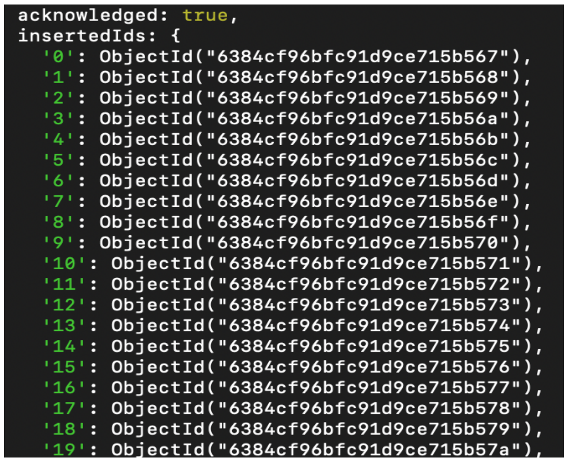
NoSQL Implementation: (A&S Consulting Group)

Q1) Inserting Many Rows in the collection newcity.

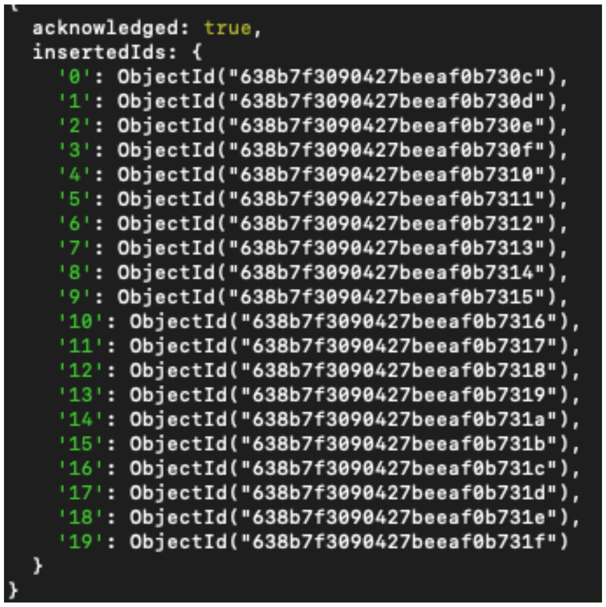
```
db.newcity.insertMany([
...   {
...     "name_city": "Abidjan",
...     "country_code": "CIV",
...     "city_proper_pop": 4765000,
...     "metroarea_pop": 0,
...     "urbanarea_pop": 4765000
...   },
...   {
...     "name_city": "Abu Dhabi",
...     "country_code": "ARE",
...     "city_proper_pop": 1145000,
...     "metroarea_pop": 0,
...     "urbanarea_pop": 1145000
...   },
...   {
...     "name_city": "Abuja",
...     "country_code": "NGA",
...     "city_proper_pop": 1235880,
...     "metroarea_pop": 6000000,
...     "urbanarea_pop": 1235880
...
     }])
```



```
 acknowledged: true,
 insertedIds: {
   '0': ObjectId("6384cf96bfc91d9ce715b567"),
   '1': ObjectId("6384cf96bfc91d9ce715b568"),
   '2': ObjectId("6384cf96bfc91d9ce715b569"),
   '3': ObjectId("6384cf96bfc91d9ce715b56a"),
   '4': ObjectId("6384cf96bfc91d9ce715b56b"),
   '5': ObjectId("6384cf96bfc91d9ce715b56c"),
   '6': ObjectId("6384cf96bfc91d9ce715b56d"),
   '7': ObjectId("6384cf96bfc91d9ce715b56e"),
   '8': ObjectId("6384cf96bfc91d9ce715b56f"),
   '9': ObjectId("6384cf96bfc91d9ce715b570"),
   '10': ObjectId("6384cf96bfc91d9ce715b571"),
   '11': ObjectId("6384cf96bfc91d9ce715b572"),
   '12': ObjectId("6384cf96bfc91d9ce715b573"),
   '13': ObjectId("6384cf96bfc91d9ce715b574"),
   '14': ObjectId("6384cf96bfc91d9ce715b575"),
   '15': ObjectId("6384cf96bfc91d9ce715b576"),
   '16': ObjectId("6384cf96bfc91d9ce715b577"),
   '17': ObjectId("6384cf96bfc91d9ce715b578"),
   '18': ObjectId("6384cf96bfc91d9ce715b579"),
   '19': ObjectId("6384cf96bfc91d9ce715b57a"),
```

Q2) Inserting many rows in the collection customer

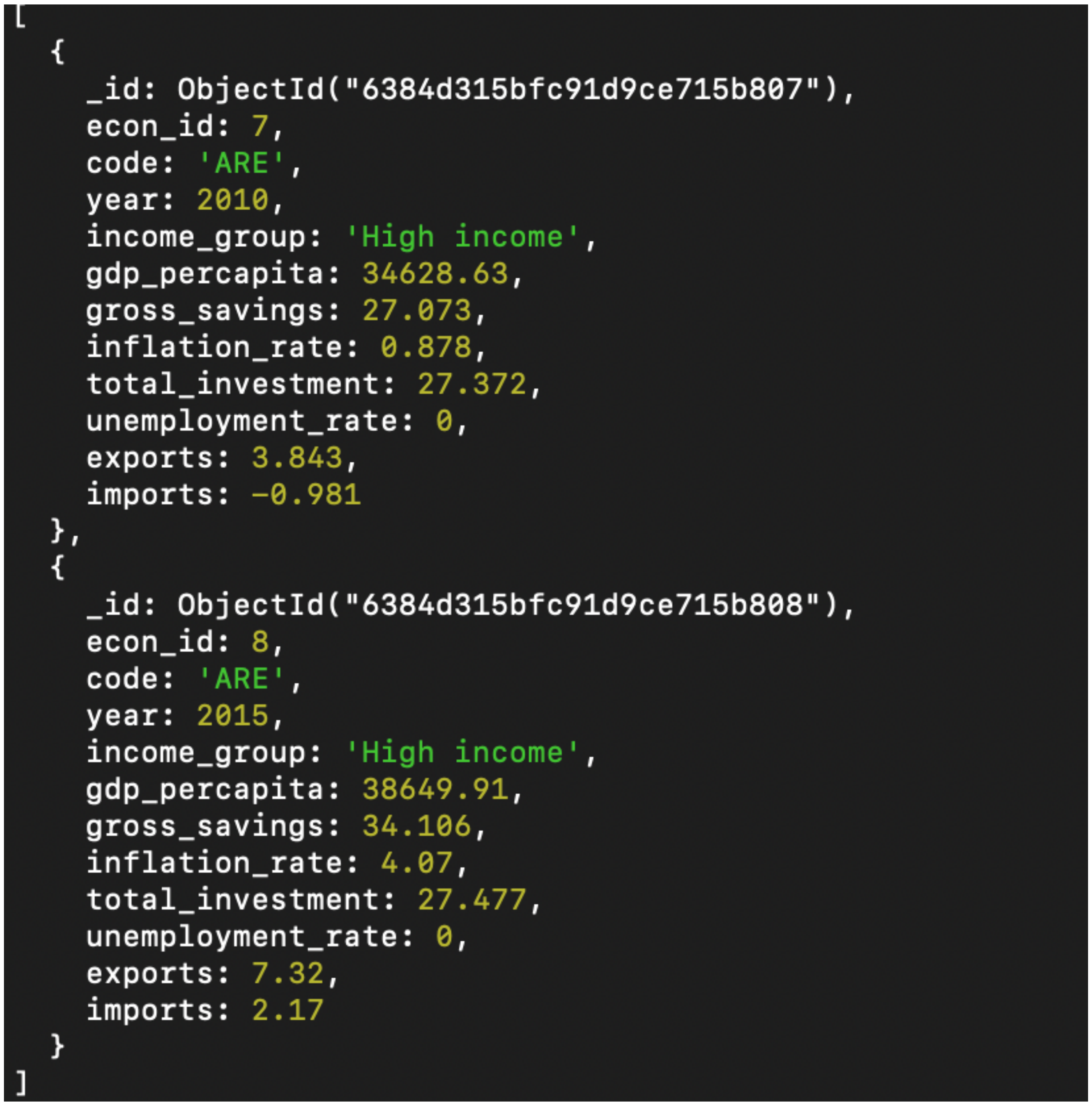
```
ASConsutingGroup> db.customer.insertMany([{
...     "Cust_Id": 4,
...     "Customer_First_Name": "Sam",
...     "Customer_Last_Name": "Li",
...     "Customer_DOB": "2000-05-05",
...     "Street": "3, Park Street",
...     "City": "Biejling",
...     "Country": "China",
...     "Postal_Code": 99849,
...     "Phone": 99553246
...   },
...   {
...     "Cust_Id": 5,
...     "Customer_First_Name": "Jennifer",
...     "Customer_Last_Name": "Rose",
...     "Customer_DOB": "1980-08-24",
...     "Street": "99 Calum Ave",
...     "City": "Barcelona",
...     "Country": "Spain",
...     "Postal_Code": 28902,
...     "Phone": [53251345, 65656565]
...   },
...   {
...     "Cust_Id": 6,
...     "Customer_First_Name": "Manuel",
...     "Customer_Last_Name": "Ola",
...     "Customer_DOB": "1977-06-22",
...     "Street": "5 Hery Road",
...     "City": "Singapore",
...     "Country": "Singapore",
...     "Postal_Code": 89764,
...     "Phone": [66556655, 78587779]
...   },
```



```
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("638b7f3090427beeaf0b730c"),
    '1': ObjectId("638b7f3090427beeaf0b730d"),
    '2': ObjectId("638b7f3090427beeaf0b730e"),
    '3': ObjectId("638b7f3090427beeaf0b730f"),
    '4': ObjectId("638b7f3090427beeaf0b7310"),
    '5': ObjectId("638b7f3090427beeaf0b7311"),
    '6': ObjectId("638b7f3090427beeaf0b7312"),
    '7': ObjectId("638b7f3090427beeaf0b7313"),
    '8': ObjectId("638b7f3090427beeaf0b7314"),
    '9': ObjectId("638b7f3090427beeaf0b7315"),
    '10': ObjectId("638b7f3090427beeaf0b7316"),
    '11': ObjectId("638b7f3090427beeaf0b7317"),
    '12': ObjectId("638b7f3090427beeaf0b7318"),
    '13': ObjectId("638b7f3090427beeaf0b7319"),
    '14': ObjectId("638b7f3090427beeaf0b731a"),
    '15': ObjectId("638b7f3090427beeaf0b731b"),
    '16': ObjectId("638b7f3090427beeaf0b731c"),
    '17': ObjectId("638b7f3090427beeaf0b731d"),
    '18': ObjectId("638b7f3090427beeaf0b731e"),
    '19': ObjectId("638b7f3090427beeaf0b731f")
  }
}
```

Q3) Show the first two results of the Economy summary, which includes the income group as High Income.

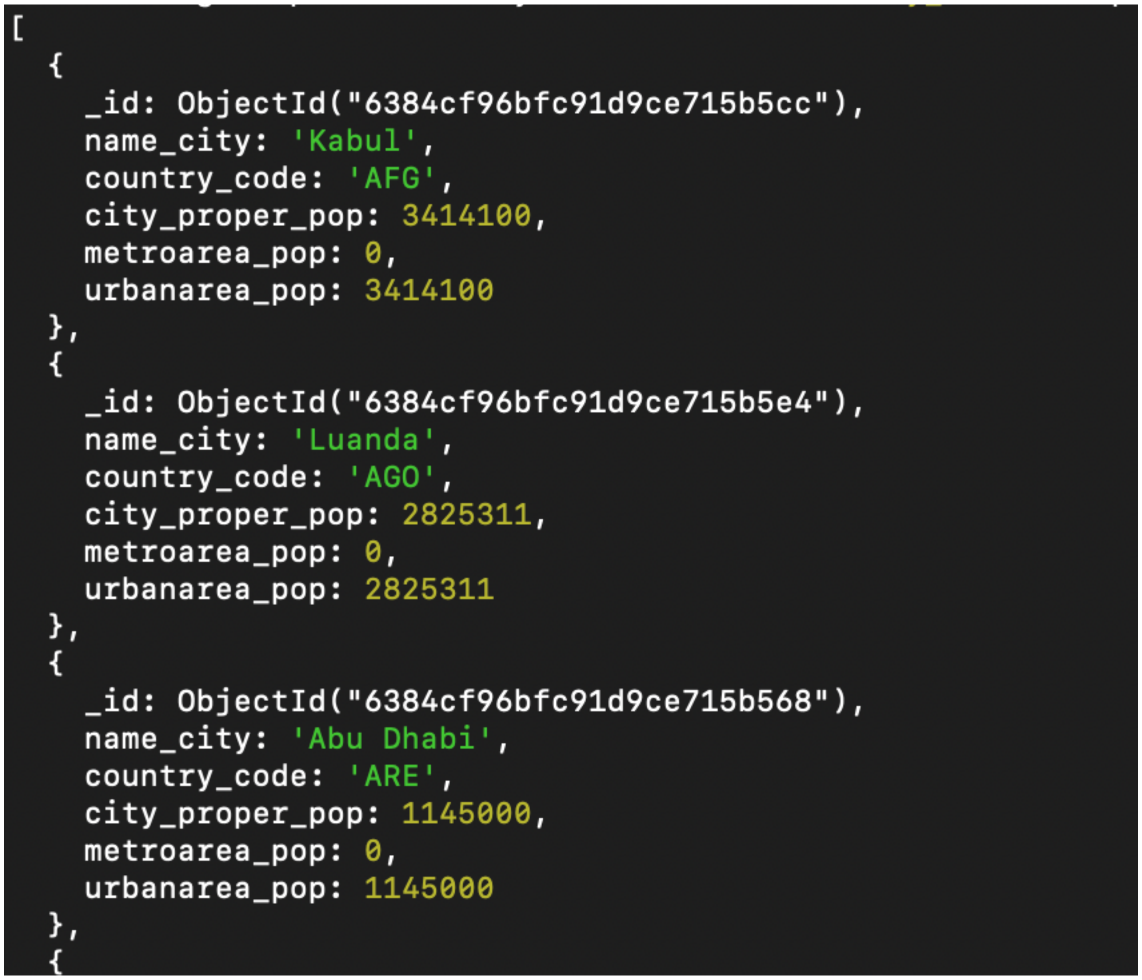
```
db.Economy.find({'income_group': 'High income'}).limit(2)
```



```
[
  {
    _id: ObjectId("6384d315bfc91d9ce715b807"),
    econ_id: 7,
    code: 'ARE',
    year: 2010,
    income_group: 'High income',
    gdp_percapita: 34628.63,
    gross_savings: 27.073,
    inflation_rate: 0.878,
    total_investment: 27.372,
    unemployment_rate: 0,
    exports: 3.843,
    imports: -0.981
  },
  {
    _id: ObjectId("6384d315bfc91d9ce715b808"),
    econ_id: 8,
    code: 'ARE',
    year: 2015,
    income_group: 'High income',
    gdp_percapita: 38649.91,
    gross_savings: 34.106,
    inflation_rate: 4.07,
    total_investment: 27.477,
    unemployment_rate: 0,
    exports: 7.32,
    imports: 2.17
  }
]
```

Q4) Display the City collection after sorting the city by country code in ascending order.

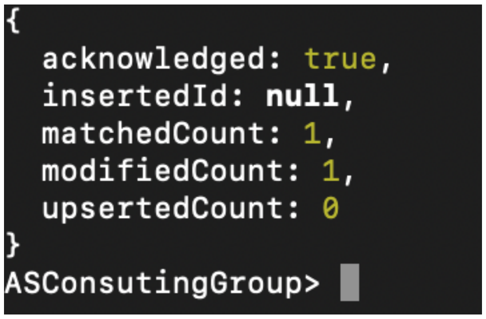
```
ASConsutingGroup> db.newcity.find().sort({country_code:1}).pretty()
```



```
[
  {
    _id: ObjectId("6384cf96bfc91d9ce715b5cc"),
    name_city: 'Kabul',
    country_code: 'AFG',
    city_proper_pop: 3414100,
    metroarea_pop: 0,
    urbanarea_pop: 3414100
  },
  {
    _id: ObjectId("6384cf96bfc91d9ce715b5e4"),
    name_city: 'Luanda',
    country_code: 'AGO',
    city_proper_pop: 2825311,
    metroarea_pop: 0,
    urbanarea_pop: 2825311
  },
  {
    _id: ObjectId("6384cf96bfc91d9ce715b568"),
    name_city: 'Abu Dhabi',
    country_code: 'ARE',
    city_proper_pop: 1145000,
    metroarea_pop: 0,
    urbanarea_pop: 1145000
  },
  {
```

Q5) Display the result of updating the population collection with population id number 19 by increasing its population size by 100.

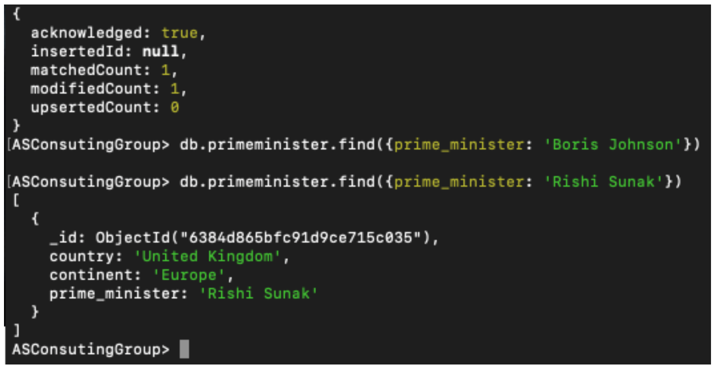
```
ASConsutingGroup> db.population.update({pop_id: 19}, {$inc:{ size:
100}})
```



```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
ASConsutingGroup> 
```

Q6) Update the prime minister collection by adding the new Prime Minister of the United Kingdom, Rishi Sunak, and display the result.

```
ASConsutingGroup> db.primeminister.update({prime_minister: 'Boris Johnson'}, {$set: {'country': 'United Kingdom', 'continent': 'Europe', 'prime_minister': 'Rishi Sunak'}})
```

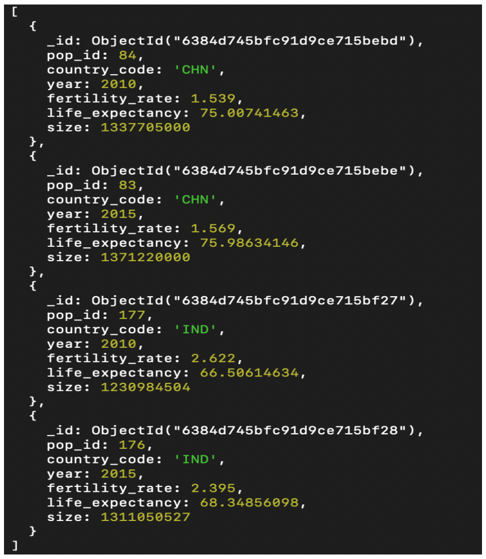


```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
ASConsutingGroup> db.primeminister.find({prime_minister: 'Boris Johnson'})

ASConsutingGroup> db.primeminister.find({prime_minister: 'Rishi Sunak'})
[
  {
    _id: ObjectId("6384d865bfc91d9ce715c035"),
    country: 'United Kingdom',
    continent: 'Europe',
    prime_minister: 'Rishi Sunak'
  }
]
ASConsutingGroup>
```

Q7) Find a population summary of size more than 400 million people.

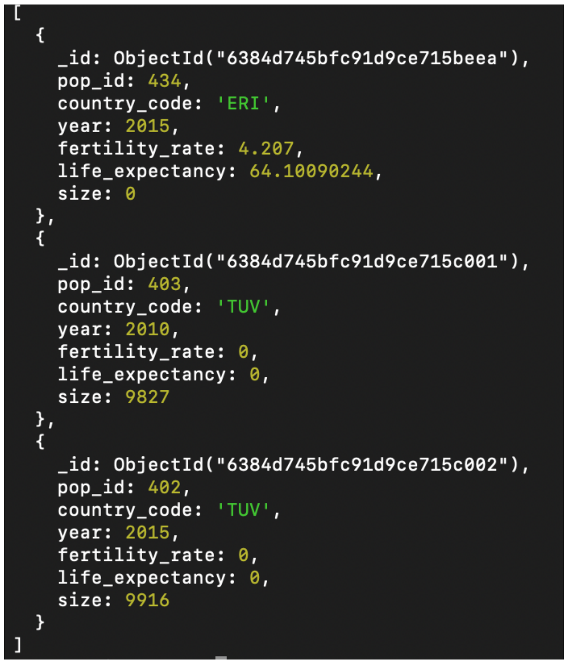
```
ASConsutingGroup> db.population.find({size: {$gt: 400000000}})
```



```
[
  {
    _id: ObjectId("6384d745bfc91d9ce715bebd"),
    pop_id: 84,
    country_code: 'CHN',
    year: 2010,
    fertility_rate: 1.539,
    life_expectancy: 75.00741463,
    size: 1337705000
  },
  {
    _id: ObjectId("6384d745bfc91d9ce715bebe"),
    pop_id: 83,
    country_code: 'CHN',
    year: 2015,
    fertility_rate: 1.569,
    life_expectancy: 75.98634146,
    size: 1371220000
  },
  {
    _id: ObjectId("6384d745bfc91d9ce715bf27"),
    pop_id: 177,
    country_code: 'IND',
    year: 2010,
    fertility_rate: 2.622,
    life_expectancy: 66.50614634,
    size: 1230984504
  },
  {
    _id: ObjectId("6384d745bfc91d9ce715bf28"),
    pop_id: 176,
    country_code: 'IND',
    year: 2015,
    fertility_rate: 2.395,
    life_expectancy: 68.34856098,
    size: 1311050527
  }
]
```

Q8) Find a population summary of size less than 10000 people.

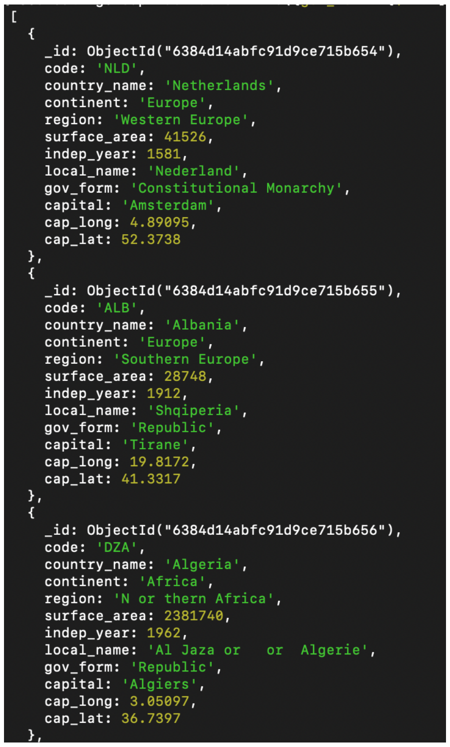
```
ASConsutingGroup> db.population.find({size: {$lt: 10000}})
```



```
[
  {
    _id: ObjectId("6384d745bfc91d9ce715beea"),
    pop_id: 434,
    country_code: 'ERI',
    year: 2015,
    fertility_rate: 4.207,
    life_expectancy: 64.10090244,
    size: 0
  },
  {
    _id: ObjectId("6384d745bfc91d9ce715c001"),
    pop_id: 403,
    country_code: 'TUV',
    year: 2010,
    fertility_rate: 0,
    life_expectancy: 0,
    size: 9827
  },
  {
    _id: ObjectId("6384d745bfc91d9ce715c002"),
    pop_id: 402,
    country_code: 'TUV',
    year: 2015,
    fertility_rate: 0,
    life_expectancy: 0,
    size: 9916
  }
]
```

Q9) List the countries which has government form Constitutional Monarchy and Republic

```
ASConsutingGroup> db.Nation.find({gov_form: {$in: ['Constitutional Monarchy', 'Republic']}})
```

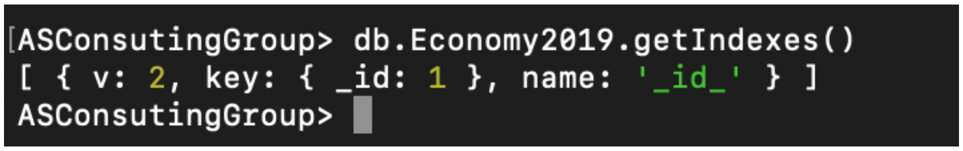


```
[
  {
    _id: ObjectId("6384d14abfc91d9ce715b654"),
    code: 'NLD',
    country_name: 'Netherlands',
    continent: 'Europe',
    region: 'Western Europe',
    surface_area: 41526,
    indep_year: 1581,
    local_name: 'Nederland',
    gov_form: 'Constitutional Monarchy',
    capital: 'Amsterdam',
    cap_long: 4.89095,
    cap_lat: 52.3738
  },
  {
    _id: ObjectId("6384d14abfc91d9ce715b655"),
    code: 'ALB',
    country_name: 'Albania',
    continent: 'Europe',
    region: 'Southern Europe',
    surface_area: 28748,
    indep_year: 1912,
    local_name: 'Shqiperia',
    gov_form: 'Republic',
    capital: 'Tirane',
    cap_long: 19.8172,
    cap_lat: 41.3317
  },
  {
    _id: ObjectId("6384d14abfc91d9ce715b656"),
    code: 'DZA',
    country_name: 'Algeria',
    continent: 'Africa',
    region: 'N or thern Africa',
    surface_area: 2381740,
    indep_year: 1962,
    local_name: 'Al Jaza or    or  Algerie',
    gov_form: 'Republic',
    capital: 'Algiers',
    cap_long: 3.05097,
    cap_lat: 36.7397
  },
```

Q10) Check the index on the Economy2019 collection

```
ASConsutingGroup> db.Economy2019.getIndexes()
```

```
Index is applied on ID
```

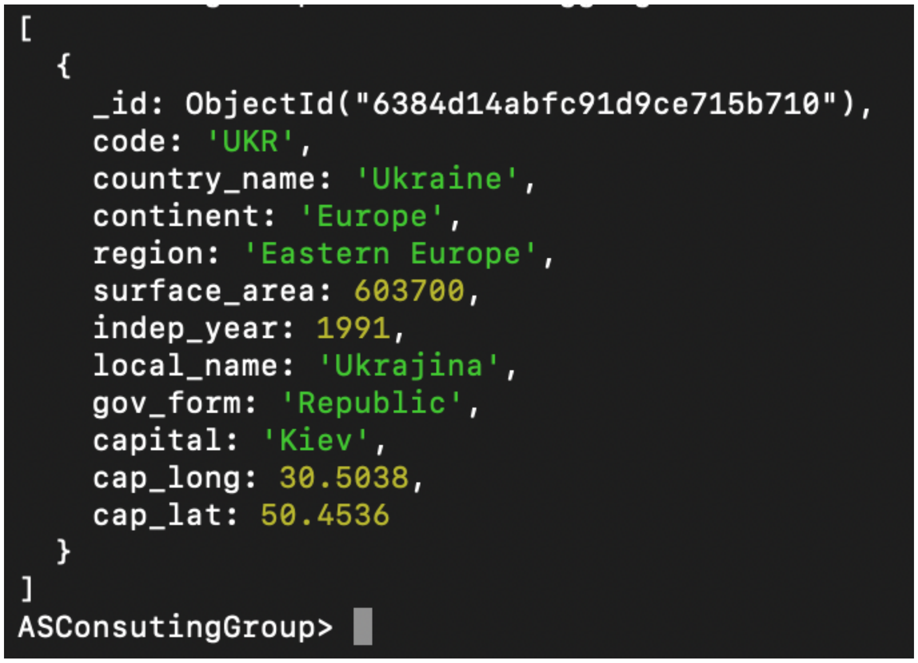


```
[ASConsutingGroup> db.Economy2019.getIndexes()
[ { v: 2, key: { _id: 1 }, name: '_id_' } ]
ASConsutingGroup>
```

Aggregation Pipeline

Q11) Find a country with a government in the form of a republic, on the European continent, with a land area greater than 600,000 square feet.

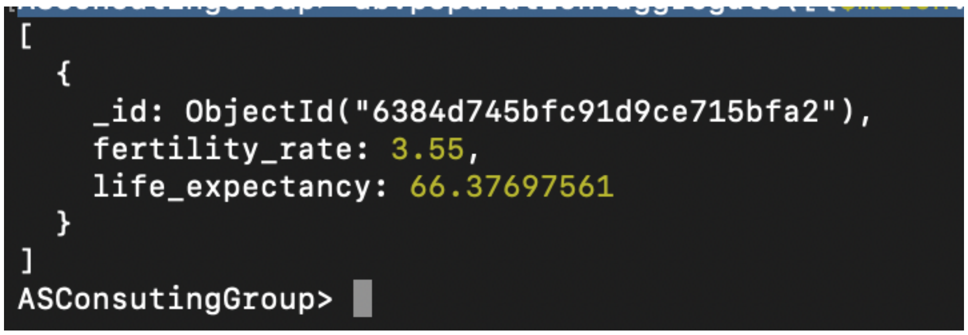
```
ASConsutingGroup> db.Nation.aggregate([{"$match": {$and:[{'gov_form':
'Republic'}, {'continent':'Europe'}, {'surface_area': {'$gt':
600000}}]}}])
```



```
[
  {
    _id: ObjectId("6384d14abfc91d9ce715b710"),
    code: 'UKR',
    country_name: 'Ukraine',
    continent: 'Europe',
    region: 'Eastern Europe',
    surface_area: 603700,
    indep_year: 1991,
    local_name: 'Ukrajina',
    gov_form: 'Republic',
    capital: 'Kiev',
    cap_long: 30.5038,
    cap_lat: 50.4536
  }
]
ASConsutingGroup>
```

Q12) For year 2015, Display the fertility rate and life expectancy for the country code Pakistan.

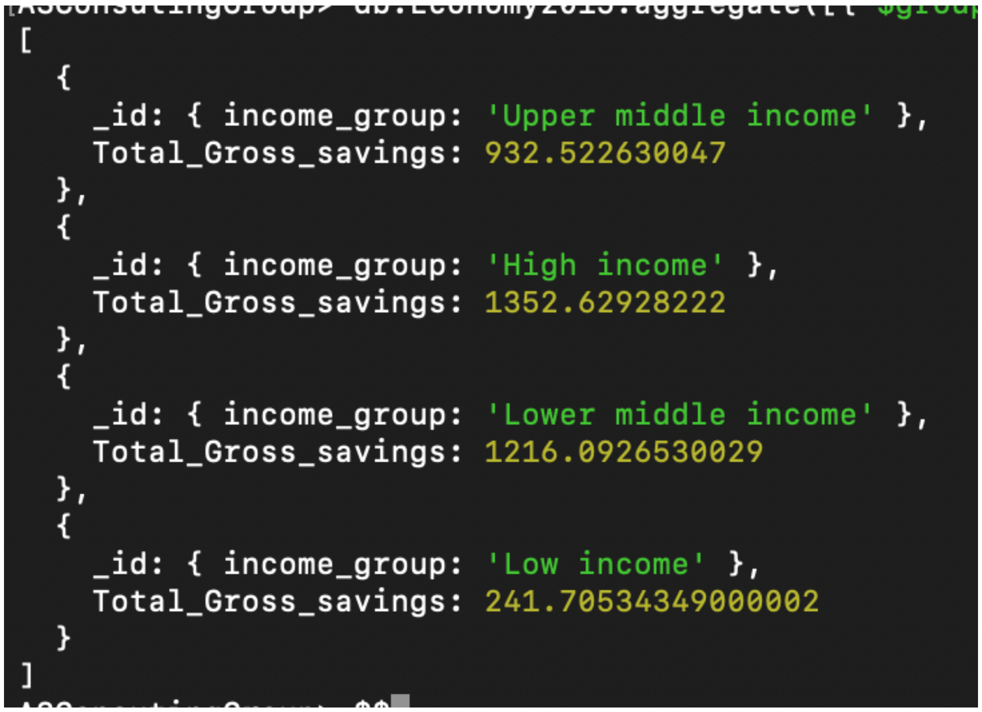
```
ASConsutingGroup> db.population.aggregate([{$match:
{$and:[{'country_code': 'PAK'}, {'year': 2015}]}},{'$project':
{'fertility_rate': 1, 'life_expectancy':1}}])
```



```
[
  {
    _id: ObjectId("6384d745bfc91d9ce715bfa2"),
    fertility_rate: 3.55,
    life_expectancy: 66.37697561
  }
]
ASConsutingGroup> 
```

Q13) Show the total gross savings of each economic group

```
ASConsutingGroup> db.Economy2015.aggregate([{"$group": {'_id':
{'income_group': '$income_group'}, 'Total_Gross_savings': {'$sum':
'$gross_savings'}}}])
```



```
[
  {
    _id: { income_group: 'Upper middle income' },
    Total_Gross_savings: 932.522630047
  },
  {
    _id: { income_group: 'High income' },
    Total_Gross_savings: 1352.62928222
  },
  {
    _id: { income_group: 'Lower middle income' },
    Total_Gross_savings: 1216.0926530029
  },
  {
    _id: { income_group: 'Low income' },
    Total_Gross_savings: 241.70534349000002
  }
]
```

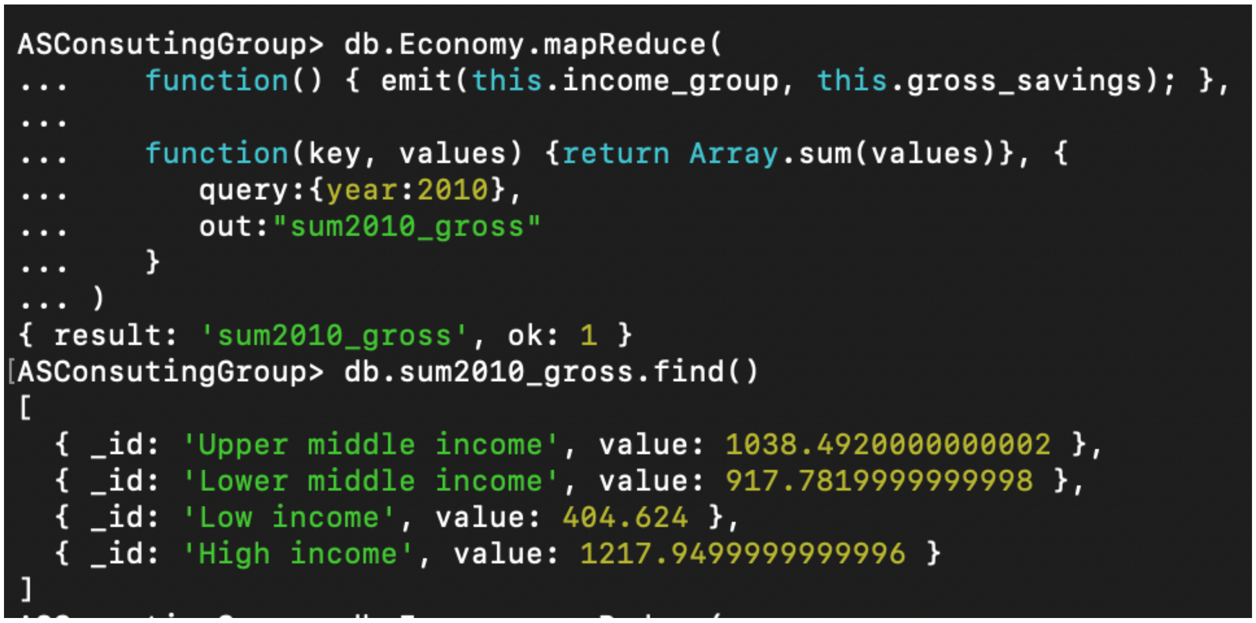
{ MAP REDUCE PIPELINE }

Q14) Display the total gross savings of each economic group for the 2010 year.

```
>db.Economy.mapReduce(
   function() { emit(this.income_group, this.gross_savings); },

   function(key, values) {return Array.sum(values)}, {
      query:{year:2010},
      out:"sum2010_gross"
   }
)

db.sum2010_gross.find()
```



```
ASConsutingGroup> db.Economy.mapReduce(
...    function() { emit(this.income_group, this.gross_savings); },
...
...    function(key, values) {return Array.sum(values)}, {
...       query:{year:2010},
...       out:"sum2010_gross"
...    }
... )
{ result: 'sum2010_gross', ok: 1 }
ASConsutingGroup> db.sum2010_gross.find()
[
  { _id: 'Upper middle income', value: 1038.4920000000002 },
  { _id: 'Lower middle income', value: 917.7819999999998 },
  { _id: 'Low income', value: 404.624 },
  { _id: 'High income', value: 1217.9499999999996 }
]
```

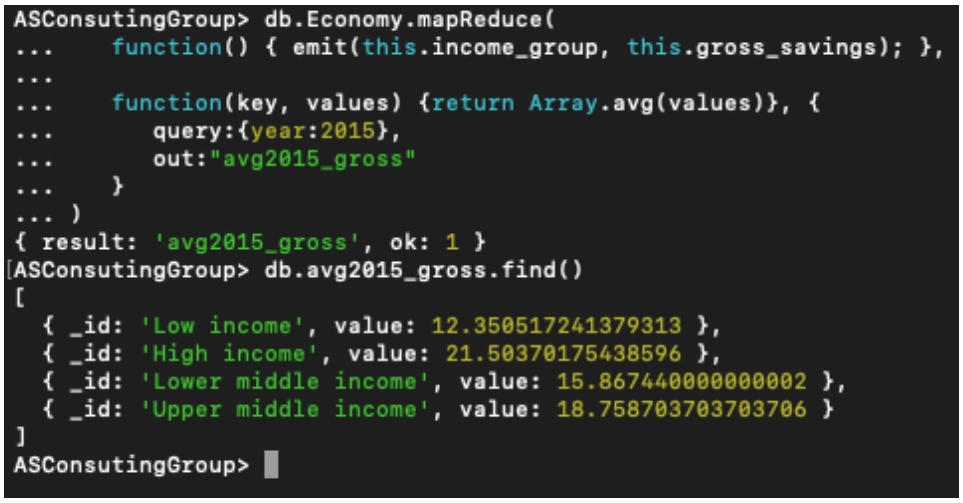
Q15) Display the Average gross savings of each economic group for the 2015 year.

```
>db.Economy.mapReduce(
  function() { emit(this.income_group, this.gross_savings); },

  function(key, values) {return Array.avg(values)}, {
    query:{year:2015},
    out:"avg2015_gross"
  }
)
```

```
ASConsutingGroup> db.Economy.mapReduce(
...     function() { emit(this.income_group, this.gross_savings); },
...
...     function(key, values) {return Array.avg(values)}, {
...        query:{year:2015},
...        out:"avg2015_gross"
...     }
... )
{ result: 'avg2015_gross', ok: 1 }
```

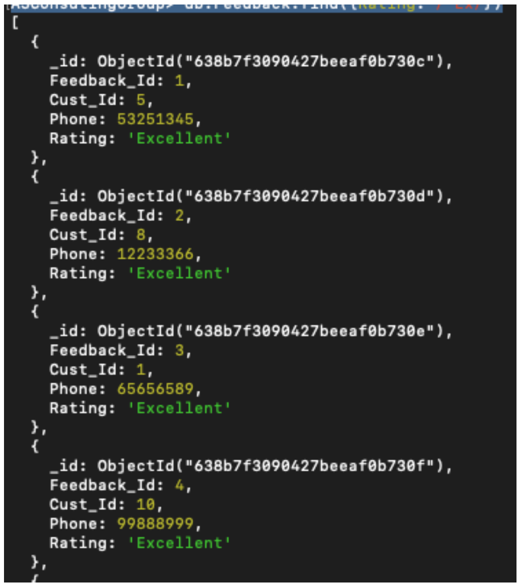
```
db.avg2015_gross.find()
```



```
ASConsutingGroup> db.Economy.mapReduce(
...     function() { emit(this.income_group, this.gross_savings); },
...
...     function(key, values) {return Array.avg(values)}, {
...        query:{year:2015},
...        out:"avg2015_gross"
...     }
... )
{ result: 'avg2015_gross', ok: 1 }
ASConsutingGroup> db.avg2015_gross.find()
[
  { _id: 'Low income', value: 12.350517241379313 },
  { _id: 'High income', value: 21.50370175438596 },
  { _id: 'Lower middle income', value: 15.867440000000002 },
  { _id: 'Upper middle income', value: 18.758703703703706 }
]
ASConsutingGroup>
```

Q16) Find the result of customer collection whose rating begins with 'Ex' to discover about the Excellent customer details.

```
ASConsutingGroup> db.Feedback.find({Rating: /^Ex/})
```



```
[
  {
    _id: ObjectId("638b7f3090427beeaf0b730c"),
    Feedback_Id: 1,
    Cust_Id: 5,
    Phone: 53251345,
    Rating: 'Excellent'
  },
  {
    _id: ObjectId("638b7f3090427beeaf0b730d"),
    Feedback_Id: 2,
    Cust_Id: 8,
    Phone: 12233366,
    Rating: 'Excellent'
  },
  {
    _id: ObjectId("638b7f3090427beeaf0b730e"),
    Feedback_Id: 3,
    Cust_Id: 1,
    Phone: 65656589,
    Rating: 'Excellent'
  },
  {
    _id: ObjectId("638b7f3090427beeaf0b730f"),
    Feedback_Id: 4,
    Cust_Id: 10,
    Phone: 99888999,
    Rating: 'Excellent'
  },
```