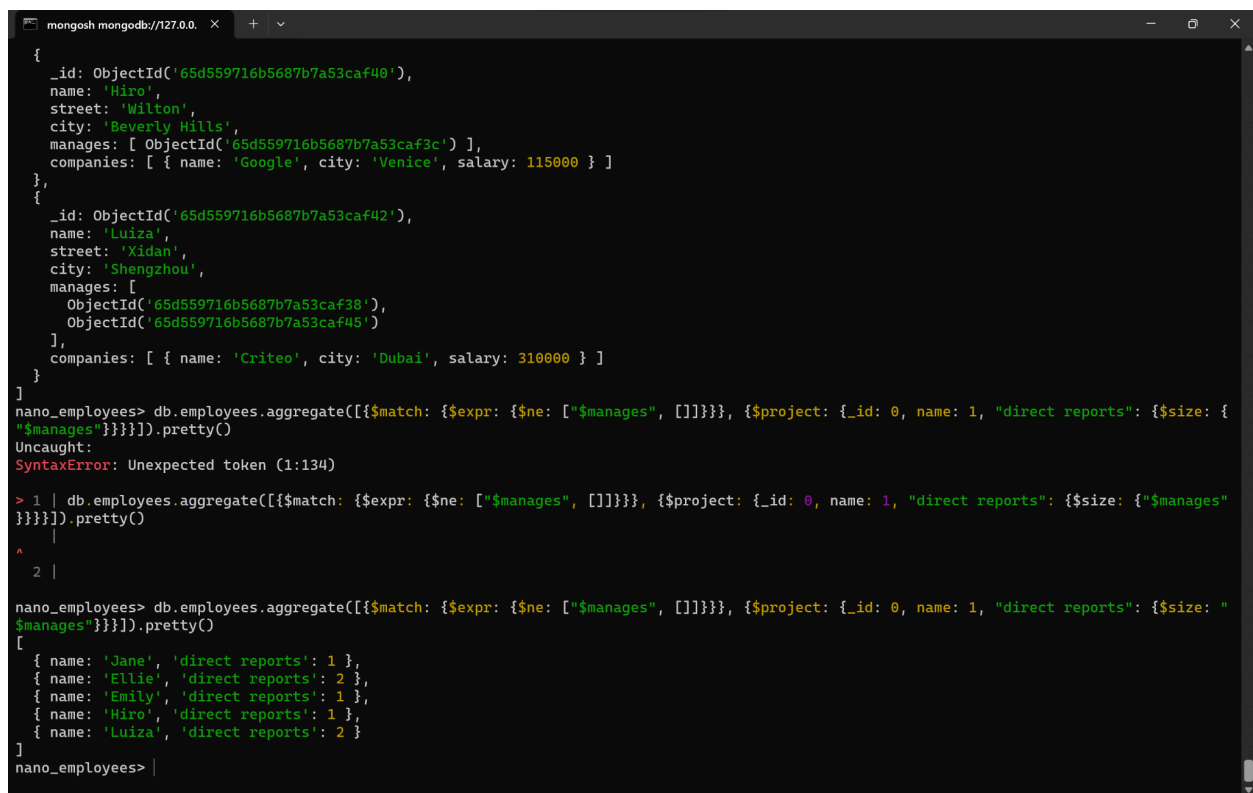
The screenshot shows a MongoDB shell window with the following command and results:

```
nano_employees> db.employees.find({'companies.name': {'$ne': "Deloitte"}}).sort({'name': 1}).limit(4).pretty()
```

```
[
  {
    _id: ObjectId('65d559716b5687b7a53caf3c'),
    name: 'Emily',
    street: 'Melrose',
    city: 'Los Angeles',
    manages: [ ObjectId('65d559716b5687b7a53caf39') ],
    companies: [
      { name: 'Criteo', city: 'Paris', salary: 450000 },
      { name: 'Google', city: 'Venice', salary: 200000 }
    ]
  },
  {
    _id: ObjectId('65d559716b5687b7a53caf40'),
    name: 'Hiro',
    street: 'Wilton',
    city: 'Beverly Hills',
    manages: [ ObjectId('65d559716b5687b7a53caf3c') ],
    companies: [ { name: 'Google', city: 'Venice', salary: 115000 } ]
  },
  {
    _id: ObjectId('65d559716b5687b7a53caf3e'),
    name: 'Ivan',
    street: 'Rodeo',
    city: 'Los Angeles',
    manages: [],
    companies: []
  },
  {
    _id: ObjectId('65d559716b5687b7a53caf38'),
    name: 'Jane',
    street: 'Third',
    city: 'Los Angeles',
    manages: [ ObjectId('65d559716b5687b7a53caf3f') ],
    companies: [
      { name: 'Hulu', city: 'Los Angeles', salary: 125572.27 },
      { name: 'Librato', city: 'San Jose', salary: 130000 }
    ]
  }
]
```

Query 5: Return the set of workers that directly manage other workers. Each document in the result set must contain a name and direct reports property, the latter being the number of workers that are directly managed by name.

```
db.employees.aggregate([
  {$match: {$expr: {$ne: ["$manages", []]}}},
  {$project: {_id: 0, name: 1,
    "direct reports": {$size: "$manages"}}}
]).pretty()
```



The screenshot shows a MongoDB terminal window with the following content:

```
mongosh mongodb://127.0.0.1:27010/employees
```

```
{
  _id: ObjectId('65d559716b5687b7a53caf40'),
  name: 'Hiro',
  street: 'Wilton',
  city: 'Beverly Hills',
  manages: [ ObjectId('65d559716b5687b7a53caf3c') ],
  companies: [ { name: 'Google', city: 'Venice', salary: 115000 } ]
},
{
  _id: ObjectId('65d559716b5687b7a53caf42'),
  name: 'Luiza',
  street: 'Xidan',
  city: 'Shengzhou',
  manages: [
    ObjectId('65d559716b5687b7a53caf38'),
    ObjectId('65d559716b5687b7a53caf45')
  ],
  companies: [ { name: 'Criteo', city: 'Dubai', salary: 310000 } ]
}
]
```

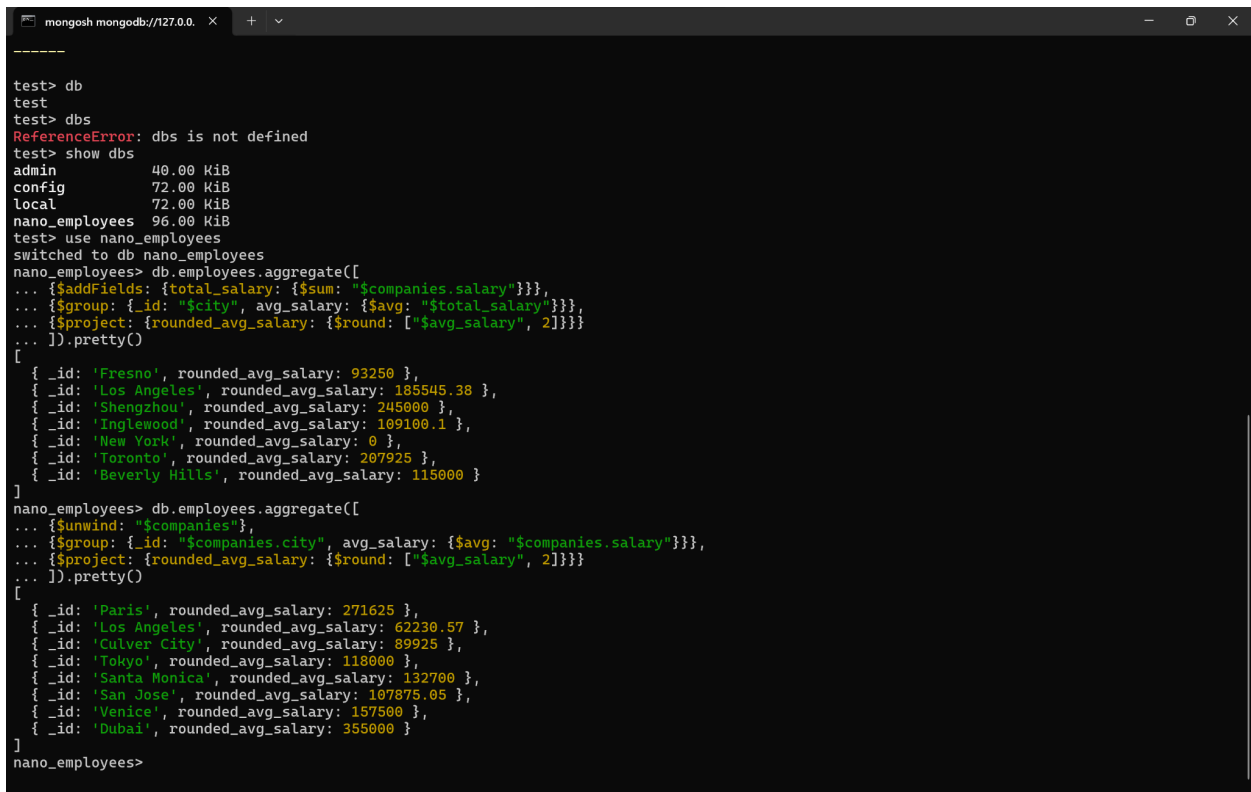
```
nano_employees> db.employees.aggregate([{$match: {$expr: {$ne: ["$manages", []]}}}, {$project: {_id: 0, name: 1, "direct reports": {$size: {"$manages"}}}}]).pretty()
Uncaught:
SyntaxError: Unexpected token (1:134)

> 1 | db.employees.aggregate([{$match: {$expr: {$ne: ["$manages", []]}}}, {$project: {_id: 0, name: 1, "direct reports": {$size: {"$manages"}}}}]).pretty()
    |
    ^
  2 |

nano_employees> db.employees.aggregate([{$match: {$expr: {$ne: ["$manages", []]}}}, {$project: {_id: 0, name: 1, "direct reports": {$size: "$manages"}}}]).pretty()
[
  { name: 'Jane', 'direct reports': 1 },
  { name: 'Ellie', 'direct reports': 2 },
  { name: 'Emily', 'direct reports': 1 },
  { name: 'Hiro', 'direct reports': 1 },
  { name: 'Luiza', 'direct reports': 2 }
]
nano_employees> |
```

Query 6: Return all cities with companies located in that city, along with the average salary for people working in (the companies headquartered in) that city. The salary should be rounded to two digits of precision after the decimal point. Each document in the result-set should have these properties: `_id` (the name of the city) and `rounded_avg_salary`.

```
db.employees.aggregate([
  {$unwind: "$companies"},
  {$group: {_id: "$companies.city",
    avg_salary: {$avg: "$companies.salary"}}},
  {$project: {rounded_avg_salary: {$round: ["$avg_salary", 2]}}}
]).pretty()
```



```
-----
test> db
test
test> dbs
ReferenceError: dbs is not defined
test> show dbs
admin          40.00 KiB
config         72.00 KiB
local          72.00 KiB
nano_employees 96.00 KiB
test> use nano_employees
switched to db nano_employees
nano_employees> db.employees.aggregate([
...  {$addFields: {total_salary: {$sum: "$companies.salary"}}},
...  {$group: {_id: "$city", avg_salary: {$avg: "$total_salary"}}},
...  {$project: {rounded_avg_salary: {$round: ["$avg_salary", 2]}}}
... ]).pretty()
[
  { _id: 'Fresno', rounded_avg_salary: 93250 },
  { _id: 'Los Angeles', rounded_avg_salary: 185545.38 },
  { _id: 'Shengzhou', rounded_avg_salary: 245000 },
  { _id: 'Inglewood', rounded_avg_salary: 109100.1 },
  { _id: 'New York', rounded_avg_salary: 0 },
  { _id: 'Toronto', rounded_avg_salary: 207925 },
  { _id: 'Beverly Hills', rounded_avg_salary: 115000 }
]
nano_employees> db.employees.aggregate([
...  {$unwind: "$companies"},
...  {$group: {_id: "$companies.city", avg_salary: {$avg: "$companies.salary"}}},
...  {$project: {rounded_avg_salary: {$round: ["$avg_salary", 2]}}}
... ]).pretty()
[
  { _id: 'Paris', rounded_avg_salary: 271625 },
  { _id: 'Los Angeles', rounded_avg_salary: 62230.57 },
  { _id: 'Culver City', rounded_avg_salary: 89925 },
  { _id: 'Tokyo', rounded_avg_salary: 118000 },
  { _id: 'Santa Monica', rounded_avg_salary: 132700 },
  { _id: 'San Jose', rounded_avg_salary: 107875.05 },
  { _id: 'Venice', rounded_avg_salary: 157500 },
  { _id: 'Dubai', rounded_avg_salary: 355000 }
]
nano_employees>
```

Query 7: Which company **name** has the most people working for it across **all locations**? How many people work for it? The result-set should be an array containing a single document: the document has an `_id` property containing the name of the company along with a `num_people` property.

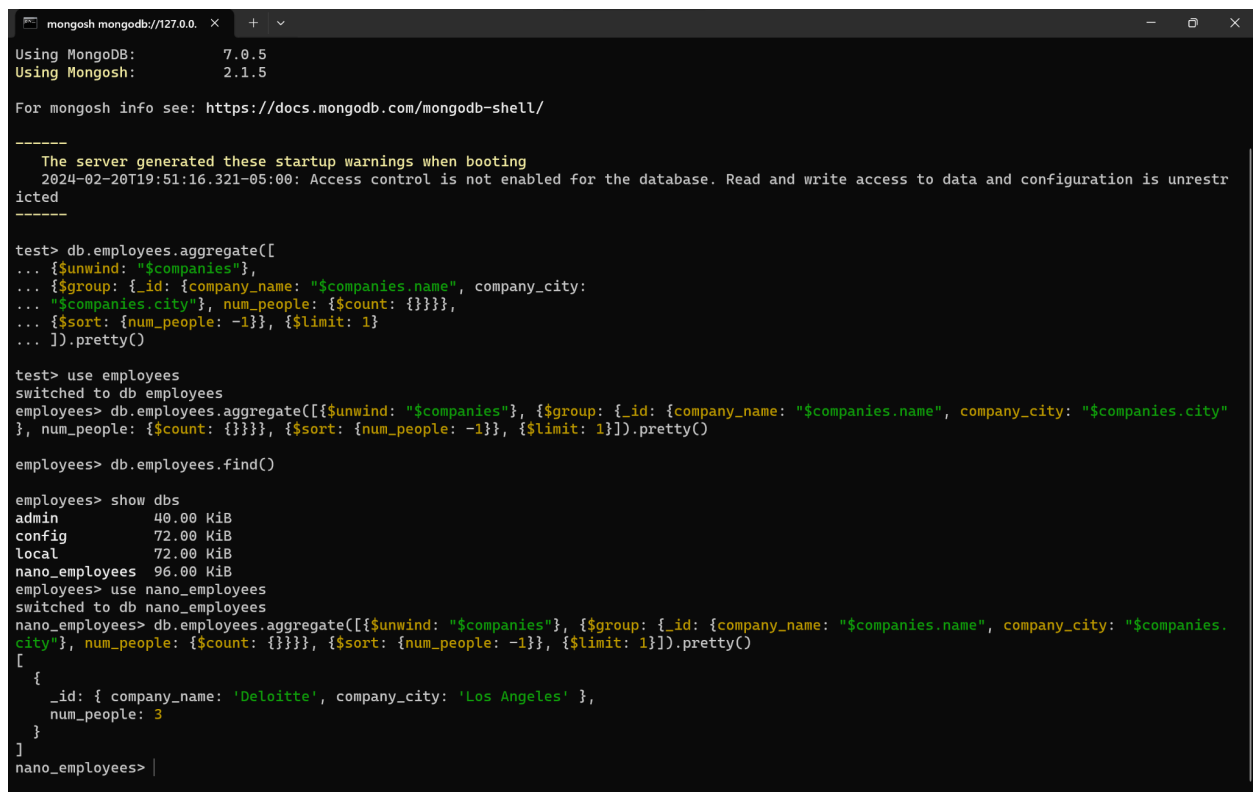
```
db.employees.aggregate([
  {$unwind: "$companies"},
  {$group: {_id: {name: "$name", company: "$companies.name"}}},
  {$group: {_id: "$_id.company", num_people: {$count: {}}}},
  {$sort: {num_people: -1}}, {$limit: 1}
]).pretty()
```



```
mongosh mongodb://127.0.0.1:27000/
le: {$count: {}}, {$project: {"$city": 1}}, {$sort: {"$num_people": -1}}, {$limit: 1})).pretty()
MongoServerError[Location16410]: Invalid $project :: caused by :: FieldPath field names may not start with '$'. Consider using $getField or $setField.
nano_employees> db.employees.aggregate([{$unwind: "$companies"}, {$group: {_id: {name: "$companies.name", city: "$companies.city"}, num_people: {$count: {}}}}, {$sort: {num_people: -1}}, {$limit: 1})).pretty()
[
  { _id: { name: 'Hulu', city: 'Los Angeles' }, num_people: 1 },
  { _id: { name: 'Librato', city: 'San Jose' }, num_people: 2 },
  { _id: { name: 'Criteo', city: 'Dubai' }, num_people: 2 },
  { _id: { name: 'Google', city: 'Venice' }, num_people: 2 },
  { _id: { name: 'Deloitte', city: 'Los Angeles' }, num_people: 3 },
  { _id: { name: 'Sony', city: 'Culver City' }, num_people: 1 },
  { _id: { name: 'Sony', city: 'Tokyo' }, num_people: 1 },
  { _id: { name: 'Iviva', city: 'Santa Monica' }, num_people: 1 },
  { _id: { name: 'Criteo', city: 'Paris' }, num_people: 2 }
]
nano_employees> db.employees.aggregate([{$unwind: "$companies"}, {$group: {_id: {name: "$companies.name", city: "$companies.city"}, num_people: {$count: {}}}}, {$sort: {num_people: -1}}, {$limit: 1})).pretty()
[ { _id: { name: 'Deloitte', city: 'Los Angeles' }, num_people: 3 } ]
nano_employees> db.employees.aggregate([{$unwind: "$companies"}, {$group: {_id: {name: "$companies.name", city: "$companies.city"}, num_people: {$count: {}}}}, {$sort: {num_people: -1}}, {$limit: 1})).pretty()
[ { _id: { name: 'Criteo', city: 'Paris' }, num_people: 4 } ]
nano_employees>
```


Query 8: Which company **name** has the most people working for it in a **given location**? How many people work for it? The result-set should be an array containing a single document: the `_id` of that document should be an object containing `company_name` and `company_city` properties. In addition, the top-level document should contain a `num_people` property.

```
db.employees.aggregate([
  {$unwind: "$companies"},
  {$group: {_id: {company_name: "$companies.name", company_city:
    "$companies.city"}, num_people: {$count: {}}}},
  {$sort: {num_people: -1}}, {$limit: 1}
]).pretty()
```



```
mongosh mongodb://127.0.0.1:27010
Using MongoDB: 7.0.5
Using Mongosh: 2.1.5

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

-----
The server generated these startup warnings when booting
2024-02-20T19:51:16.321-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestr
icted
-----

test> db.employees.aggregate([
... {$unwind: "$companies"},
... {$group: {_id: {company_name: "$companies.name", company_city:
... "$companies.city"}, num_people: {$count: {}}}},
... {$sort: {num_people: -1}}, {$limit: 1}
... ]).pretty()

test> use employees
switched to db employees
employees> db.employees.aggregate([{$unwind: "$companies"}, {$group: {_id: {company_name: "$companies.name", company_city: "$companies.city"
}, num_people: {$count: {}}}}, {$sort: {num_people: -1}}, {$limit: 1}]).pretty()

employees> db.employees.find()

employees> show dbs
admin          40.00 KiB
config         72.00 KiB
local          72.00 KiB
nano_employees 96.00 KiB
employees> use nano_employees
switched to db nano_employees
nano_employees> db.employees.aggregate([{$unwind: "$companies"}, {$group: {_id: {company_name: "$companies.name", company_city: "$companies.
city"}, num_people: {$count: {}}}}, {$sort: {num_people: -1}}, {$limit: 1}]).pretty()
[
  {
    _id: { company_name: 'Deloitte', company_city: 'Los Angeles' },
    num_people: 3
  }
]
nano_employees> |
```

Query 9: Return an array containing a document for each company. Each document specifies the company name in its `_id` property. The document also contains payroll (the sum of worker salaries), `num_people` (number of people working for it across all locations), and `min_salary`, `max_salary`, `avg_salary` properties. The result set should be sorted in descending payroll values.

I have two screenshots, because the results were too big to return in one.

```
db.employees.aggregate([
  {$unwind: "$companies"},
  {$group: {_id: {name: "$name", company: "$companies.name"},
    salary: {$sum: "$companies.salary"}}},
  {$group: {_id: "$_id.company", payroll: {$sum: "$salary"},
    num_people: {$count: {}}, min_salary: {$min: "$salary"},
    max_salary: {$max: "$salary"},
    avg_salary: {$avg: "$salary"}}},
  {$sort: {payroll: -1}}
]).pretty()
```



The screenshot shows a MongoDB shell window with the following command and output:

```
nano_employees> db.employees.aggregate([{$unwind: "$companies"}, {$group: {_id: {name: "$name", company: "$companies.name"}, salary: {$sum: "$companies.salary"}}}, {$group: {_id: "$_id.company", payroll: {$sum: "$salary"}, num_people: {$count: {}}, min_salary: {$min: "$salary"}, max_salary: {$max: "$salary"}, avg_salary: {$avg: "$salary"}}}, {$sort: {payroll: -1}}]).pretty()
```

```
[
  {
    _id: 'Criteo',
    payroll: 1253250,
    num_people: 4,
    min_salary: 93250,
    max_salary: 450000,
    avg_salary: 313312.5
  },
  {
    _id: 'Google',
    payroll: 315000,
    num_people: 2,
    min_salary: 115000,
    max_salary: 200000,
    avg_salary: 157500
  },
  {
    _id: 'Librato',
    payroll: 215750.1,
    num_people: 2,
    min_salary: 85750.1,
    max_salary: 130000,
    avg_salary: 107875.05
  },
  {
    _id: 'Sony',
    payroll: 207925,
    num_people: 1,
    min_salary: 207925,
    max_salary: 207925,
    avg_salary: 207925
  },
  {
    _id: 'Iviva',
    payroll: 132700,
    num_people: 1,
    min_salary: 132700,
    max_salary: 132700,
    avg_salary: 132700
  }
]
```

```
mongosh mongodb://127.0.0.1:27017/
{
  "_id": "Librato",
  "payroll": 215750.1,
  "num_people": 2,
  "min_salary": 85750.1,
  "max_salary": 130000,
  "avg_salary": 107875.05
},
{
  "_id": "Sony",
  "payroll": 207925,
  "num_people": 1,
  "min_salary": 207925,
  "max_salary": 207925,
  "avg_salary": 207925
},
{
  "_id": "Iviva",
  "payroll": 132700,
  "num_people": 1,
  "min_salary": 132700,
  "max_salary": 132700,
  "avg_salary": 132700
},
{
  "_id": "Hulu",
  "payroll": 125572.27,
  "num_people": 1,
  "min_salary": 125572.27,
  "max_salary": 125572.27,
  "avg_salary": 125572.27
},
{
  "_id": "Deloitte",
  "payroll": 123350,
  "num_people": 3,
  "min_salary": 23350,
  "max_salary": 75000,
  "avg_salary": 41116.666666666664
}
]
nano_employees> |
```