o/p:

{ "_id" : "Mumbai", "employeecount" : 2 }

{ "_id" : "Hyderabad", "employeecount" : 3 }

{ "_id" : "Chennai", "employeecount" : 2 }

Aggregation Pipeline:

We can define multiple stages in the aggregation and all these stages will form pipeline, which is known as aggregation pipeline.

```
db.collection.aggregate([
{stage-1},
{stage-2},
{stage-3},
{stage-4},
{stage-5}
...
])
```

All these stages will be executed one by one.
The output of previous stage will become input to next stage.

Pipeline will take collection as input and provides aggregated results in our required format.

Note: In Documentation just explore as many stages as possible.
reference-->Operators--->Aggregation Pipeline stages

2. $sort stage:

It sorts all input documents and returns them to the pipeline in sorted order.

The $sort stage has the following prototype form:

{ $sort: { <field1>: <sort order>, <field2>: <sort order> ... } }

The <sort order> can be either 1 or -1.
1 --->Ascending Order
-1 ---> Descending Order

eg-1: Find citywise sum of salaries and print based on descending order of totalsalary?

```
db.employees.aggregate([
  { $group: {_id:"$eaddr",totalSalary:{$sum:"$esal"}}},
  { $sort:{totalSalary: -1}}
])
```

o/p:

```
{ "_id" : "Hyderabad", "totalSalary" : 12000 }
{ "_id" : "Chennai", "totalSalary" : 11000 }
{ "_id" : "Mumbai", "totalSalary" : 5000 }
```

eg-2: Find citywise number of employees and print based on alphabetical order of city name?

```
db.employees.aggregate([
{ $group: {_id:"$eaddr", employeeCount: {$sum:1}}},
{ $sort: {_id:1}}
])
```

o/p:

{ "_id" : "Chennai", "employeeCount" : 2 }
{ "_id" : "Hyderabad", "employeeCount" : 3 }
{ "_id" : "Mumbai", "employeeCount" : 2 }

3. $project stage:

By using this $project stage, we can restrict documents with our required fields only. Here we can include new fields also. With the existing field values we can create new fields also.

eg: first name and last name fields are there. We can combined these fields with just name field.

Syntax:
{ $project: { field:0|1 } }

0 or false --->To exclude the field
1 or true --->To include the field

eg-1: To find total salary of all employees?

```
db.employees.aggregate([
 { $group: {_id:null, totalSalary:{$sum:"$esal"}}},
 { $project: {_id:0}}
])
```

o/p:
{ "totalSalary" : 28000 }

eg-2: Find city wise total salary and city name should be in uppercase and sort the documents in ascending order of salaries?

```
db.employees.aggregate([
  {$group: {_id:"$eaddr",totalSalary:{$sum:"$esal"}}},
```

```
  {$project: {_id:0,city:{$toUpper:"$_id"},totalSalary:1}},
  {$sort: {totalSalary: 1}}
])
```

o/p:

```
{ "totalSalary" : 5000, "city" : "MUMBAI" }
{ "totalSalary" : 11000, "city" : "CHENNAI" }
{ "totalSalary" : 12000, "city" : "HYDERABAD" }
```

```
db.employees.aggregate([
  {$group: {_id:"$eaddr",totalSalary:{$sum:"$esal"}}},
  {$project: {_id:0,city:{$concat:["$_id"," ","City"]},totalSalary:1}},
  {$sort: {totalSalary: 1}}
])
```

o/p:

```
{ "totalSalary" : 5000, "city" : "Mumbai City" }
{ "totalSalary" : 11000, "city" : "Chennai City" }
{ "totalSalary" : 12000, "city" : "Hyderabad City" }
```

$group vs $project:

If we want to group multiple documents into a single document then we should use $group stage.

eg: $sum,$avg,$max etc

But if we want to include or exclude existing fields and to create new fields with new values or already existing field values then we can use $project. Here the number of input documents and output documents are always same. But number of fields may be changed.

4. $match stage:

To filter documents based on required condition. It is exactly same as find() method <query>.

Syntax:

```
{ $match: { <query> } }
```

eg: To find the number of employees whose salary greater than 1500. Find such employees count citywise. Display documents in ascending order of employee count.

```
db.employees.aggregate([
  {$match: {esal:{$gt: 1500}}},
  {$group: {_id:"$eaddr",employeeCount:{$sum:1}}},
  {$sort: {employeeCount:1}}
])
```

o/p:

```
{ "_id" : "Mumbai", "employeeCount" : 1 }
{ "_id" : "Chennai", "employeeCount" : 2 }
{ "_id" : "Hyderabad", "employeeCount" : 3 }
```