

1. `db.customers.aggregate({$group: {_id: null, maxDrugQuantity: {$max: '$drugQuantity'}}});`

Find the maximum quantity of drugs:

```
> db.customers.aggregate({$group: {_id: null, maxDrugQuantity: {$max: '$drugQuantity'}}});
Error: clone(t={}){const r=t.loc||{};return e({loc:new Position("line"in r?r.line:this.loc.line,"
> db.customers.aggregate({$group: {_id: null, maxDrugQuantity: {$max: '$drugQuantity'}}});
< {
  _id: null,
  maxDrugQuantity: 15
}
drugManagementSystem>
```

2. `db.customers.aggregate({$match: {'payment': 'card'}}, {$group: {_id: '$Name'}});`

Find all names which are having the payment method card.

```
> db.customers.aggregate({$match: {'payment': 'card'}}, {$group: {_id: '$Name'}});
< {
  _id: 'Arun'
}
{
  _id: 'John'
}
{
  _id: 'Aliza'
}
```

3. `db.customers.aggregate([{$match: {'payment': 'card'}}, {$sort: {'drugQuantity': -1}}, {$limit: 1}, {$group: {_id: '$Name'}}];`

Display the name which is having the payment method is card and having the maximum drugQuantity.

```
> db.customers.aggregate([{$match: {'payment': 'card'}}, {$sort: {'drugQuantity': -1}}, {$limit: 1}, {$group: {_id: '$Name'}}];
< {
  _id: 'Aliza'
}
```

4. `db.customers.aggregate([ { $match: { 'payment': 'card' } } ,{$sort: {'Name': -1 }}, {$group: { _id: '$Name'}}]);`

Here, names are sorted in descending order which are having payment method card.

```
> db.customers.aggregate([ { $match: { 'payment': 'card' } } ,{$sort: {'Name': -1 }}, {$group: { _id: '$Name'}}]);
< {
  _id: 'John'
}
{
  _id: 'Aliza'
}
{
  _id: 'Arun'
}
drugManagementSystem>
```

5. `db.customers.aggregate([{$sort: {'amount': -1}}]);`

Sort all the data in descending order according to amount.

```
>_MONGOSH
}
> db.customers.aggregate([{$sort: {'amount': -1}}]);
< {
  _id: ObjectId("6438fa4242b0cdbcdf3d1ff8"),
  Name: 'Aliza',
  phoneNumber: 241256965652,
  amount: 2500,
  drugQuantity: 15,
  payment: 'card',
  drug: [
    {
      drugId: ObjectId("6438f08742b0cdbcdf3d1fce")
    }
  ]
}
{
  _id: ObjectId("6438f5df42b0cdbcdf3d1ff7"),
  Name: 'jack',
  phoneNumber: 241256965652,
  amount: 1600,
  drugQuantity: 7,
  payment: 'online',
  drug: [
```

6. `db.customers.aggregate([{$group: { _id: null, avgAmount: { $avg: '$amount' } } }]);`

This is calculating the average of the amount.

```
> db.customers.aggregate([{$group: { _id: null, avgAmount: { $avg: '$amount' } } }]);
< {
  _id: null,
  avgAmount: 935
}
drugManagementSystem>
```

7. `db.customers.aggregate([{$group: { _id: 'Null', Sum: {$sum: '$amount'}}}]);`

Here, the total sum of the amount.

```
> db.customers.aggregate([{$group: { _id: 'Null', Sum: {$sum: '$amount'}}}]);
< {
  _id: 'Null',
  Sum: 9350
}
drugManagementSystem>
```

8. `db.customers.aggregate({ $match: { 'payment': 'online' } }, {$group: { _id: 'Name', 'amount': {$push: {amount: '$amount'}}}});`

Here, making the array of amounts which are having the payment method online.

```
> db.customers.aggregate({ $match: { 'payment': 'online' } }, {$group: { _id: 'Name', 'amount': {$push: {amount: '$amount'}}}});
< {
  _id: 'Name',
  amount: [
    {
      amount: 1000
    },
    {
      amount: 1200
    },
    {
      amount: 1600
    }
  ]
}
```

drugManagementSystem>

9. `db.customers.aggregate([{$sort: {'drugQuantity': 1}}, {$group: { _id: null, 'Quantity': {$push: {Quantity: '$drugQuantity'}}}}];`

Here, we are sorting the drug quantity in increasing order and pushing that in array.

```
> db.customers.aggregate([{$sort: {'drugQuantity': 1}}, {$group: { _id: null, 'Quantity': {$push: {Quantity: '$drugQuantity'}}}}];
< {
  _id: null,
  Quantity: [
    {
      Quantity: 1
    },
    {
      Quantity: 2
    },
    {
      Quantity: 3
    },
    {
      Quantity: 3
    },
    {
      Quantity: 4
    }
  ]
}
```

10. `db.customers.aggregate([{$match: {'amount': { $gt: 1000 } }}, {$group: { _id: '$amount', 'Name': { $push: '$Name' } } }]);`

Here, we find the amount greater than 1000 and then push all the names in the array.

```
My Queries
> MONGOSH

> db.customers.aggregate([{$match: { 'amount': { $gt: 1000 } }}, {$group: { _id: '$amount', 'Name': { $push: '$Name' } } }]);
< [
  {
    _id: 1050,
    Name: [
      'Alex'
    ]
  },
  {
    _id: 2500,
    Name: [
      'Aliza'
    ]
  },
  {
    _id: 1200,
    Name: [
      'Mishel'
    ]
  },
  {
    _id: 1600,
    Name: [

```

11. `db.customers.aggregate([{$match: {'amount': { $gt: 1000, $lt: 2000 } }}, {$group: { _id: '$amount', 'Name': { $push: '$Name' } } }]);`

Here, we are filtering the data by the range that how much data is present between the amount of 1000 to 2000, and we push the data into array

```
db.customers.aggregate([{$match: { 'amount': { $gt: 1000, $lt: 2000 } }}, {$group: { _id: '$amount', 'Name': { $push: '$Name' } } }]);
< [
  {
    _id: 1200,
    Name: [
      'Mishel'
    ]
  },
  {
    _id: 1050,
    Name: [
      'Alex'
    ]
  },
  {
    _id: 1600,
    Name: [
      'jack'
    ]
  }
]
drugManagementSystem>
```

12. `db.customers.aggregate([{$match: { 'payment': 'cash' } }, {$group: { _id: 'null', 'TotalCustomer': { $sum: 1 } } }]);`

Here, we find the total customers that are having the payment method cash.

```
> db.customers.aggregate([{$match: { 'payment': 'cash' } }, {$group: { _id: 'null', 'TotalCustomer': { $sum: 1 } } }]);
< {
  _id: 'null',
  TotalCustomer: 4
}
drugManagementSystem>
```

13. `db.customers.aggregate([{$match: { 'Name': 'Alex' } }, {$addFields: { 'Price': { $multiply: ['$amount', '$drugQuantity'] } } }, {$group: { _id: null, 'TotalPrice': { $sum: '$Price' } } }]);`

Here, we are multiply the amount and quantity of Alex(Customer)

```
db.customers.aggregate([{$match:
{
  _id: null,
  TotalPrice: 10500
}
drugManagementSystem>
```

14. `db.customers.aggregate([{$group: { _id: '$amount', 'Name': { $push: '$Name' } } }]);`

Here, we are pushing all the customer names in array that are present in collection.

```
> db.customers.aggregate([{$group: { _id: '$amount', 'Name': { $push: '$Name' } } }]);
< {
  _id: 1050,
  Name: [
    'Alex'
  ]
}
{
  _id: 600,
  Name: [
    'Max'
  ]
}
{
  _id: 500,
  Name: [
    'somi'
  ]
}
{
  _id: 1600,
  Name: [
```

15. `db.customers.aggregate({$match:{"Name": {$in:['Alex', 'John']}}});`

Here, we are getting the object of customer names Alex and John.

```
> _MONGOSH
> db.customers.aggregate({$match:{"Name": {$in:['Alex', 'John']}}});
< {
  _id: ObjectId("6438f26142b0cdbcdf3d1fef"),
  Name: 'John',
  phoneNumber: 2412569635,
  amount: 550,
  drugQuantity: 2,
  payment: 'card',
  drugId: [
    {
      drugId: ObjectId("6438f08742b0cdbcdf3d1fe6")
    }
  ]
}
{
  _id: ObjectId("6438f2bf42b0cdbcdf3d1ff0"),
  Name: 'Alex',
  phoneNumber: 2412569635,
  amount: 1050,
  drugQuantity: 10,
```



16. `db.customers.aggregate([{$match:{"Name": {$in:['Alex', 'John']}}, {$match:{"payment": 'cash'}}]);`

Here, we are finding between Alex and John which one is having the payment method cash.

```
    }
  > db.customers.aggregate([{$match:{"Name": {$in:['Alex', 'John']}}, {$match:{"payment": 'cash'}}]);
  < {
    _id: ObjectId("6438f2bf42b0cdbcdf3d1ff0"),
    Name: 'Alex',
    phoneNumber: 2412569635,
    amount: 1050,
    drugQuantity: 10,
    payment: 'cash',
    drugId: [
      {
        drugId: ObjectId("6438f08742b0cdbcdf3d1fc0")
      }
    ]
  }
  drugManagementSystem>
```

17. `db.customers.aggregate([{$match:{"Name": {$in:['Varun', 'Arun']}}, { $sort: { "amount": -1 } }, {$limit: 1}]);`

Here, we are finding between Varun and Arun which one is having the more amount.

```
monogosevererror: unrecognized pipeline stage name: $max
  > db.customers.aggregate([{$match:{"Name": {$in:['Varun', 'Arun']}}, { $sort: { "amount": -1 } }, {$limit: 1}]);
  < {
    _id: ObjectId("6438f12d42b0cdbcdf3d1fee"),
    Name: 'Arun',
    phoneNumber: 7412589635,
    amount: 200,
    drugQuantity: 5,
    payment: 'card',
    drugId: [
      {
        drugId: ObjectId("6438f08742b0cdbcdf3d1fd1")
      }
    ]
  }
  drugManagementSystem>
```

18. `db.customers.aggregate([{$match:{"Name": {$in:['Varun', 'Arun']}}, {$group: { _id: null, totalAmount: { $sum: "$amount" } } }]);`

Here, we are finding the sum of the amounts of Varun and Arun.

```
> db.customers.aggregate([{$match:{"Name": {$in:['Varun', 'Arun']}}, {$group: { _id: null, totalAmount: { $sum: "$amount" } } }]);
< {
  _id: null,
  totalAmount: 350
}
drugManagementSystem>
```

19. `db.customers.aggregate([{$match:{"phoneNumber": {$exists: false, $eq: null}}},{ $count: "phoneNumber" }]);`

Here, we are checking that whether is any object having empty phone Number or the PhoneNumber is exist or not.

```
> db.customers.aggregate([{$match:{"phoneNumber": {$exists: false, $eq: null}}},{ $count: "phoneNumber" }]);
<
> db.customers.aggregate([{$match: { "phoneNumber": { $in: [null, undefined] } }}, { $count: "phoneNumber" }]);
<
drugManagementSystem>
```

20. `db.customers.aggregate([{$match: { "payment": "online" }},{ $group: { _id: null, totalDrugQuantity: { $sum: "$drugQuantity" } } }]);`

Here, filtering the total drug Quantity of the payment method online.

```
> db.customers.aggregate([{$match: { "payment": "online" }},{ $group: { _id: null, totalDrugQuantity: { $sum: "$drugQuantity" } } }]);
< {
  _id: null,
  totalDrugQuantity: 16
}
drugManagementSystem>
```

21. `db.customers.aggregate([{$match: { "_id": ObjectId("6438f2bf42b0cdbcdf3d1ff0") }},{ $lookup: {from: "drugs",localField: "drugId.drugId",foreignField: "_id", as: "drugDetails"}}]);`

Here, we are finding the details of the drug through the drugId which is present in the customer collection with the lookup method.

```
> .aggregate([{$match: { "_id": ObjectId("6438f2bf42b0cdbcdf3d1ff0") }},{ $1
< {
  _id: ObjectId("6438f2bf42b0cdbcdf3d1ff0"),
  Name: 'Alex',
  phoneNumber: 2412569635,
  amount: 1050,
  drugQuantity: 10,
  payment: 'cash',
  drugId: [
    {
      drugId: ObjectId("6438f08742b0cdbcdf3d1fc0")
    }
  ],
  drugDetails: [
    {
      _id: ObjectId("6438f08742b0cdbcdf3d1fc0"),
      id: 55,
      drugName: 'Strattera',
      drugPrice: 94,
      drugQuantity: 51,
      drugType: true,
      drugDose: false
    }
  ]
}
```