## **Análisis de bases de datos Olist - ECommerce Brazil**

### **¿Cuál es la necesidad o problema a resolver?** 🚀

Queremos familiarizarnos con un ecommerce e investigar a las empresas que ya operan para usarlas como referencia. Queremos investigar qué métodos de pago se usan, cuánto tiempo tardan en entregar, cuál es el flujo de los productos, qué categorías se venden mejor, implicaciones que tienen el peso y las fotos de los productos con sus ventas, etc.

### **¿Cuál es la propuesta para resolver esa necesidad o problema?** 💡

Usar un dataset de la empresa Olist para analizar la información de sus clientes, vendedores y en general de su ECommerce, en Brazil, que es un país que se asemeja bastante a México.

### **¿Cómo lo vas a hacer?** 📋

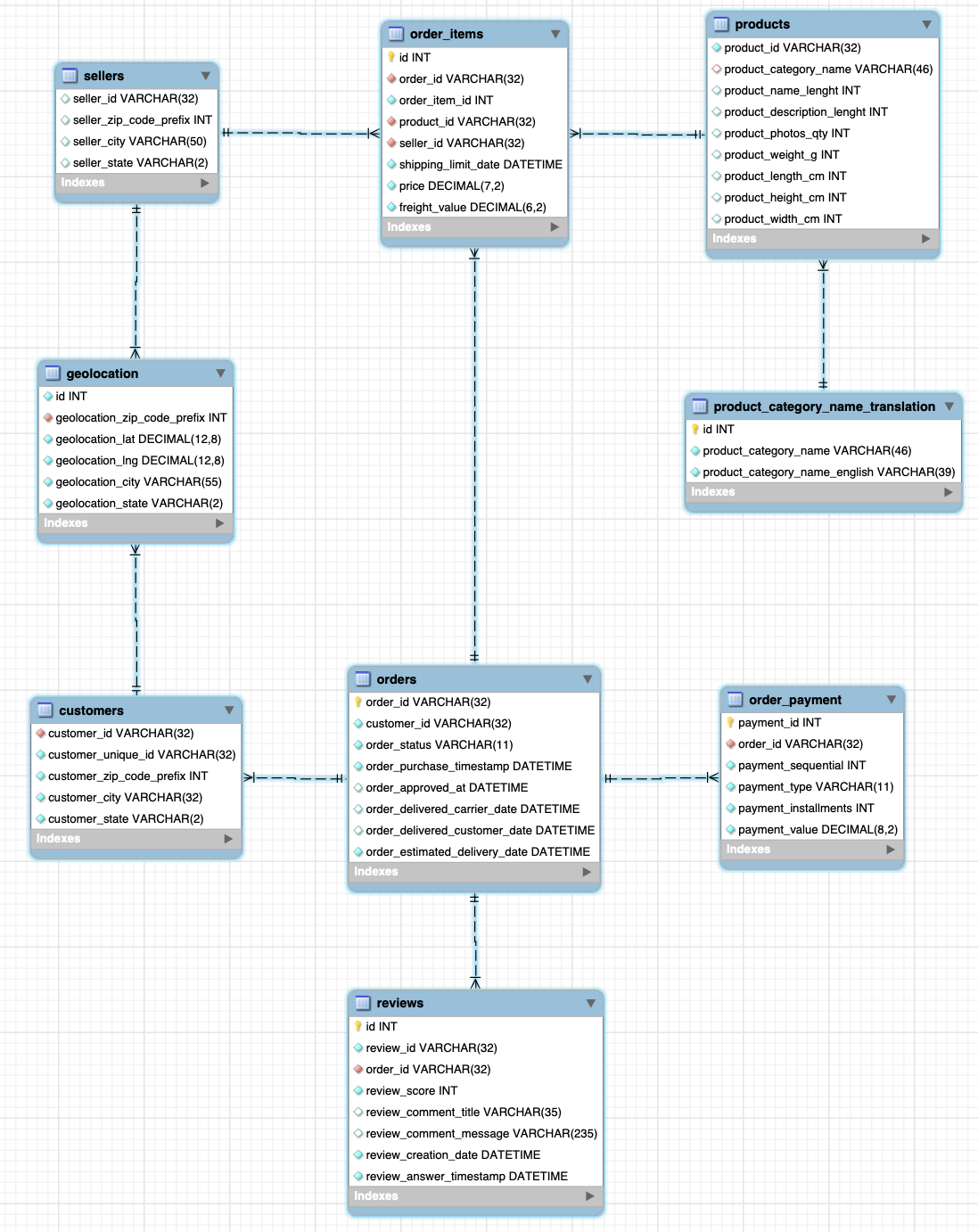
Analizaremos la información real de sus bases de datos para darnos una idea de cómo funciona y conocer mejor el negocio del comercio electrónico.

### **Estructura de la información**

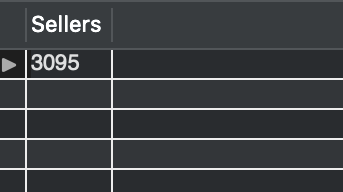
La base de datos cuenta con 9 tablas diferentes:

1. orders - contiene la siguiente información de las ordenes:
   1. order\_id - identificador de la orden hecho de caracteres y números
   2. customer\_id - identificador de cliente hecho de caracteres y números
   3. order\_status - estado de la orden (entregado, en tránsito, etc.)
   4. order\_purchase\_timestamp - fecha de compra
   5. order\_approved\_at - fecha en que se aprobó la orden
   6. order\_delivered\_carrier\_date - fecha en que se surtió orden a courier
   7. order\_delivered\_customer\_date - fecha de entrega a cliente
   8. order\_estimated\_delivery\_date - fecha estimada de entrega
2. order\_items
   1. id - llave primaria autoincrementable
   2. order\_id - identificador de la orden hecho de caracteres y números
   3. order\_item\_id - número de ítem dentro de la misma orden
   4. product\_id - identificador de producto hecho de caracteres y números
   5. seller\_id - identificador de vendedor hecho de caracteres y números
   6. shipping\_limit\_date - fecha límite de envío para cumplir con fecha de entrega estimada
   7. price - precio del ítem
   8. freight\_value - valor del envío
3. customers
   1. customer\_id - identificador de cliente hecho de caracteres y números
   2. customer\_unique\_id - identificador único de cliente hecho de caracteres y números (customer\_id también es único)
   3. customer\_zip\_code\_prefix - prefijo de código postal
   4. customer\_city - ciudad en donde se registró el usuario
   5. customer\_state - estado en donde se registró el usuario
4. order\_payment
   1. payment\_id - identificador del pago hecho de caracteres y números
   2. order\_id - identificador de la orden hecho de caracteres y números
   3. payment\_sequential - NA
   4. payment\_type - tipo de pago (tarjeta de crédito, débito, etc.)
   5. payment\_installments - número de pagos
   6. payment\_value - valor de cada pago
5. reviews
   1. id - llave primaria autoincrementable
   2. review\_id - identificador de la reseña hecho de caracteres y números
   3. order\_id - identificador de la orden hecho de caracteres y números
   4. review\_score - calificación asignada por usuario (de 1 a 5)
   5. review\_comment\_title - título de review (puede quedar vacío)
   6. review\_comment\_message - mensaje de reseña (puede quedar vacío)
   7. review\_creation\_date - fecha de creación de reseña
   8. review\_answer\_timestamp - fecha de respuesta de reseña
6. sellers
   1. seller\_id - identificador del vendedor hecho de caracteres y números
   2. seller\_zip\_code\_prefix - prefijo de código postal
   3. seller\_city - ciudad en donde se registró el vendedor
   4. seller\_state - estado en donde se registró el vendedor
7. geolocation
   1. id - llave primario autoincrementable
   2. geolocation\_zip\_code\_prefix - prefijo de código postal
   3. geolocation\_lat - coordenadas (longitud)
   4. geolocation\_lon - coordenadas (latitud)
   5. geolocation\_city - ciudad a la que pertenecen coordenadas
   6. geolocation\_state - estado al que pertenecen coordenadas
8. products
   1. product\_id - identificador de producto hecho de caracteres y números
   2. product\_category\_name - nombre de categoría de producto (en portugués)  
      product\_name\_length - longitud de nombre de producto
   3. product\_description\_length - longitud de descripción de producto
   4. product\_photos\_qty - cantidad de fotos asignadas al producto
   5. product\_weight\_g - peso del producto en gramos
   6. product\_length\_cm - largo del producto en cm
   7. product\_height\_cm - alto del producto en cm
   8. product\_width\_cm - ancho del producto en cm
9. product\_category\_name\_translation
   1. id - llave primario autoincrementable
   2. product\_category\_name - nombre de categoría de producto (en portugués)
   3. product\_category\_name\_translation - traducción de nombre de categoría a inglés.

A continuación en el esquema se grafican las relaciones entre las tablas:



### **Preguntas**

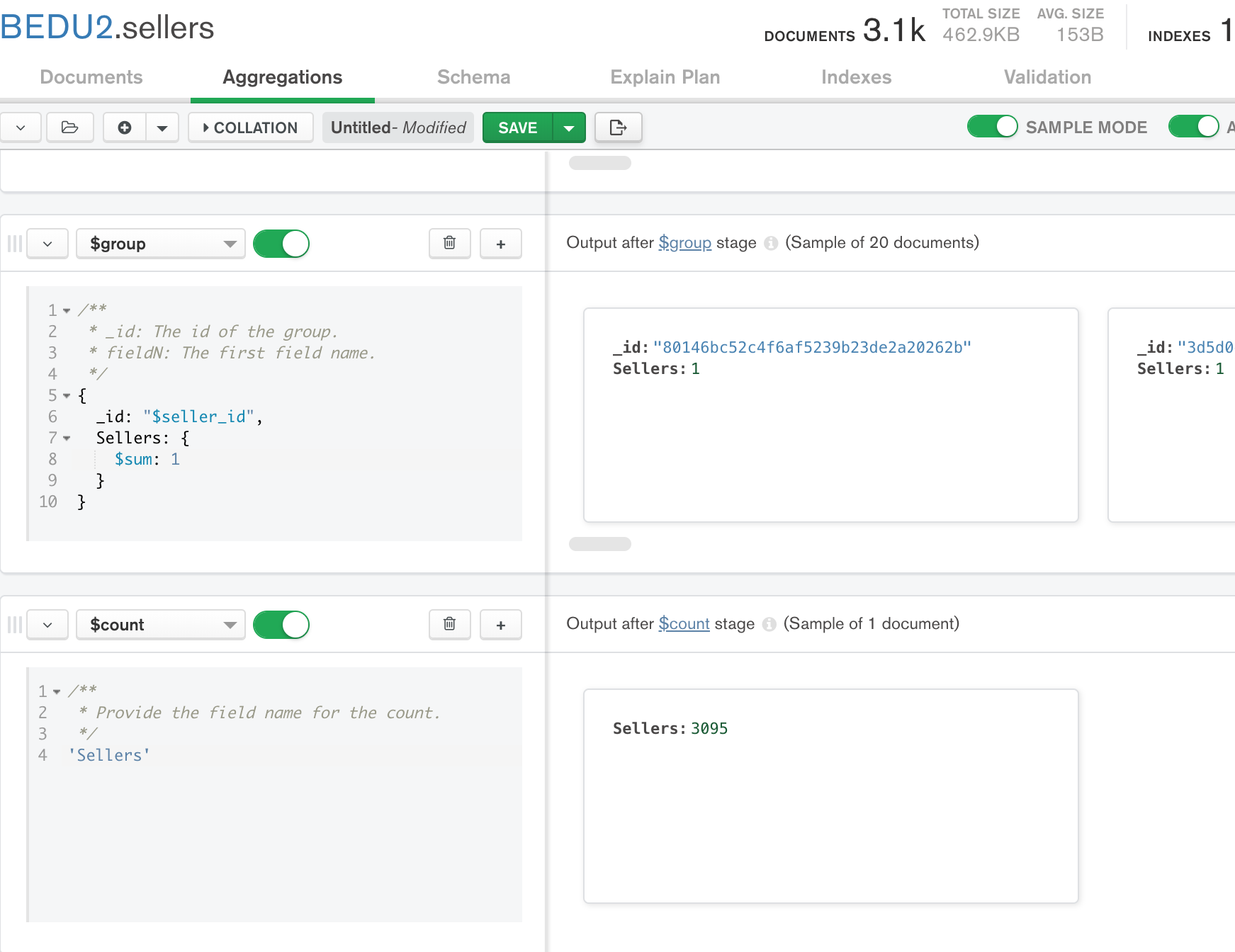
1. ¿Cuántos distintos sellers hay registrados?
   1. MYSQL  
      SELECT COUNT(DISTINCT(seller\_id)) FROM sellers;  
      
   2. MongoDB  
      [{$group: {

\_id: "$seller\_id",

Sellers: {

$sum: 1

}

}},   
 {$count: 'Sellers'  
}]  


1. ¿Cuáles son las distintas locaciones (estados) de los Sellers?
   1. MySQL  
      SELECT DISTINCT(seller\_city), seller\_state FROM sellers

ORDER BY seller\_city ASC;  


* 1. MongoDB  
     [{

$group: {

\_id: {  
 city: "$seller\_city", state: "$seller\_state"}

}  
}, {  
 $addFields: {

City: "$\_id.city",

State: "$\_id.state"

}  
}, {$project: {

City: 1,

State: 1,

\_id: 0

}  
}, {$sort: {

City: 1

} }, {$match: {

City: {"$ne": "04482255"}

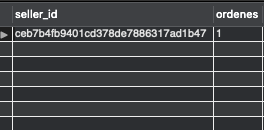
}  
}]  


1. ¿Cuántas órdenes se hicieron por medio del seller registrado en una ciudad invalida?
   1. MySQL  
      SELECT seller\_id, COUNT(order\_id) AS ordenes

FROM order\_items

WHERE seller\_id =

(SELECT seller\_id FROM sellers WHERE seller\_city = '04482255')

GROUP BY seller\_id;  


* 1. MongoDB  
     [{$lookup: {

from: 'sellers',

localField: 'seller\_id',

foreignField: 'seller\_id',

as: 'city\_array'

}}, {$addFields: {

city\_object: { $arrayElemAt: ["$city\_array", 0] }

}}, {$addFields: {

seller\_city: "$city\_object.seller\_city"

}}, {$project: {

seller\_city: 1,

order\_id: 1,

seller\_id: 1,

}}, {$match: {

seller\_city: '04482255'

}}, {$group: {

\_id: "$seller\_city",

ordenes: {

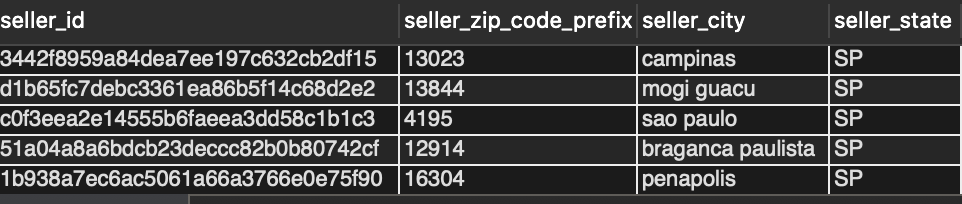
$sum: 1

}

}}]



1. ¿Quiénes son los sellers de Sao Paulo (SP)?
   1. MySQL  
      SELECT \* FROM sellers

WHERE seller\_state like "SP";  


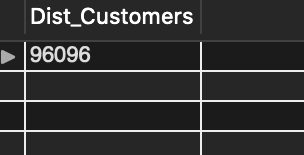
* 1. MongoDB  
     db.sellers.find(

{seller\_state:'SP'},

{seller\_id:1,seller\_state:1}

)  


1. ¿Cuantos customers distintos hay?
   1. MySQL  
      SELECT COUNT(DISTINCT(customer\_unique\_id)) AS Dist\_Customers

FROM customers;  


* 1. MongoDB  
     [{$group: {

\_id: "$customer\_unique\_id",

Dist\_Customers: {

$sum: 1

}

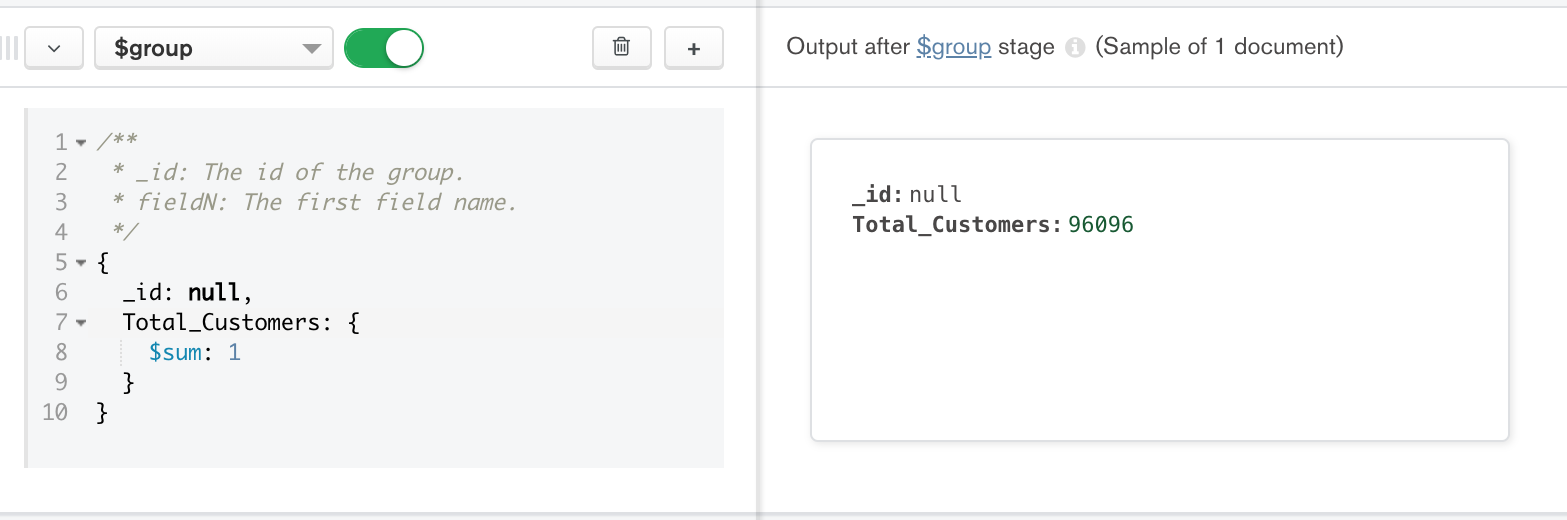
}}, {$group: {

\_id: null,

Total\_Customers: {

$sum: 1

}

}}]  


1. ¿De dónde son los customers (estados)?
   1. MySQL  
      SELECT DISTINCT(customer\_city), customer\_state

FROM customers

ORDER BY customer\_state;



* 1. MongoDB  
     [{$group: {

\_id: "$customer\_city",

Dist\_Cities: {

$sum: 1

}

}}, {$lookup: {

from: 'customers',

localField: '\_id',

foreignField: 'customer\_city',

as: 'string'

}}, {$addFields: {

state\_obj: {$arrayElemAt: ["$string", 0]}

}}, {$addFields: {

State: "$state\_obj.customer\_state"

}}, {$project: {

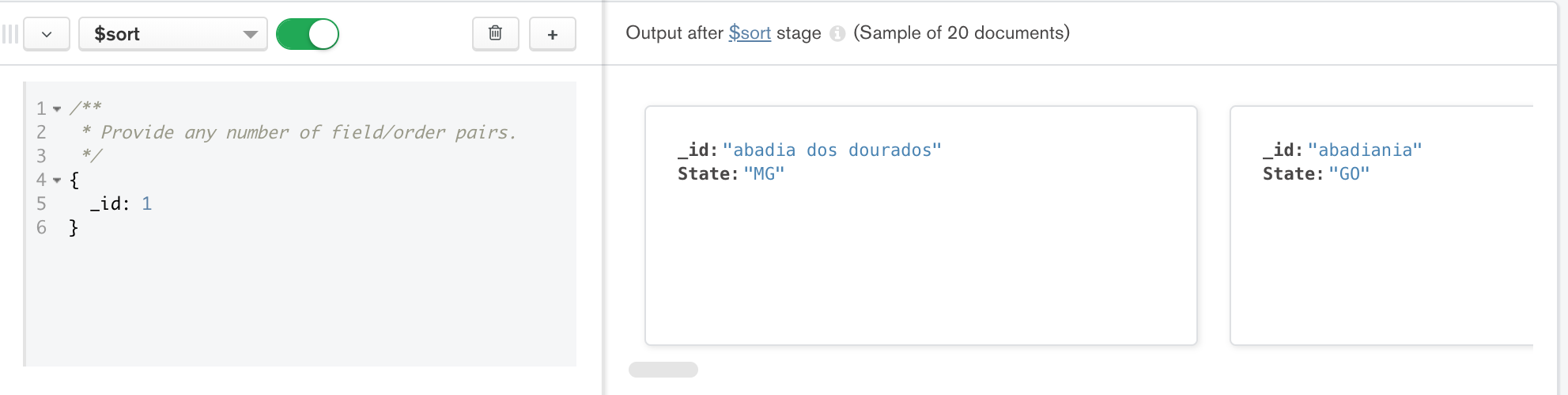
\_id: 1,

State: 1

}}, {$sort: {

\_id: 1

}}]

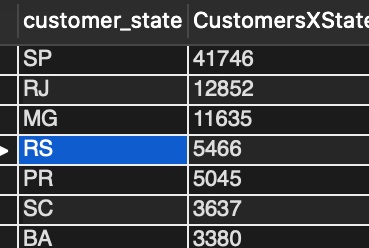


1. ¿Cuántos usuarios hay de cada estado?
   1. MySQL  
      SELECT customer\_state, COUNT(\*) AS CustomersXState

FROM customers

GROUP BY customer\_state

ORDER BY 2 DESC;



* 1. MongoDB  
     [{$group: {

\_id: {customer: "$customer\_id", state: "$customer\_state"}

}}, {$unwind: {

path: "$\_id"

}}, {$addFields: {

customer: "$\_id.customer",

state: "$\_id.state"

}}, {$group: {

\_id: "$state",

customerXstate: {

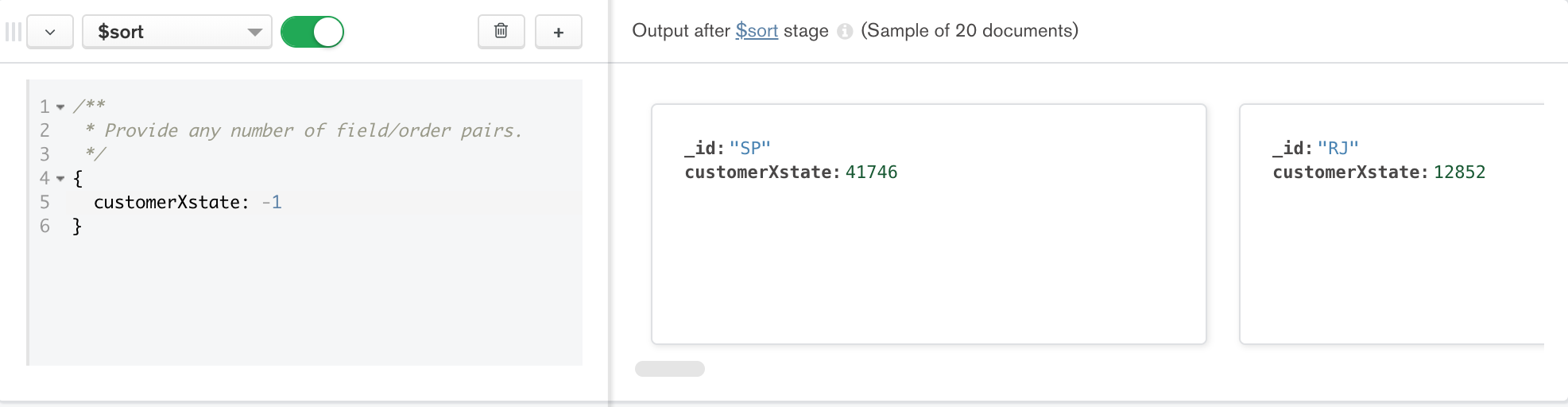
$sum: 1

}

}}, {$sort: {

customerXstate: -1

}}]



1. ¿Cuáles son los distintos métodos de pago y cuál es su share general?
   1. MySQL

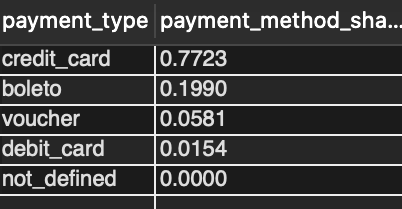
SELECT DISTINCT(payment\_type),

COUNT(\*)/(SELECT COUNT(DISTINCT(order\_id)) FROM order\_payment) AS payment\_method\_share

FROM order\_payment

GROUP BY payment\_type

ORDER BY payment\_method\_share DESC;



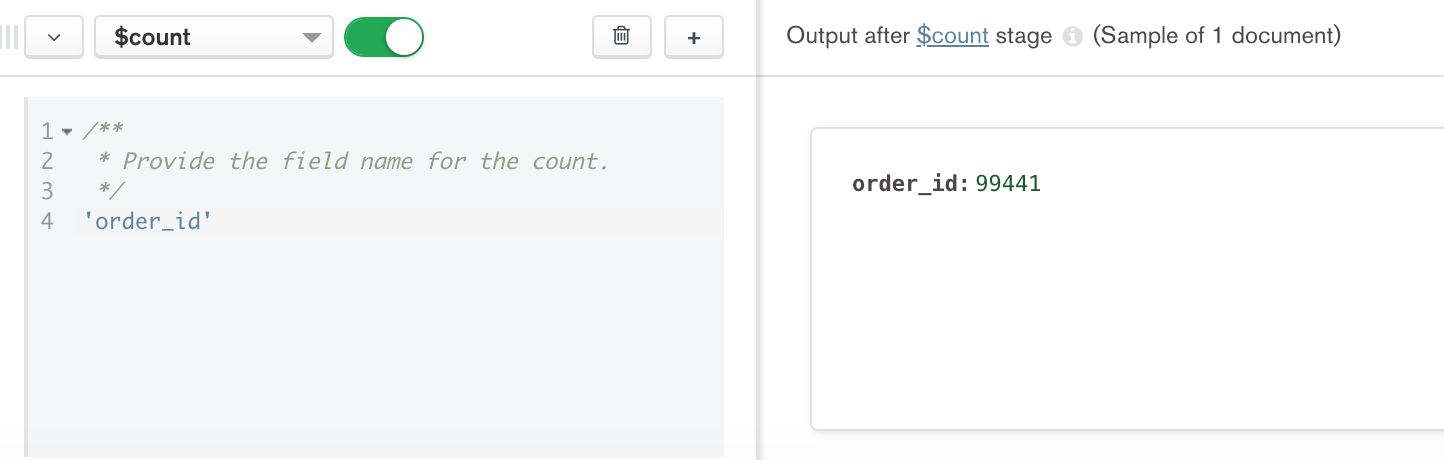
* 1. MongoDB  
     Para realizar el problema en MongoDB, tuve que dividirlo en 2 partes. Primero calculé el número total de órdenes distintas y después ya calculé el share.  
       
     Número total de órdenes:  
     [{$group: {

\_id: "$order\_id",

ordenes: {

$sum: 1

}

}}, {$count: 'order\_id'}]  


Cálculo de share:

[{$group: {

\_id: "$payment\_type",

ordenes: {

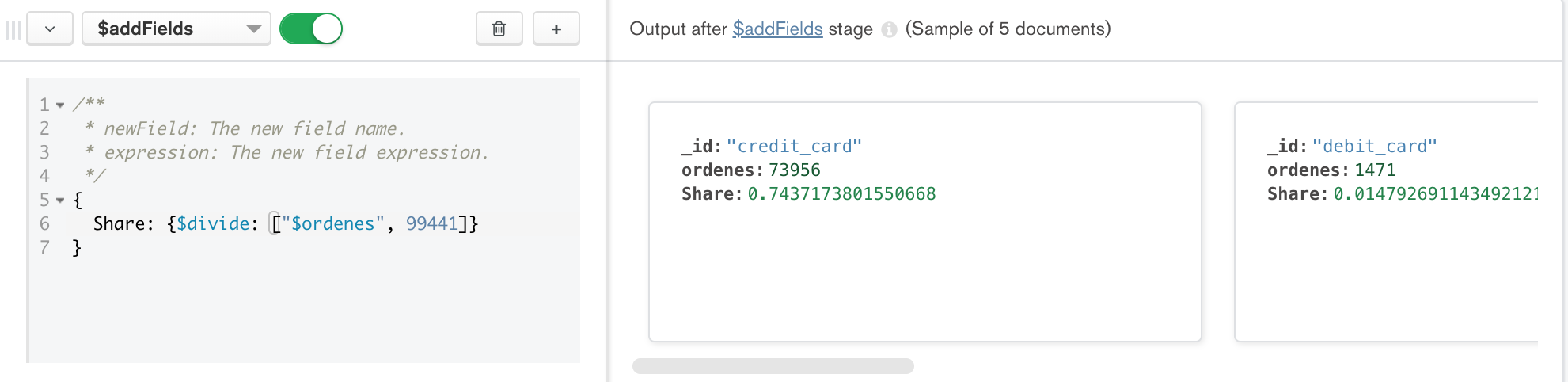
$sum: 1

}

}}, {$addFields: {

Share: {$divide: ["$ordenes", 99441]}

}}]



1. ¿Cuántas órdenes hay por mes?
   1. MySQL  
      SELECT MONTH(order\_purchase\_timestamp) AS Month, YEAR(order\_purchase\_timestamp) AS Year, COUNT(\*) AS Sales

FROM orders

GROUP BY 1, 2

ORDER BY 2 ASC, 1 DESC;



* 1. MongoDB  
     En MongoDB no encontré la manera de quitar duplicados de las órdenes y al mismo tiempo agruparlas, pero sí encontré cómo contar las órdenes por año usando $Buckets y se me hizo interesante:

[{$addFields: {

Month: {$month:["$order\_purchase\_timestamp"]},

Year: {$year:["$order\_purchase\_timestamp"]}

}}, {$addFields: {

formated\_date: { $dateFromParts: { 'year' : "$Year", 'month' : "$Month", 'day': 1} },

Month\_Year: {$concat: [{ $toString: "$Year"},".",{ $toString: "$Month" }]}

}}, {$bucket: {

groupBy: "$Year",

boundaries:[2016, 2017, 2018, 2019] ,

default: "Other",

output: {

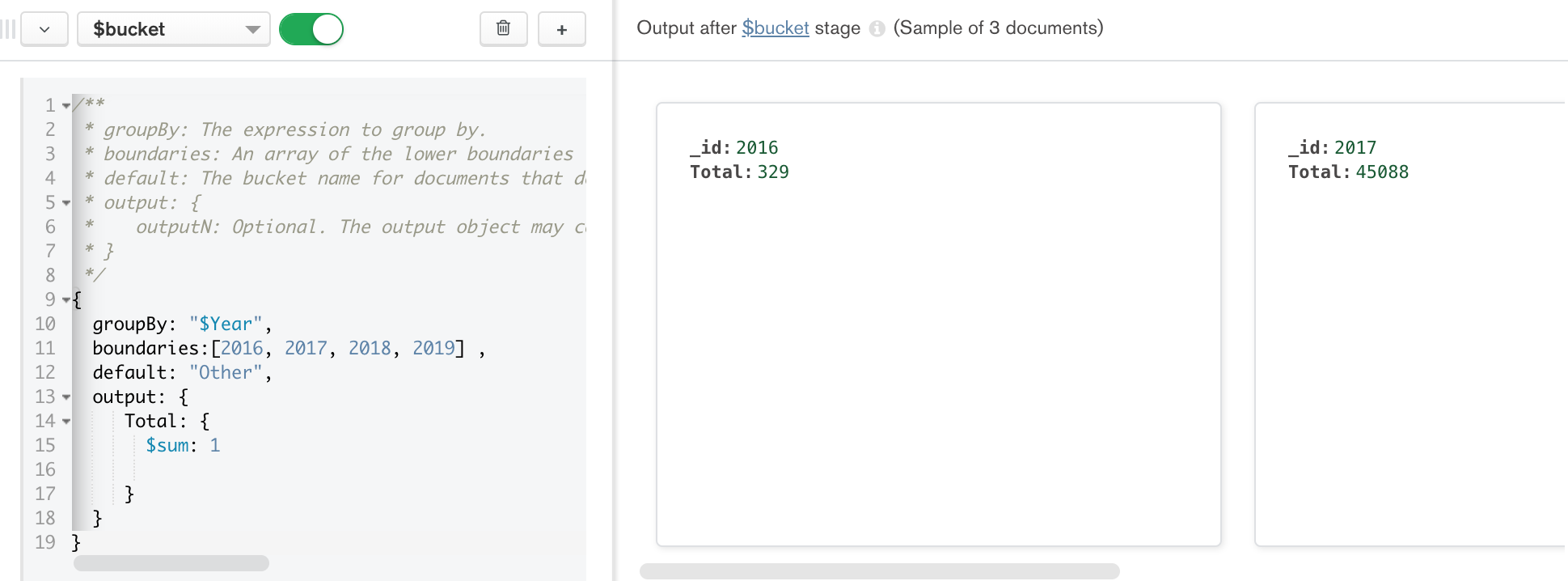
Total: {

$sum: 1

}

}

}}]

  
  
En teoría con esto ya debería de salir la opción de órdenes por mes pero MongoDB no logra sacar el output. Usé $addToSet en vez de $push para quitar duplicados:  
  
[{$group: {

\_id: "$order\_id",

ordenes: {

$addToSet: "$order\_id"

}

}}, {$project: {

ordenes: 1,

\_id: 0

}}, {$unwind: {

path: "$ordenes"

}}, {$lookup: {

from: 'orders',

localField: 'ordenes',

foreignField: 'order\_id',

as: 'order\_array'

}}, {$addFields: {

order\_obj: {$arrayElemAt: ["$order\_array",0]}

}}, {$addFields: {

order\_date: "$order\_obj.order\_purchase\_timestamp"

}}, {$addFields: {

month: {$month: ["$order\_date"]},

year: {$year: ["$order\_date"]}

}}, {$addFields: {

new\_date:

{ $dateFromParts: {

'year' : "$year",

'month' : "$month",

'day': 1

}

}

}}, {$project: {

ordenes: 1,

month: 1,

year: 1,

new\_date: 1

}}, {$group: {

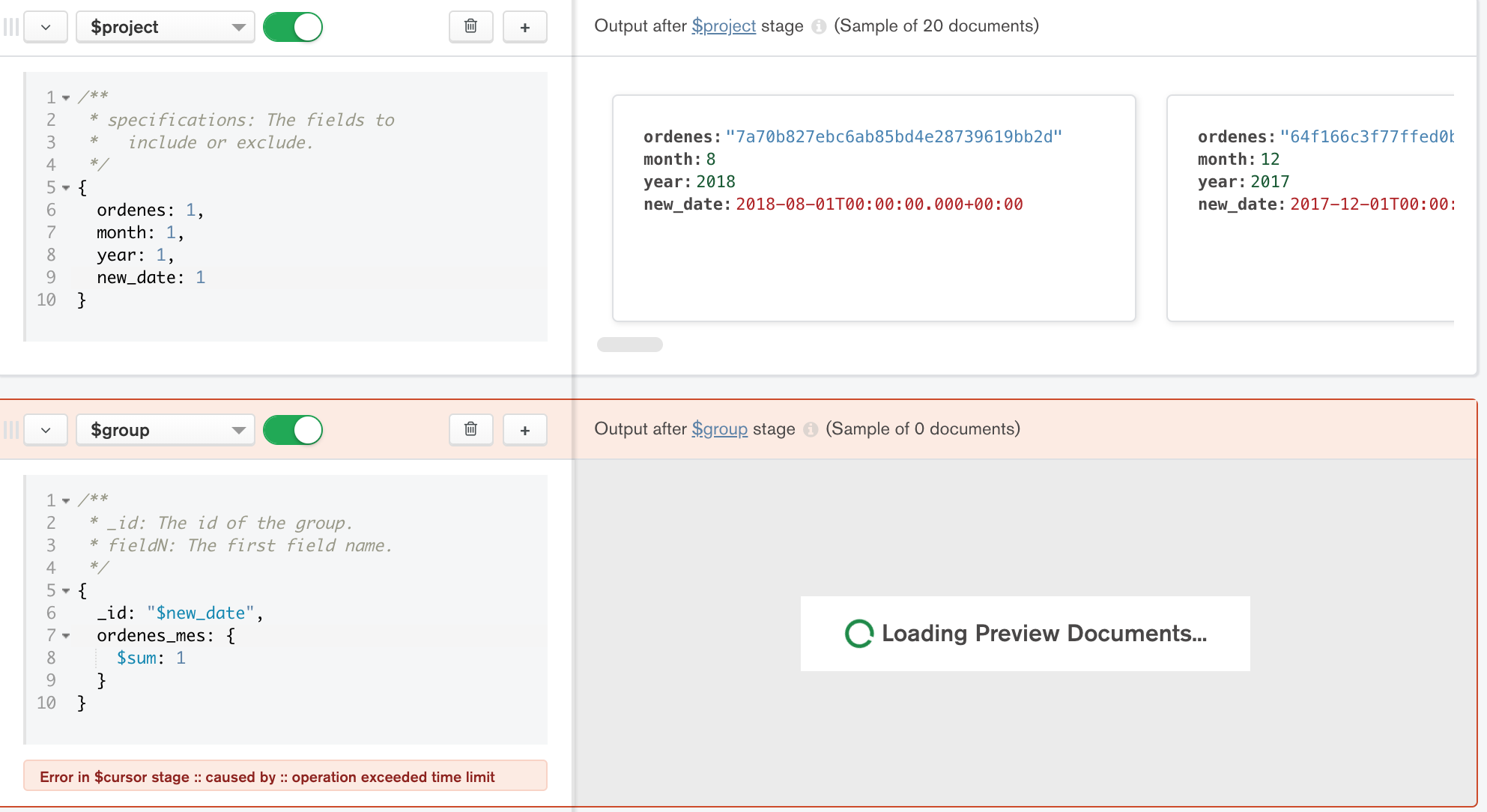
\_id: "$new\_date",

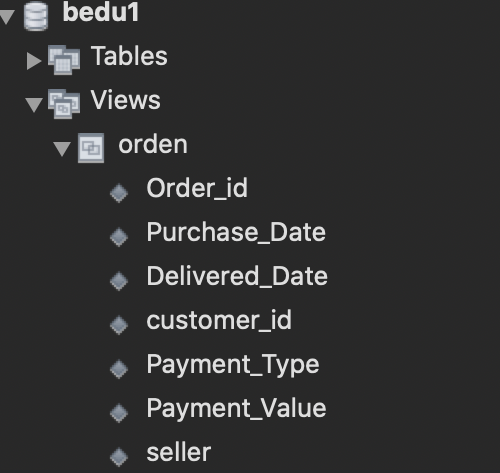
ordenes\_mes: {

$sum: 1

}

}}]



1. ¿Cuál es el tiempo promedio de espera en pedidos que sí fueron entregados?
   1. MySQL  
      Se creó una vista orden para hacer consultas más fáciles:  
        
      SELECT MONTH(Purchase\_Date) AS Purchase\_Month,   
       YEAR(Purchase\_Date) AS Purchase\_Year,

AVG(DATEDIFF(Delivered\_Date, Purchase\_Date)) AS  
AVG\_Delivery\_Days

FROM orden

WHERE Delivered\_Date IS NOT NULL

GROUP BY Purchase\_Month, Purchase\_Year

ORDER BY Purchase\_Month ASC;



* 1. MongoDB  
     [{$match: {

order\_status: "delivered"

}}, {$addFields: {

date\_diff\_milliseconds: {

$subtract: ["$order\_delivered\_customer\_date",

"$order\_purchase\_timestamp"]

}

}}, {$addFields: {

date\_diff\_days: {$divide: [

"$date\_diff\_milliseconds",

1000 \* 60 \* 60 \* 24

]},

group\_date: { $dateFromParts: {

'year' : {$year: "$order\_purchase\_timestamp"},

'month' : {$month: "$order\_purchase\_timestamp"},

'day': 1,} }

}}, {$group: {

\_id: "$group\_date",

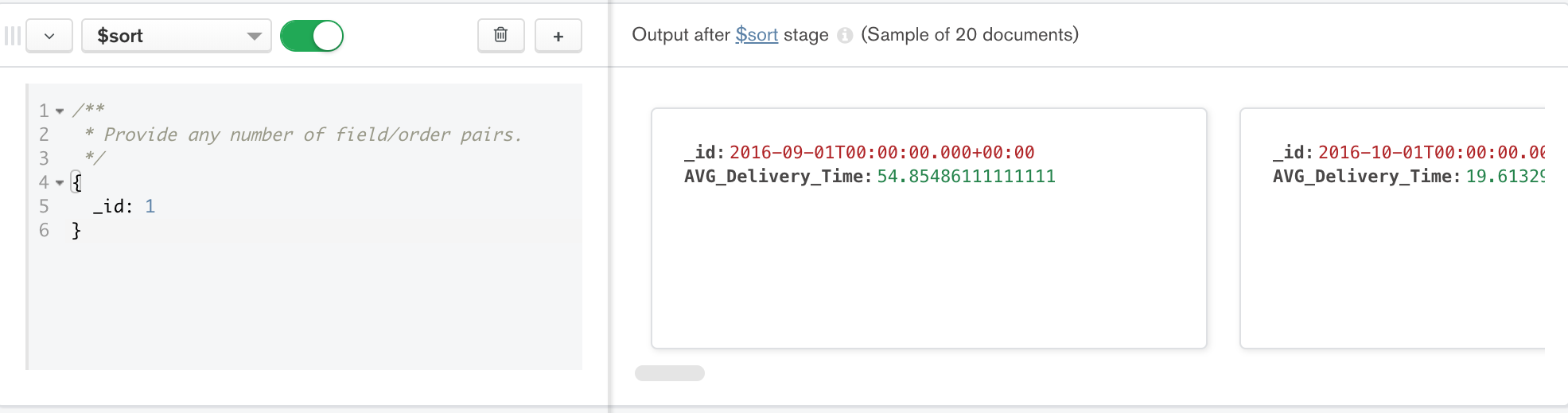
AVG\_Delivery\_Time: {

$avg: "$date\_diff\_days"

}

}}, {$sort: {

\_id: 1

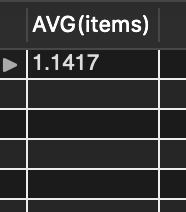
}}]  


(Existen pequeñas diferencias porque en MySQL se creó una vista con órdenes que tenían fecha de entrega y en MongoDB se filtraron las órdenes con order\_status de entregado).

1. ¿Cuántos ítems promedio tiene cada orden?
   1. MySQL  
      SELECT AVG(items) FROM

(SELECT order\_id, COUNT(\*) AS items

FROM order\_items

GROUP BY order\_id) AS sq;  


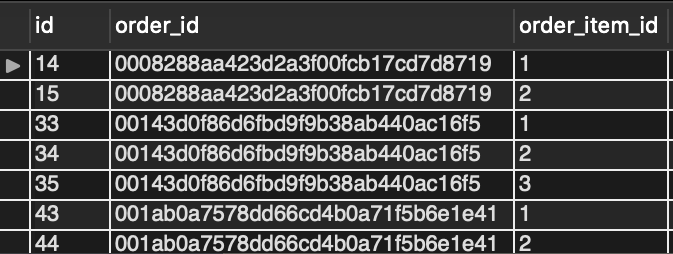
* 1. MongoDB  
     Añadiré pregunta extra 1 para reponer.

1. ¿Cuáles son las órdenes que tienen más de 1 item?
   1. MySQL  
      SELECT id, order\_id, order\_item\_id FROM order\_items

WHERE order\_id IN (SELECT DISTINCT(order\_id) FROM order\_items

WHERE order\_item\_id > 1

ORDER BY order\_id);



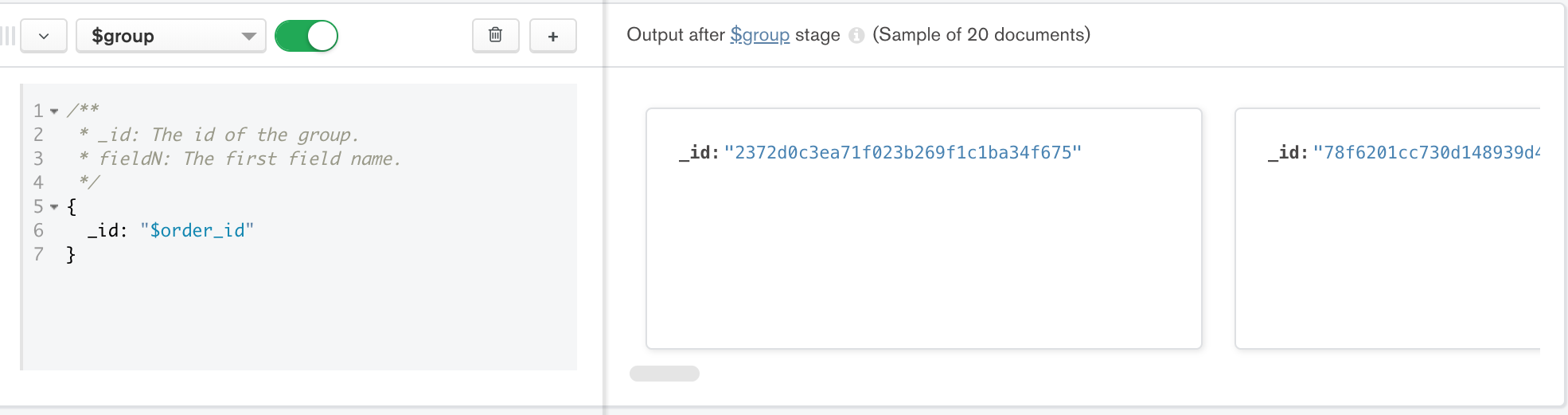
* 1. MongoDB  
     [{$match: {

order\_item\_id: {$gt: 1}

}}, {$group: {

\_id: "$order\_id"

}}]



1. ¿Cuál es la orden con más items y cuántos tiene?
   1. MySQL  
      SELECT order\_id, items FROM

(SELECT order\_id, COUNT(\*) AS items

FROM order\_items

GROUP BY order\_id) AS sq

WHERE items =

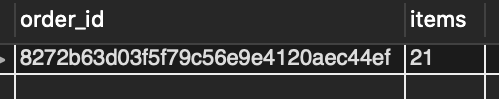
(SELECT MAX(items) FROM

(SELECT order\_id, COUNT(\*) AS items

FROM order\_items

GROUP BY order\_id) AS sq2

);



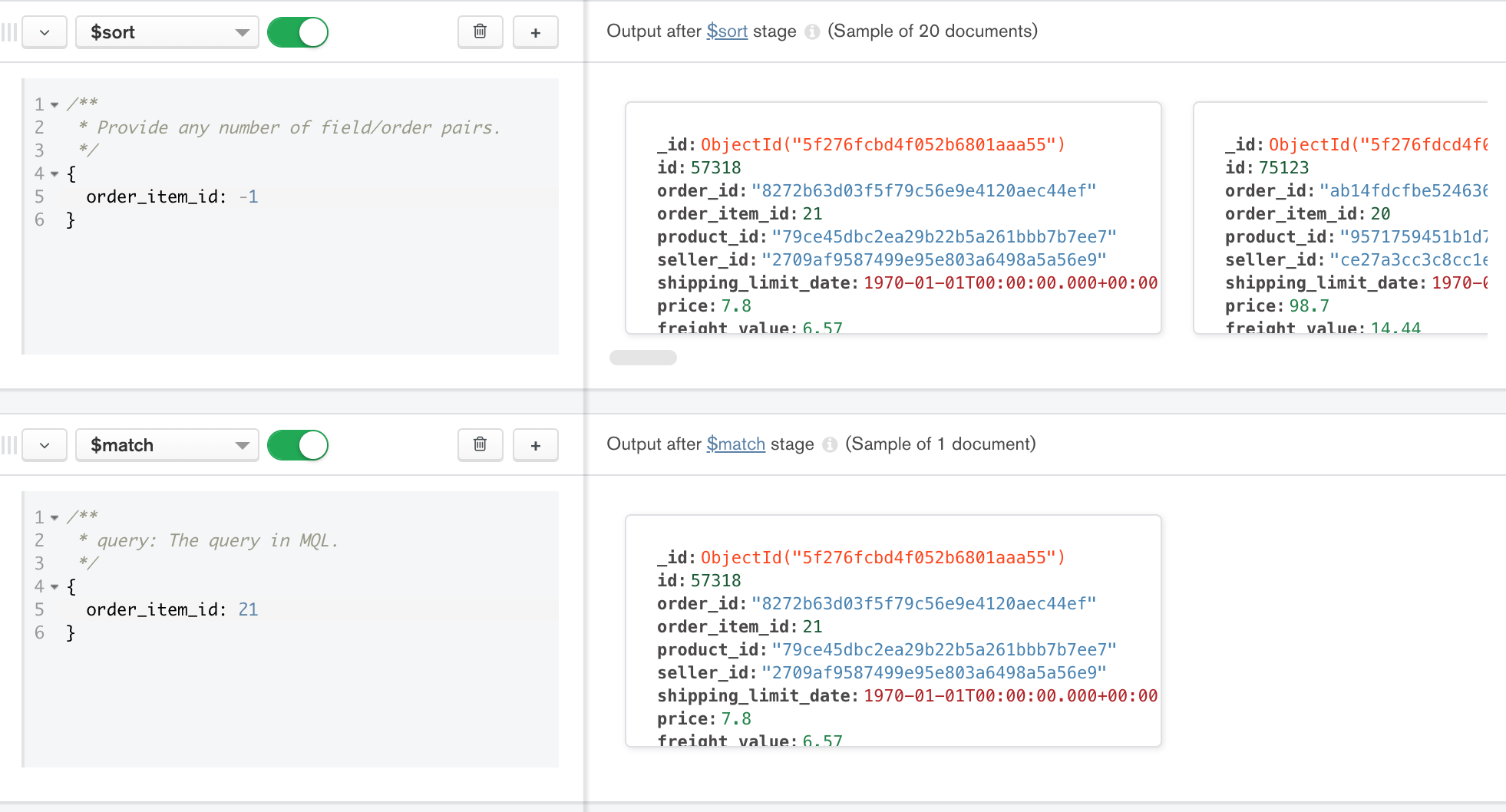
* 1. MongoDB  
     Primero ordene los registros por order\_item\_id de mayor a menor para ver cuál era el índice más alto, y después sólo hice un match para que me trajera sólo la(s) órden(es) con 21 items.  
     [{$sort: {

order\_item\_id: -1

}}, {$match: {

order\_item\_id: 21

}}]

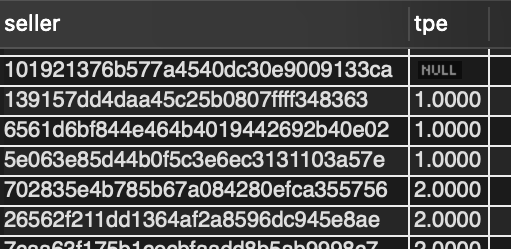


1. ¿Cuál es el tiempo promedio de entrega por Seller?
   1. MySQL  
      Usando la vista orden que se creó anteriormente, queda de la siguiente manera:  
        
      SELECT seller, AVG(DATEDIFF(Delivered\_Date, Purchase\_Date)) AS tpe

FROM orden

GROUP BY seller

ORDER BY tpe ASC;



(Trae registros en NULL porque hay sellers que no han registrado entregas)

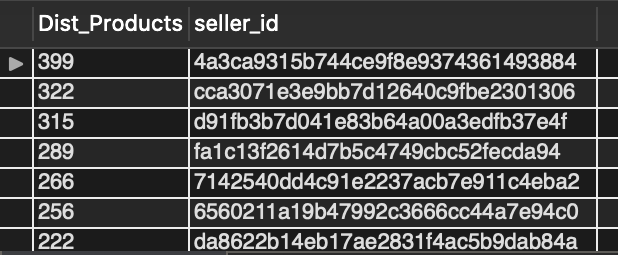
* 1. MongoDB  
     Añadiré pregunta extra 2 para reponer.

1. ¿Cuántos productos tiene cada seller?
   1. MySQL  
      SELECT COUNT(DISTINCT(product\_id)) AS Dist\_Products, seller\_id

FROM order\_items

GROUP BY seller\_id

ORDER BY 1 DESC;



* 1. MongoDB  
     Añadiré pregunta extra 3 para reponer.

### **Preguntas Extra MongoDB**

1. ¿Cuántas reviews tiene la orden con más reviews?  
   [{$group: {

\_id: "$order\_id",

reviews: {

$sum: 1

}

}}, {$sort: {

reviews: -1

}}]  


1. ¿Cuántas reviews hay con cada calificación? Crear vista.  
   [{$group: {

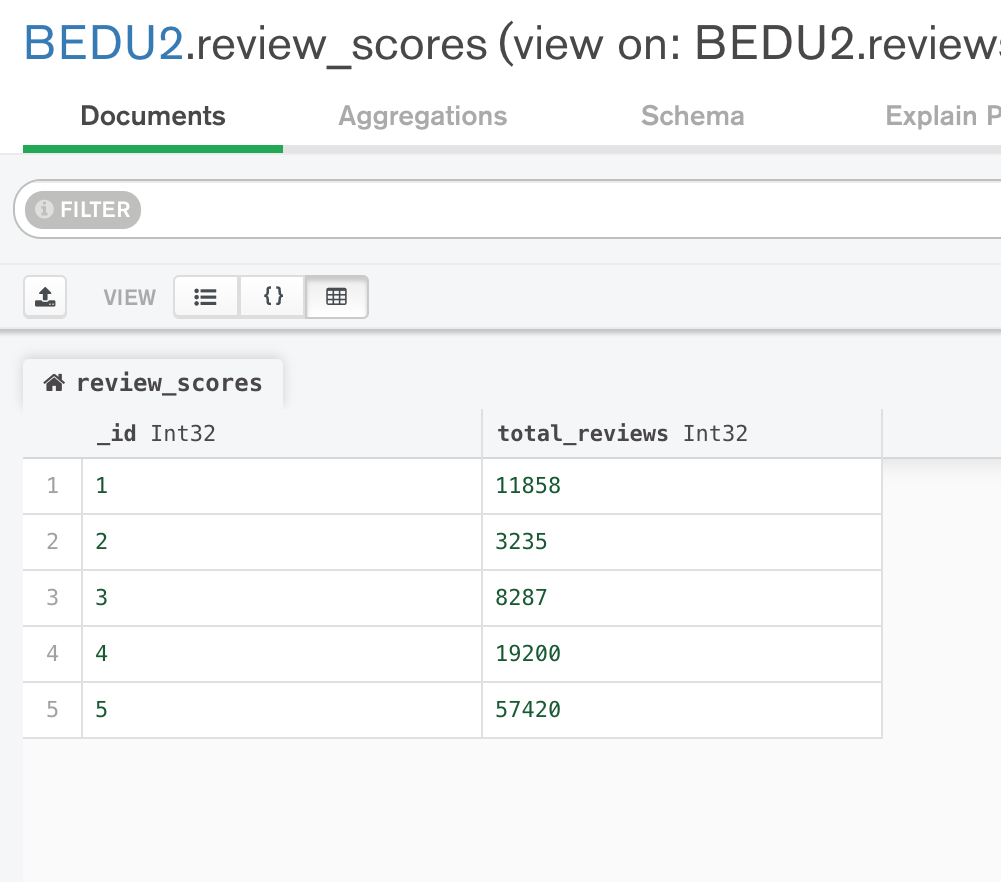
\_id: "$review\_score",

total\_reviews: {

$sum: 1

}

}}]



1. ¿Cuántas reviews hay con un mensaje escrito?

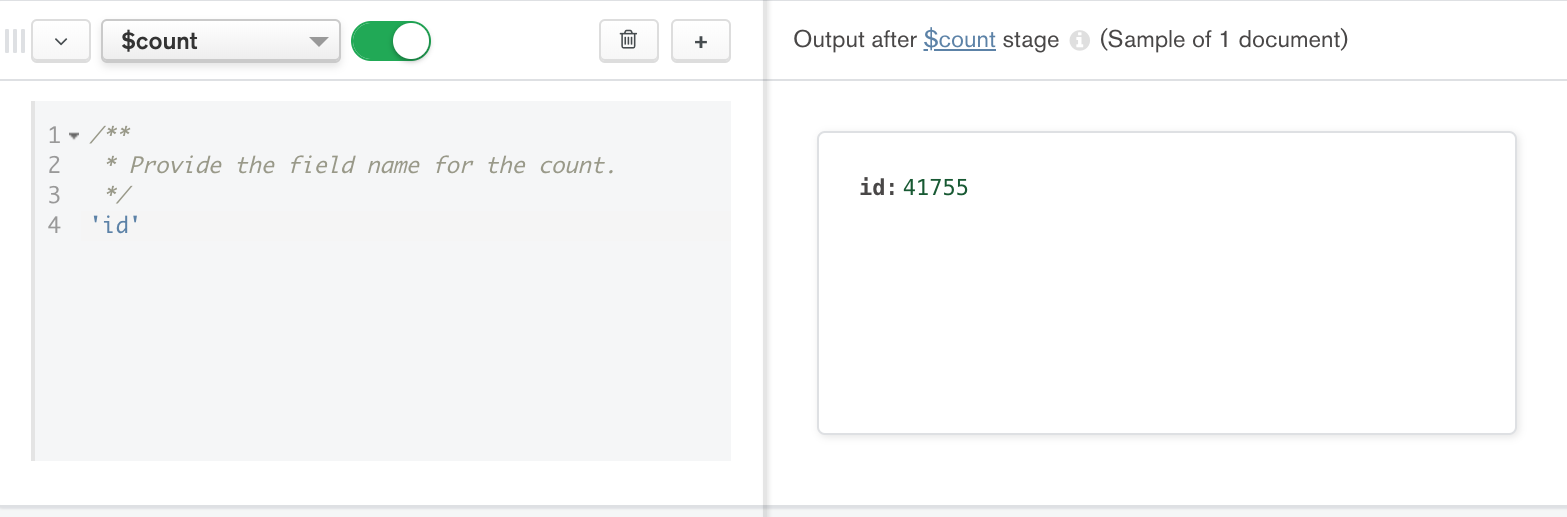
[{$match: {

review\_comment\_message: {

$ne: ""

}

}}, {$count: 'id'}]



### **Vistas**

1. Creando una vista "orden", ¿cuántos usuarios distintos realizan una compra cada mes?
   1. MySQL  
      CREATE VIEW orden AS

(SELECT o.order\_id AS Order\_id,

o.order\_purchase\_timestamp AS Purchase\_Date,

o.order\_delivered\_customer\_date AS Delivered\_Date,

o.customer\_id AS customer\_id,

op.payment\_type AS Payment\_Type,

op.payment\_value AS Payment\_Value,

s.seller\_id AS seller

FROM orders o

LEFT JOIN order\_payment op

ON o.order\_id = op.order\_id

RIGHT JOIN order\_items s

ON o.order\_id = s.order\_id

ORDER BY s.order\_id ASC);  
  
SELECT DATE\_FORMAT(Purchase\_Date, '%Y %m') AS date, COUNT(DISTINCT(customer\_id)) AS dist\_customers

FROM orden

GROUP BY DATE\_FORMAT(Purchase\_Date, '%Y %m')

ORDER BY 1;



1. Crear vista para agregar traducción a la tabla de products
   1. MySQL  
      CREATE VIEW productos AS

(SELECT p.product\_id, p.product\_category\_name,

t.product\_category\_name\_english, p.product\_photos\_qty,

p.product\_weight\_g, p.product\_length\_cm,  
 p.product\_height\_cm, p.product\_width\_cm

FROM products p

LEFT JOIN product\_category\_name\_translation t

ON p.product\_category\_name = t.product\_category\_name

WHERE p.product\_category\_name IS NOT null

ORDER BY product\_category\_name ASC);



* 1. MongoDB  
     [{$lookup: {

from: 'products\_translation',

localField: 'product\_category\_name',

foreignField: 'product\_category\_name',

as: 'translation'

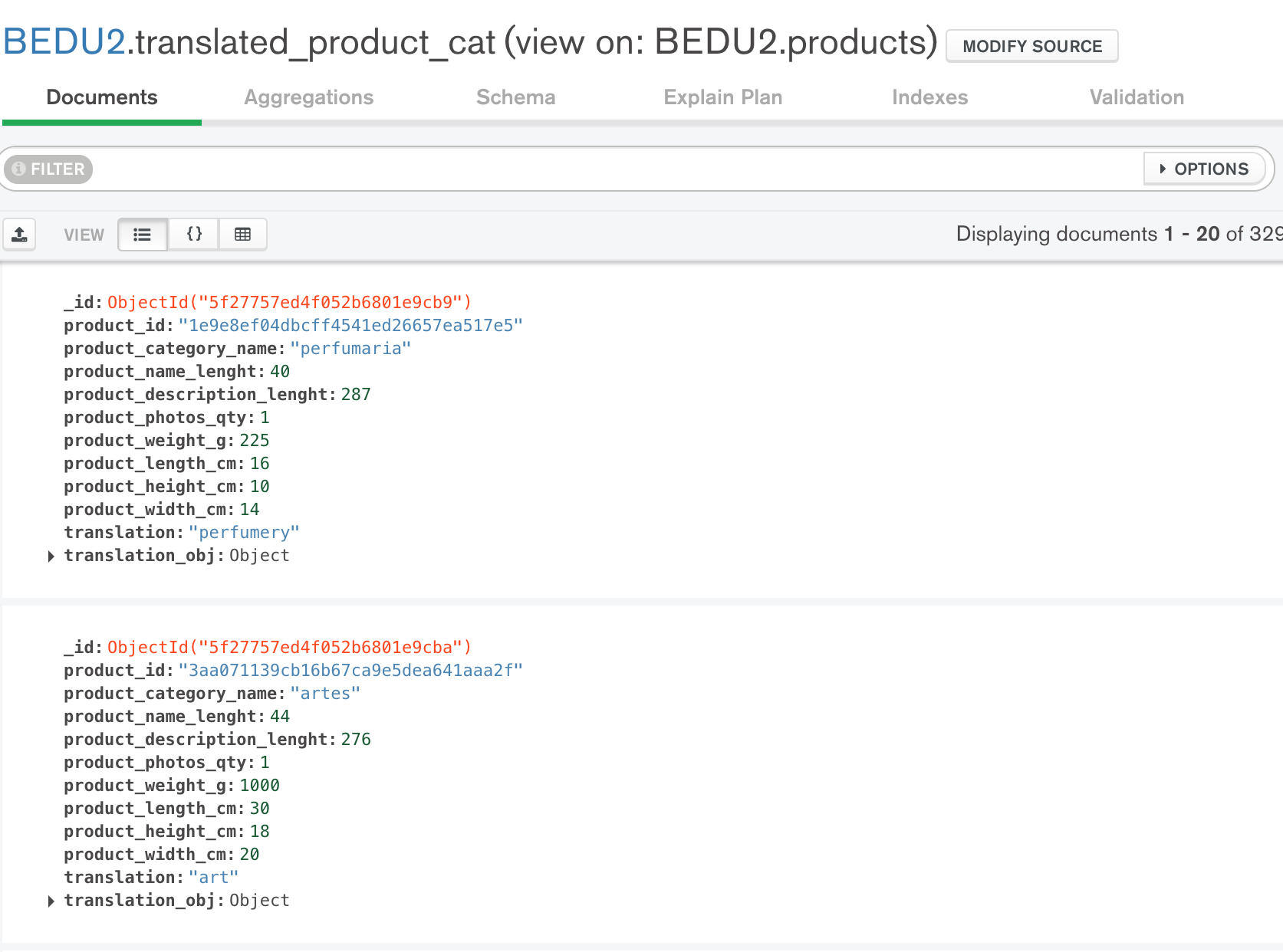
}}, {$addFields: {

translation\_obj: {$arrayElemAt: ["$translation", 0]}

}}, {$addFields: {

translation: "$translation\_obj.product\_category\_name\_english"

}}]



### 

### **Conclusiones**

Las conclusiones las dividiré en 2 grupos, las relativas al *Tipos de Bases de Datos* y las relativas al dataset de *Olist-Ecommerce Brazil*.

**Tipos de Bases de datos:**

1. Dentro de los tipos de bases de datos que cubrimos en el curso, tenemos las relacionales y no relacionales. Las relacionales son aquellas que por diseño buscan referenciar las tablas que las componen a través de la relación de campos en común. Las no relacionales, no necesitan tener una estructura predefinida, por lo que da un poco más de flexibilidad a la hora de llenarlas con información.
2. Para escoger entre un tipo o el otro, hay que fijarse entonces en la estructura de los datos. Si buscamos tener flexibilidad y no casarnos con una estructura de base de datos, es más recomendable MongoDB (o NOSQL). Pero si se tienen tablas que se relacionan entre sí, es mucho más sencillo gestionarlo con SQL.

**Olist-Ecommerce Brazil:**

1. La estructura de la base de datos está hecha en base a tablas y la relación entre ellas, por lo que debería de gestionarse en SQL.
2. Las localidades con más customers son aquellas que albergan grandes ciudades como:
   1. Sao Paulo
   2. Rio de Janeiro
   3. Minas Gerais.
3. El método de pago favorito es por medio de tarjeta de crédito (contando con un 77% de las órdenes).
4. Cuándo las personas realizan compras en Olist, lo más común es que compren únicamente un producto por transacción. El promedio de ítems por transacción es de 1.4 ítems.
5. Menos de la mitad de las reseñas tienen escritas un comentario, por lo que existe la posibilidad de separar asignar una calificación de escribir un comentario y así podríamos tener mayor número de calificaciones (al no ser tan invasivos).
6. Alrededor del 11% de las reseñas califican con la peor calificación su pedido (1), por lo que existe una gran ventana de oportunidad para conseguir que esas personas estén más conformes con sus pedidos.
7. El tiempo promedio de entrega de los sellers que sí han registrado entregas es de 8.35 días. Existen clientes que no han registrado entregas pero también existen entregas atípicas de más de un año, por lo que si eliminamos la única entrega que existe de 1,900 días, baja a 6.35. Habría que investigar bien por qué hay tiempos de entrega tan grandes.