# Aggregate

# Llamado

```
Model.aggregate(pipeline);
```

# Pipeline

```
const pipeline = [
  [pipeline stages]
]
```

# Pipeline Stages

- $project

- $skip

- $sort

- $limit

- $match

- $set / $addFields

- $count

- $lookup

- $unwind

# $project

{ $project: { : 0, : 0, ... } }

# $project

{ $skip: 100 }

# $sort

{ $sort: { age: 1 }}


1 ⇒ ascendente
-1 ⇒ descendente


Solo acepta 1 o -1.

# $limit

{ $limit: 20 }

# Ejemplo

```
const pipeline = [
  { $project: { name: 1, age: 1 } },
  { $sort: { age: 1 } },
  { $skip: 100 },
  { $limit: 20 },
];

await Users.aggregate(pipeline);
```

# $match

{ $match : { name: "john" } }

# $match

```
{
  $match: {
    $and: [{ age: { $gt: 16 } }, { licenseNumber: { $exists: true } }];
  }
}
```

# $set & $addFields

```
{ $set: { [newField]: [expression]} }
{ $addFields: { [newField]: [expression]} }
```

# $count

```
{ $count: }
```

# $lookup

```
{
  $lookup: {
    from: "mascotas",
    localField: "\_id",
    foreignField: "owner_id",
    as: "mis_mascotas"
  }
}
```

# $unwind

```
{
  $unwind: {
    path: [field path],
    includeArrayIndex: <string>,
    preserveNullAndEmptyArrays: <boolean> //default: false
  }
}
```

# Ejemplo Pipeline

```javascript
const pipeline = [
  { $project: { "name": 1, "age": 1} },
  { $sort: { age: 1 } },
  { $lookup: {
    from: "mascotas",
    localField: "_id",
    foreignField: "owner_id",
    as: "mis_mascotas"
  }},
  { $unwind: {
    path: "mis_mascotas"
  }}
  { $skip: 100 },
  { $limit: 20 },
];

const docs = await Users.aggregate(pipeline);
```

# Ejemplo Pipeline

```javascript
const pipeline = [
  { $project: { name: 1, age: 1 } },
  { $sort: { age: 1 } },
  { $skip: 100 },
  { $limit: 20 },
  {
    $lookup: {
      from: "mascotas",
      localField: "_id",
      foreignField: "owner_id",
      as: "mis_mascotas",
    },
  },
  {
    $unwind: {
      path: "mis_mascotas",
    },
  },
];

const docs = await Users.aggregate(pipeline);
```