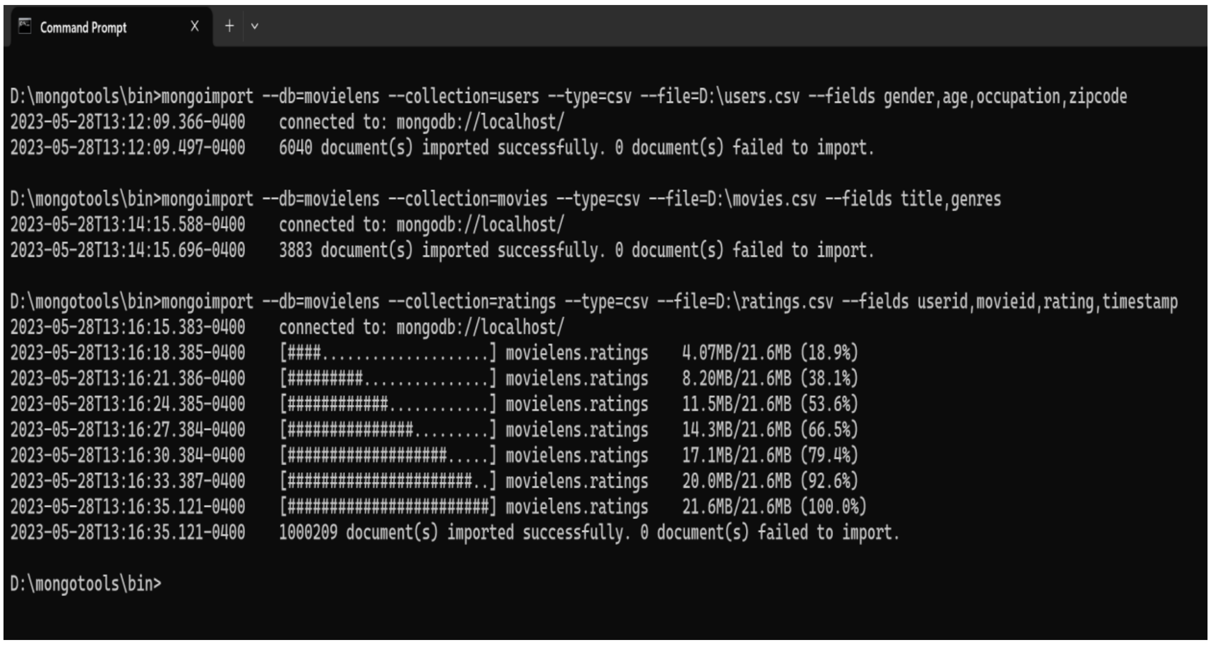
PART – 5

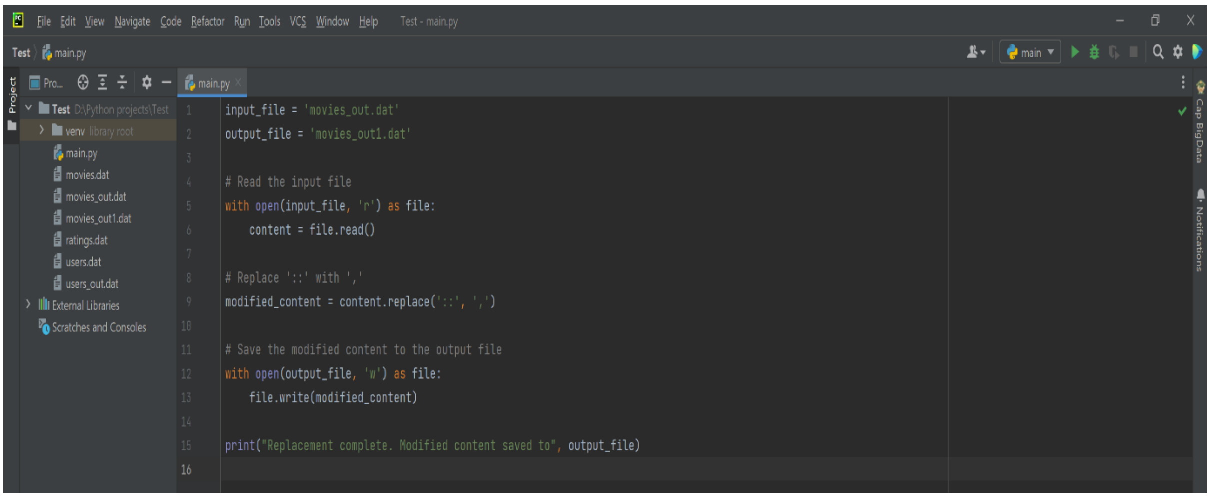
PART 5.1

Import the Movielens dataset into MongoDB. Refer to README about file contents and headings.
[https://grouplens.org/datasets/movielens/1m/ (Links to an external site.)Links to an external site.](https://grouplens.org/datasets/movielens/1m/) [you may replace :: in the dateset with comma or tab to import]



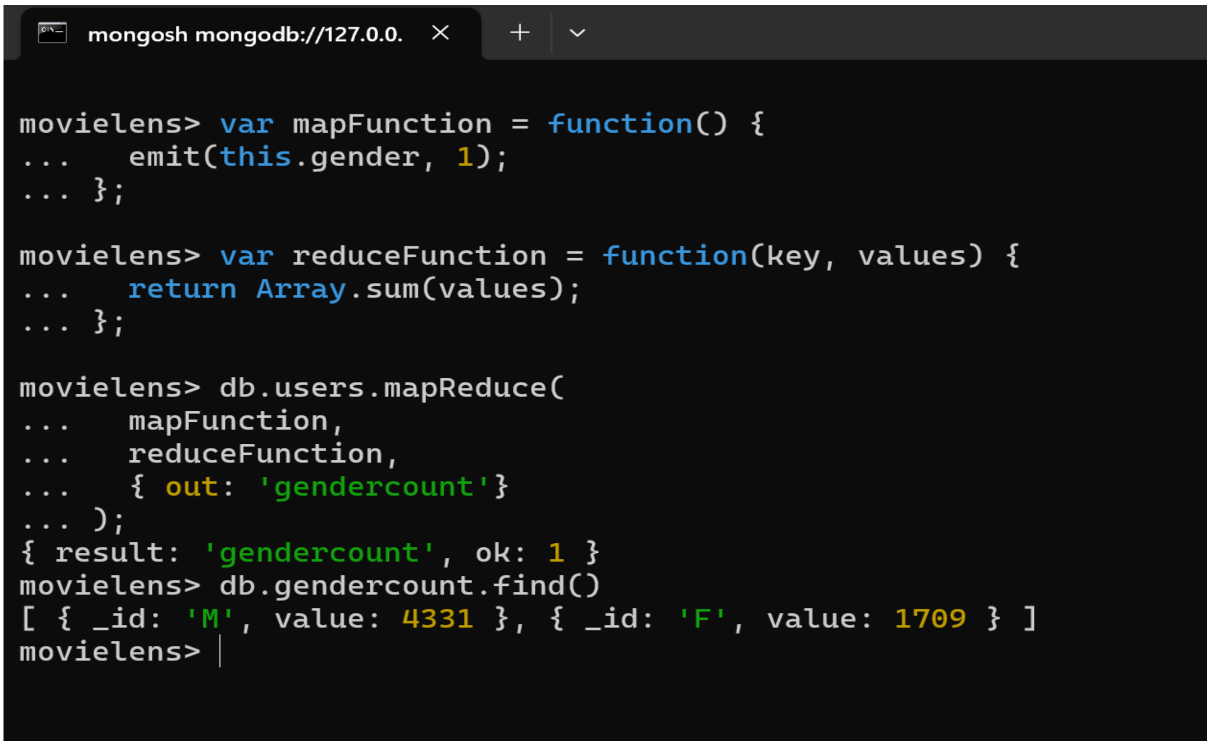
```
D:\mongotools\bin>mongoimport --db=movielens --collection=users --type=csv --file=D:\users.csv --fields gender,age,occupation,zipcode
2023-05-28T13:12:09.366-0400    connected to: mongodb://localhost/
2023-05-28T13:12:09.497-0400    6040 document(s) imported successfully. 0 document(s) failed to import.

D:\mongotools\bin>mongoimport --db=movielens --collection=movies --type=csv --file=D:\movies.csv --fields title,genres
2023-05-28T13:14:15.588-0400    connected to: mongodb://localhost/
2023-05-28T13:14:15.696-0400    3883 document(s) imported successfully. 0 document(s) failed to import.

D:\mongotools\bin>mongoimport --db=movielens --collection=ratings --type=csv --file=D:\ratings.csv --fields userid,movieid,rating,timestamp
2023-05-28T13:16:15.383-0400    connected to: mongodb://localhost/
2023-05-28T13:16:18.385-0400    [####...................] movielens.ratings    4.07MB/21.6MB (18.9%)
2023-05-28T13:16:21.386-0400    [########...............] movielens.ratings    8.20MB/21.6MB (38.1%)
2023-05-28T13:16:24.385-0400    [############...........] movielens.ratings    11.5MB/21.6MB (53.6%)
2023-05-28T13:16:27.384-0400    [###############........] movielens.ratings    14.3MB/21.6MB (66.5%)
2023-05-28T13:16:30.384-0400    [##################.....] movielens.ratings    17.1MB/21.6MB (79.4%)
2023-05-28T13:16:33.387-0400    [#####################..] movielens.ratings    20.0MB/21.6MB (92.6%)
2023-05-28T13:16:35.121-0400    [#######################] movielens.ratings    21.6MB/21.6MB (100.0%)
2023-05-28T13:16:35.121-0400    1000209 document(s) imported successfully. 0 document(s) failed to import.

D:\mongotools\bin>
```

Used python script to preprocess the data & format it to .CSV



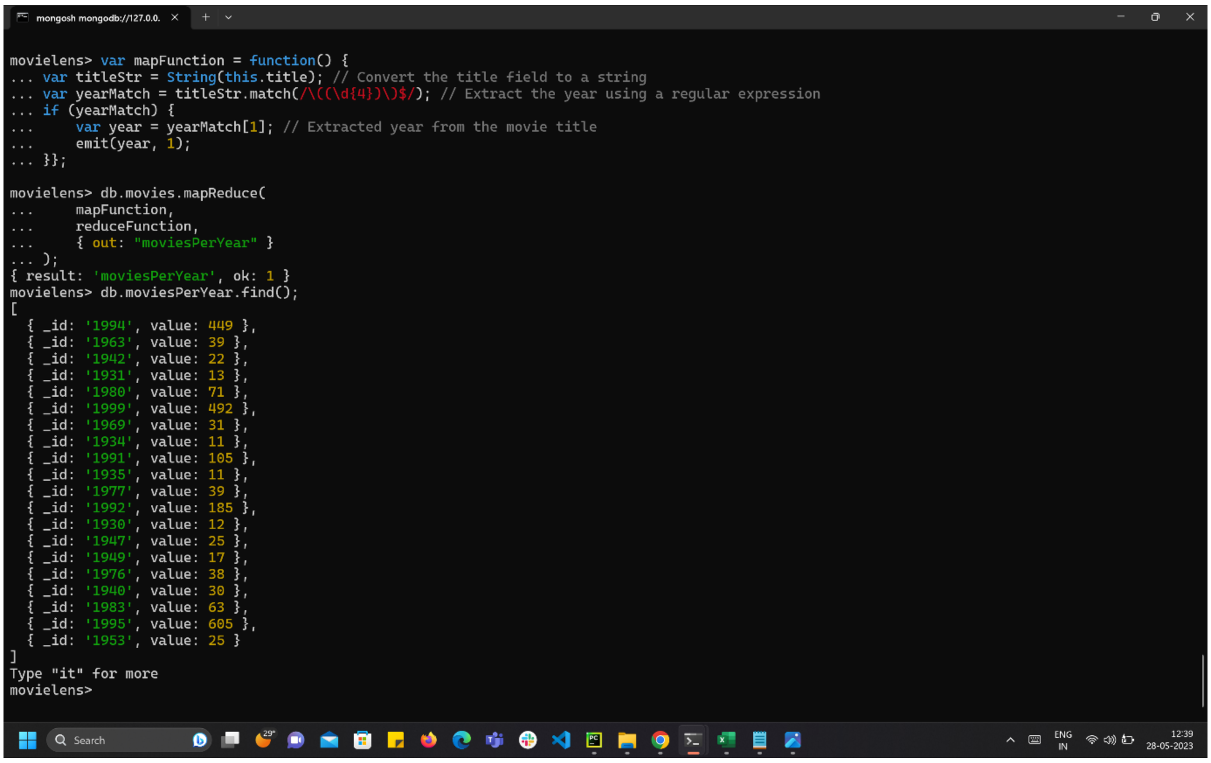
```python
input_file = 'movies_out.dat'
output_file = 'movies_out1.dat'

# Read the input file
with open(input_file, 'r') as file:
    content = file.read()

# Replace '::' with ','
modified_content = content.replace('::', ',')

# Save the modified content to the output file
with open(output_file, 'w') as file:
    file.write(modified_content)

print("Replacement complete. Modified content saved to", output_file)
```

1. Find the number Females and Males from the users collection using MapReduce. Do the same thing using count() to compare the results.



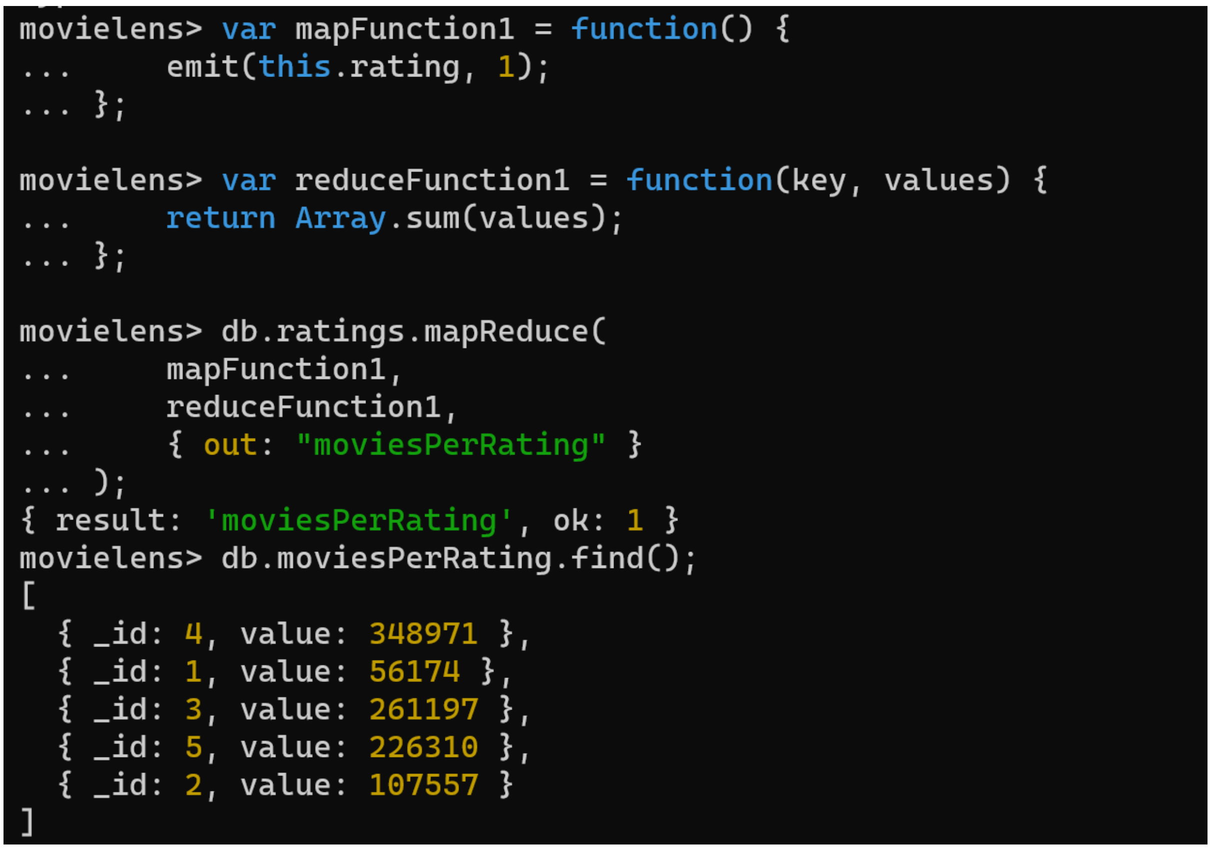
```
movielens> var mapFunction = function() {
...    emit(this.gender, 1);
... };

movielens> var reduceFunction = function(key, values) {
...    return Array.sum(values);
... };

movielens> db.users.mapReduce(
...    mapFunction,
...    reduceFunction,
...    { out: 'gendercount'}
... );
{ result: 'gendercount', ok: 1 }
movielens> db.gendercount.find()
[ { _id: 'M', value: 4331 }, { _id: 'F', value: 1709 } ]
movielens>
```

2. Find the number of Movies per year using MapReduce



```
movielens> var mapFunction = function() {
... var titleStr = String(this.title); // Convert the title field to a string
... var yearMatch = titleStr.match(/\((\d{4})\)$/); // Extract the year using a regular expression
... if (yearMatch) {
...     var year = yearMatch[1]; // Extracted year from the movie title
...     emit(year, 1);
... }};

movielens> db.movies.mapReduce(
...     mapFunction,
...     reduceFunction,
...     { out: "moviesPerYear" }
... );
{ result: 'moviesPerYear', ok: 1 }
movielens> db.moviesPerYear.find();
[
  { _id: '1994', value: 449 },
  { _id: '1963', value: 39 },
  { _id: '1942', value: 22 },
  { _id: '1931', value: 13 },
  { _id: '1980', value: 71 },
  { _id: '1999', value: 492 },
  { _id: '1969', value: 31 },
  { _id: '1934', value: 11 },
  { _id: '1991', value: 105 },
  { _id: '1935', value: 11 },
  { _id: '1977', value: 39 },
  { _id: '1992', value: 185 },
  { _id: '1930', value: 12 },
  { _id: '1947', value: 25 },
  { _id: '1949', value: 17 },
  { _id: '1976', value: 38 },
  { _id: '1940', value: 30 },
  { _id: '1983', value: 63 },
  { _id: '1995', value: 605 },
  { _id: '1953', value: 25 }
]
Type "it" for more
movielens>
```

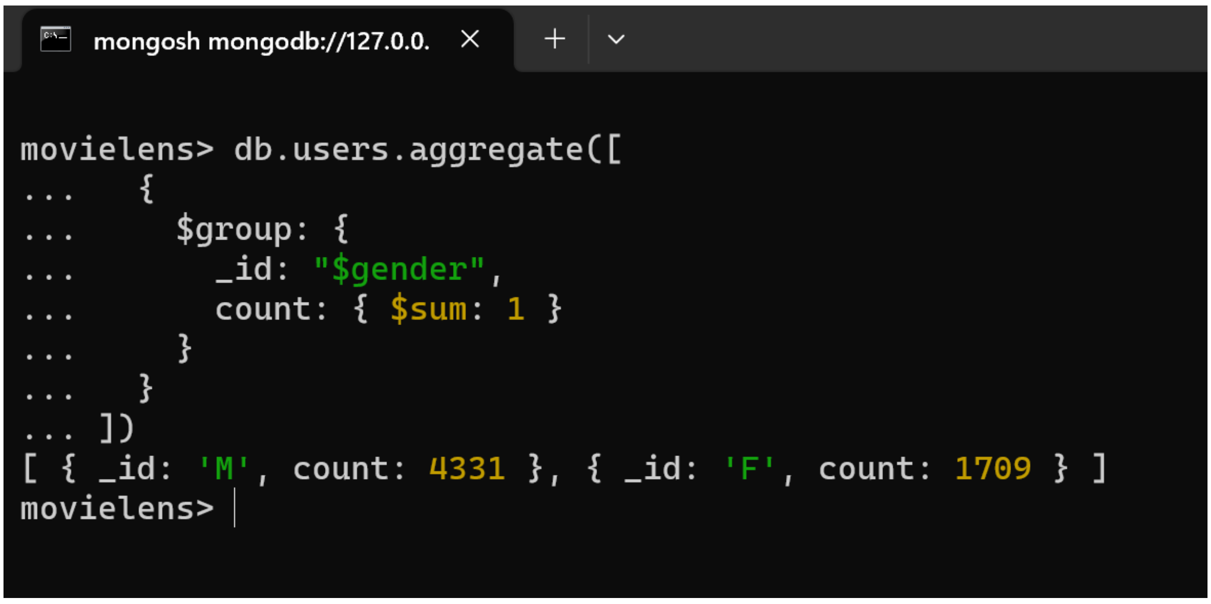
3. Find the number of Movies per rating using MapReduce



```
movielens> var mapFunction1 = function() {
...     emit(this.rating, 1);
... };

movielens> var reduceFunction1 = function(key, values) {
...     return Array.sum(values);
... };

movielens> db.ratings.mapReduce(
...     mapFunction1,
...     reduceFunction1,
...     { out: "moviesPerRating" }
... );
{ result: 'moviesPerRating', ok: 1 }
movielens> db.moviesPerRating.find();
[
  { _id: 4, value: 348971 },
  { _id: 1, value: 56174 },
  { _id: 3, value: 261197 },
  { _id: 5, value: 226310 },
  { _id: 2, value: 107557 }
]
```

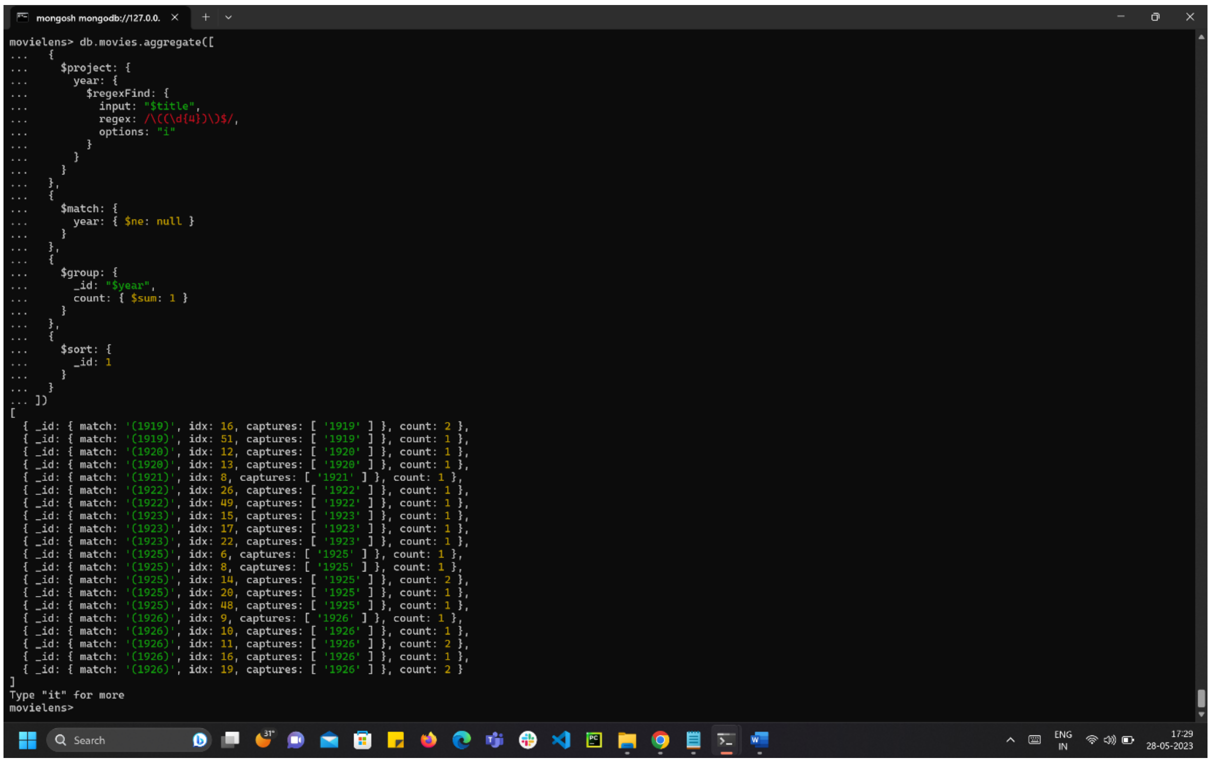
PART 5.2 – Repeat 5.1 using Aggregation Pipeline

Find the number Females and Males from the users collection using Aggregation Pipeline.



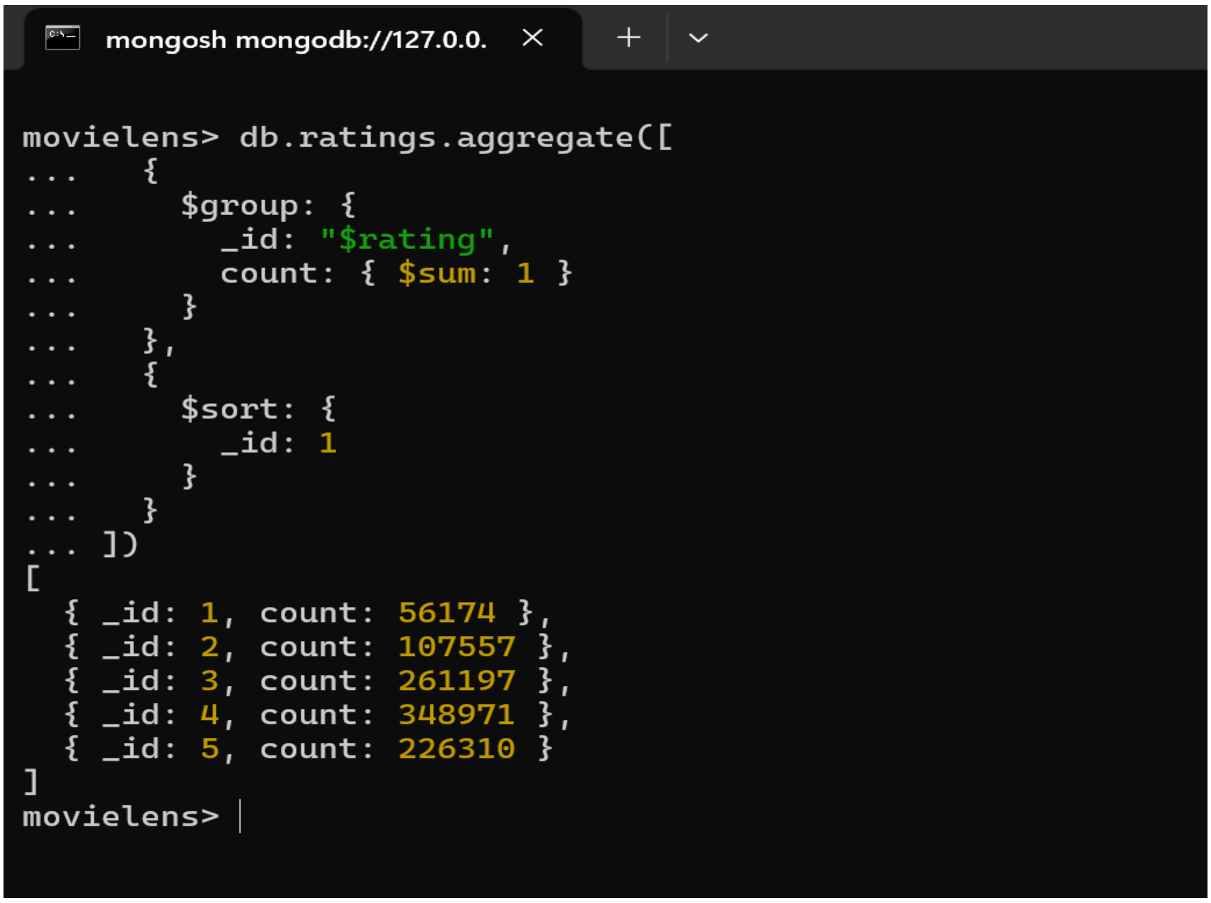
```
movielens> db.users.aggregate([
...   {
...     $group: {
...       _id: "$gender",
...       count: { $sum: 1 }
...     }
...   }
... ])
[ { _id: 'M', count: 4331 }, { _id: 'F', count: 1709 } ]
movielens>
```

Find the number of Movies per year using Aggregation Pipeline



```
movielens> db.movies.aggregate([
...   {
...     $project: {
...       year: {
...         $regexFind: {
...           input: "$title",
...           regex: /\((\d{4})\)$/,
...           options: "i"
...         }
...       }
...     }
...   },
...   {
...     $match: {
...       year: { $ne: null }
...     }
...   },
...   {
...     $group: {
...       _id: "$year",
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $sort: {
...       _id: 1
...     }
...   }
... ])
[
  { _id: { match: '(1919)', idx: 16, captures: [ '1919' ] }, count: 2 },
  { _id: { match: '(1919)', idx: 51, captures: [ '1919' ] }, count: 1 },
  { _id: { match: '(1920)', idx: 12, captures: [ '1920' ] }, count: 1 },
  { _id: { match: '(1920)', idx: 13, captures: [ '1920' ] }, count: 1 },
  { _id: { match: '(1921)', idx: 8, captures: [ '1921' ] }, count: 1 },
  { _id: { match: '(1922)', idx: 26, captures: [ '1922' ] }, count: 1 },
  { _id: { match: '(1922)', idx: 49, captures: [ '1922' ] }, count: 1 },
  { _id: { match: '(1923)', idx: 15, captures: [ '1923' ] }, count: 1 },
  { _id: { match: '(1923)', idx: 17, captures: [ '1923' ] }, count: 1 },
  { _id: { match: '(1923)', idx: 22, captures: [ '1923' ] }, count: 1 },
  { _id: { match: '(1925)', idx: 6, captures: [ '1925' ] }, count: 1 },
  { _id: { match: '(1925)', idx: 8, captures: [ '1925' ] }, count: 1 },
  { _id: { match: '(1925)', idx: 14, captures: [ '1925' ] }, count: 2 },
  { _id: { match: '(1925)', idx: 20, captures: [ '1925' ] }, count: 1 },
  { _id: { match: '(1925)', idx: 48, captures: [ '1925' ] }, count: 1 },
  { _id: { match: '(1926)', idx: 9, captures: [ '1926' ] }, count: 1 },
  { _id: { match: '(1926)', idx: 10, captures: [ '1926' ] }, count: 1 },
  { _id: { match: '(1926)', idx: 11, captures: [ '1926' ] }, count: 2 },
  { _id: { match: '(1926)', idx: 16, captures: [ '1926' ] }, count: 1 },
  { _id: { match: '(1926)', idx: 19, captures: [ '1926' ] }, count: 2 }
]
Type "it" for more
movielens>
```

Find the number of Movies per rating using Aggregation Pipeline



```
movielens> db.ratings.aggregate([
...   {
...     $group: {
...       _id: "$rating",
...       count: { $sum: 1 }
...     }
...   },
...   {
...     $sort: {
...       _id: 1
...     }
...   }
... ])
[
  { _id: 1, count: 56174 },
  { _id: 2, count: 107557 },
  { _id: 3, count: 261197 },
  { _id: 4, count: 348971 },
  { _id: 5, count: 226310 }
]
movielens>
```