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:

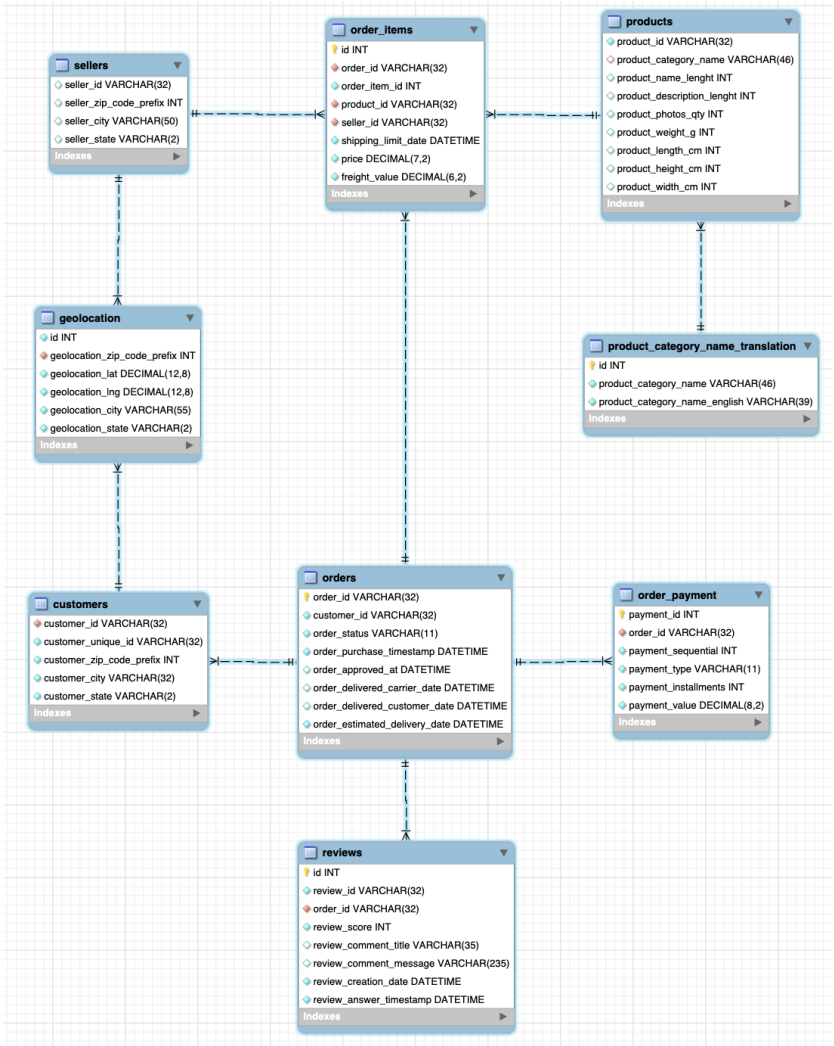
1. orders - contiene la siguiente información de las ordenes:
    a. order_id - identificador de la orden hecho de caracteres y números
    b. customer_id - identificador de cliente hecho de caracteres y números
    c. order_status - estado de la orden (entregado, en tránsito, etc.)
    d. order_purchase_timestamp - fecha de compra
    e. order_approved_at - fecha en que se aprobó la orden
    f. order_delivered_carrier_date - fecha en que se surtió orden a courier
    g. order_delivered_customer_date - fecha de entrega a cliente
    h. order_estimated_delivery_date - fecha estimada de entrega
2. order_items
    a. id - llave primaria autoincrementable
    b. order_id - identificador de la orden hecho de caracteres y números
    c. order_item_id - número de ítem dentro de la misma orden
    d. product_id - identificador de producto hecho de caracteres y números
    e. seller_id - identificador de vendedor hecho de caracteres y números

    f. shipping_limit_date - fecha límite de envío para cumplir con fecha de entrega estimada
    g. price - precio del ítem
    h. freight_value - valor del envío
3. customers
    a. customer_id - identificador de cliente hecho de caracteres y números
    b. customer_unique_id - identificador único de cliente hecho de caracteres y números (customer_id también es único)
    c. customer_zip_code_prefix - prefijo de código postal
    d. customer_city - ciudad en donde se registró el usuario
    e. customer_state - estado en donde se registró el usuario
4. order_payment
    a. payment_id - identificador del pago hecho de caracteres y números
    b. order_id - identificador de la orden hecho de caracteres y números
    c. payment_sequential - NA
    d. payment_type - tipo de pago (tarjeta de crédito, débito, etc.)
    e. payment_installments - número de pagos
    f. payment_value - valor de cada pago
5. reviews
    a. id - llave primaria autoincrementable
    b. review_id - identificador de la reseña hecho de caracteres y números
    c. order_id - identificador de la orden hecho de caracteres y números
    d. review_score - calificación asignada por usuario (de 1 a 5)
    e. review_comment_title - título de review (puede quedar vacío)
    f. review_comment_message - mensaje de reseña (puede quedar vacío)
    g. review_creation_date - fecha de creación de reseña
    h. review_answer_timestamp - fecha de respuesta de reseña
6. sellers
    a. seller_id - identificador del vendedor hecho de caracteres y números
    b. seller_zip_code_prefix - prefijo de código postal
    c. seller_city - ciudad en donde se registró el vendedor
    d. seller_state - estado en donde se registró el vendedor
7. geolocation
    a. id - llave primario autoincrementable
    b. geolocation_zip_code_prefix - prefijo de código postal
    c. geolocation_lat - coordenadas (longitud)
    d. geolocation_lon - coordenadas (latitud)
    e. geolocation_city - ciudad a la que pertenecen coordenadas
    f. geolocation_state - estado al que pertenecen coordenadas
8. products
    a. product_id - identificador de producto hecho de caracteres y números
    b. product_category_name - nombre de categoría de producto (en portugués) product_name_length - longitud de nombre de producto
    c. product_description_length - longitud de descripción de producto
    d. product_photos_qty - cantidad de fotos asignadas al producto
    e. product_weight_g - peso del producto en gramos
    f. product_length_cm - largo del producto en cm
    g. product_height_cm - alto del producto en cm

h. product_width_cm - ancho del producto en cm

9. product_category_name_translation
    a. id - llave primario autoincrementable
    b. product_category_name - nombre de categoría de producto (en portugués)
    c. product_category_name_translation - traducción de nombre de categoría a inglés.

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

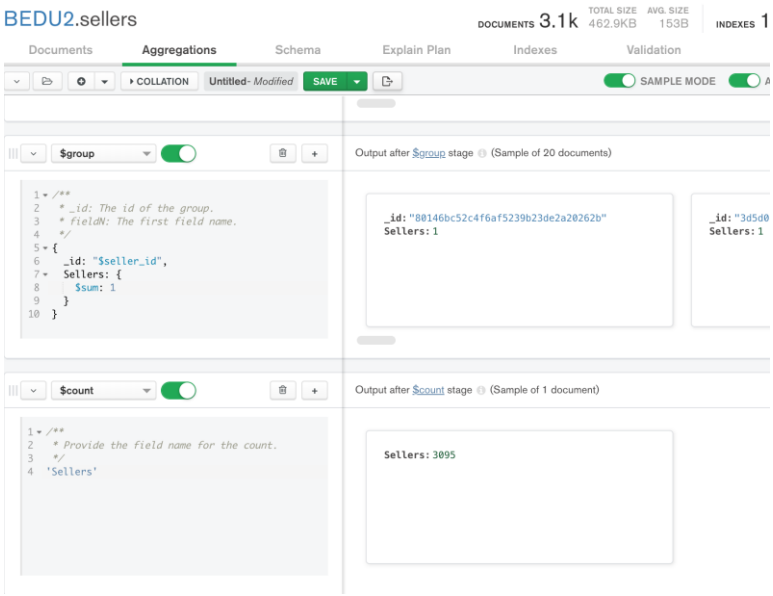


Preguntas

1. ¿Cuántos distintos sellers hay registrados?
    a. MYSQL
    SELECT COUNT(DISTINCT(seller_id)) FROM sellers;

Sellers
3095

    b. MongoDB
    [{$group: {
    _id: "$seller_id",
    Sellers: {
    $sum: 1
    }
    }},
    {$count: 'Sellers'
    }]



2. ¿Cuáles son las distintas locaciones (estados) de los Sellers?
    a. MySQL
    SELECT DISTINCT(seller_city), seller_state FROM sellers

ORDER BY seller_city ASC;

seller_city	seller_state
04482255	RJ
abadia de goias	GO
afonso claudio	ES
aguas claras df	SP
alambari	SP
alfenas	MG
almirante tamandare	PR
alvares machado	SP
alvorada	RS
americana	SP
amparo	SP

b. 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"}
                }
        }]
```

3. ¿Cuántas órdenes se hicieron por medio del seller registrado en una ciudad invalida?
   a. MySQL

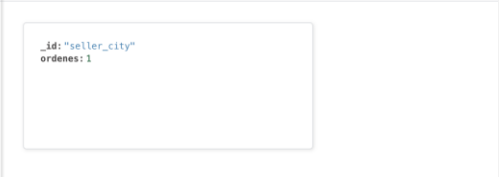
```
SELECT seller_id, COUNT(order_id) AS ordenes
FROM order_items
WHERE seller_id =
        (SELECT seller_id FROM sellers WHERE seller_city = '04482255')
```

GROUP BY seller_id;

seller_id	ordenes
ceb7b4fb9401cd378de7886317ad1b47	1

b. 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
  }
}}]
```



4. ¿Quiénes son los sellers de Sao Paulo (SP)?
   a. MySQL

```
SELECT * FROM sellers
WHERE seller_state like "SP";
```

seller_id	seller_zip_code_prefix	seller_city	seller_state
3442f8959a84dea7ee197c632cb2df15	13023	campinas	SP
d1b65fc7debc3361ea86b5f14c68d2e2	13844	mogi guacu	SP
c0f3eea2e14555b6faeea3dd58c1b1c3	4195	sao paulo	SP
51a04a8a6bdcb23deccc82b0b80742cf	12914	braganca paulista	SP
1b938a7ec6ac5061a66a3766e0e75f90	16304	penapolis	SP

b. MongoDB

```
db.sellers.find(
  {seller_state:'SP'},
  {seller_id:1,seller_state:1}
)
```

	_id ObjectId	seller_id String	seller_state String
1	5f2779aad4f052b68020a457	"3442f8959a84dea7ee197c632cb	"SP"
2	5f2779aad4f052b68020a458	"d1b65fc7debc3361ea86b5f14c6	"SP"
3	5f2779aad4f052b68020a45a	"c0f3eea2e14555b6faeea3dd58c	"SP"
4	5f2779aad4f052b68020a45b	"51a04a8a6bdcb23deccc82b0b80	"SP"
5	5f2779aad4f052b68020a45e	"1b938a7ec6ac5061a66a3766e0e	"SP"
6	5f2779aad4f052b68020a45f	"768a86e36ad6aae3d03ee3c6433	"SP"
7	5f2779aad4f052b68020a462	"a7a9b880c49781da66651ccf4ba	"SP"
8	5f2779aad4f052b68020a463	"8bd0f31cf0a614c658f6763bd02	"SP"
9	5f2779aad4f052b68020a464	"05a48cc8859962767935ab90874	"SP"
10	5f2779aad4f052b68020a46a	"f9ec7093df3a7b346b7bcf78640	"SP"
11	5f2779aad4f052b68020a46b	"4e6015589b781adaa5ce7f1892d	"SP"
12	5f2779aad4f052b68020a46c	"4cf490a58259286ada5ba8525ba	"SP"
13	5f2779aad4f052b68020a46d	"f7496d659ca9fdaf323c0aae841	"SP"
14	5f2779aad4f052b68020a46e	"2ff97219cb8622eaf3cd89b7d9c	"SP"
15	5f2779aad4f052b68020a472	"116ccb1a1604bc88e4d234a8c23	"SP"

5. ¿Cuantos customers distintos hay?
   a. MySQL

```
SELECT COUNT(DISTINCT(customer_unique_id)) AS Dist_Customers
FROM customers;
```

Dist_Customers
96096

   b. MongoDB

```
[{$group: {
  _id: "$customer_unique_id",
  Dist_Customers: {
    $sum: 1
  }
```

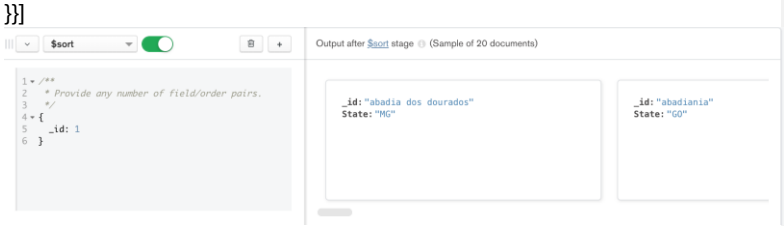
}}, {$group: {
  _id: null,
  Total_Customers: {
    $sum: 1
  }
}}]

6. ¿De dónde son los customers (estados)?
   a. MySQL
   SELECT DISTINCT(customer_city), customer_state
   FROM customers
   ORDER BY customer_state;

customer_city	customer_state
brasileia	AC
cruzeiro do sul	AC
epitaciolandia	AC
manoel urbano	AC
porto acre	AC
rio branco	AC
senador guiomard	AC
xapuri	AC
agua branca	AL
anadia	AL

   b. 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
}}]



7. ¿Cuántos usuarios hay de cada estado?
   a. MySQL

```
SELECT customer_state, COUNT(*) AS CustomersXState
FROM customers
GROUP BY customer_state
ORDER BY 2 DESC;
```

customer_state	CustomersXState
SP	41746
RJ	12852
MG	11635
RS	5466
PR	5045
SC	3637
BA	3380

   b. 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
}}]
```

8. ¿Cuáles son los distintos métodos de pago y cuál es su share general?
    a. 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;
```

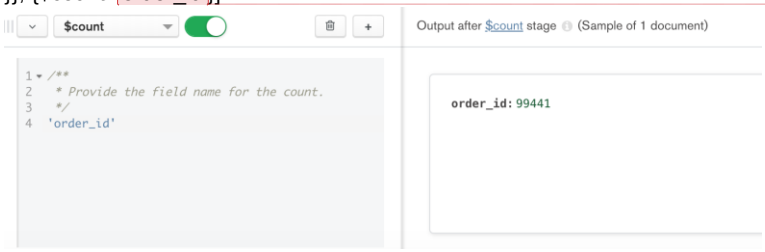
payment_type	payment_method_sha...
credit_card	0.7723
boleto	0.1990
voucher	0.0581
debit_card	0.0154
not_defined	0.0000

    b. 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:

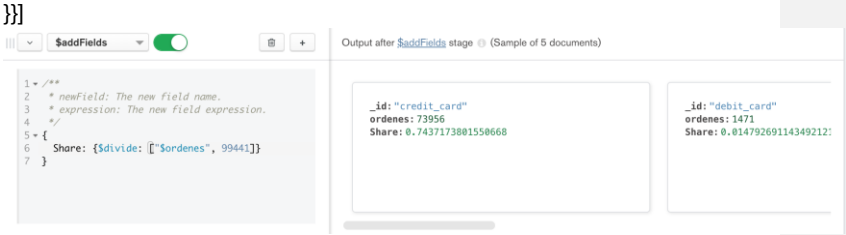
```
[{$group: {
  _id: "$order_id",
  ordenes: {
    $sum: 1
  }
}}, {$count: 'order_id'}]
```



Comentado [1]: Por alguna razón que no logro entender, tengo que son 99,441 órdenes distintas pero sé que son 99,440. Decido seguir adelante porque la diferencia es no significativa.

Cálculo de share:

```
[{$group: {
  _id: "$payment_type",
  ordenes: {
    $sum: 1
  }
}}, {$addFields: {
  Share: {$divide: ["$ordenes", 99441]}
}}]
```



9. ¿Cuántas órdenes hay por mes?
   a. 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;
```

Month	Year	Sales
12	2016	1
10	2016	324
9	2016	4
12	2017	5673
11	2017	7544
10	2017	4631
9	2017	4285

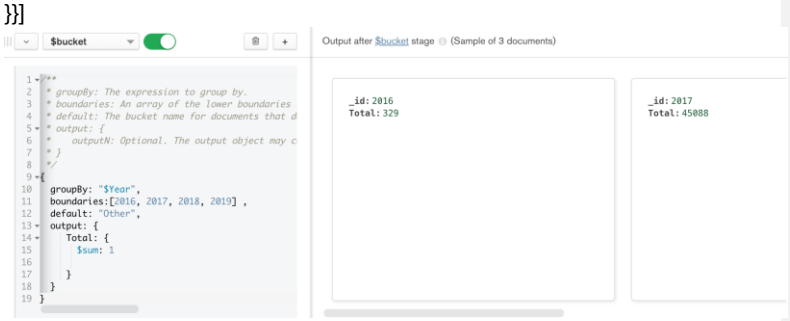
   b. MongoDB

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

          }
  }
}}]
```

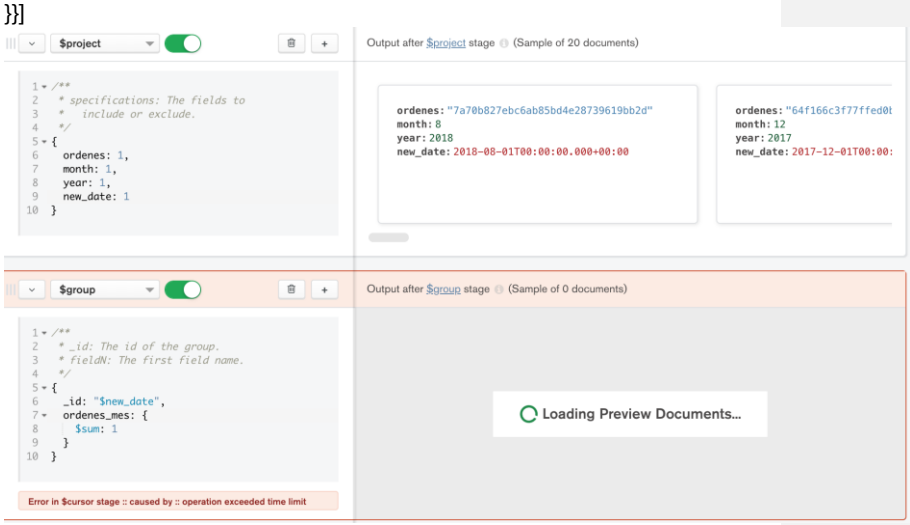


En teoría con esto ya debería de salir la opción de órdenes por mes pero MongoDB no logra sacar el output. Usé $addToSet en vez de $push para quitar duplicados:

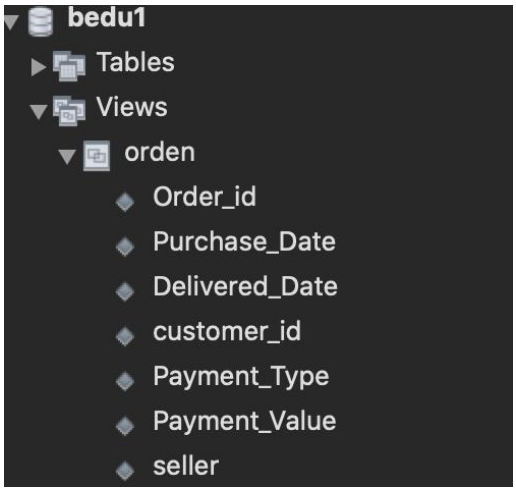
```
[{$group: {
  _id: "$order_id",
  ordenes: {
    $addToSet: "$order_id"
  }
}}, {$project: {
  ordenes: 1,
  _id: 0
}}, {$unwind: {
  path: "$ordenes"
}}, {$lookup: {
  from: 'orders',
  localField: 'ordenes',
  foreignField: 'order_id',
  as: 'order_array'
}}, {$addFields: {
  order_obj: {$arrayElemAt: ["$order_array",0]}
}}, {$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
  }
}}]
```



10. ¿Cuál es el tiempo promedio de espera en pedidos que sí fueron entregados?
    a. MySQL
    Se creó una vista orden para hacer consultas más fáciles:



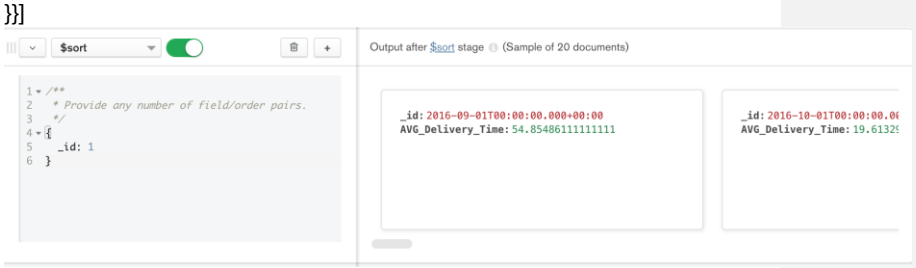

```
SELECT MONTH(Purchase_Date) AS Purchase_Month,
       YEAR(Purchase_Date) AS Purchase_Year,
       AVG(DATEDIFF(Delivered_Date, Purchase_Date)) AS
       AVG_Delivery_Days
FROM orden
WHERE Delivered_Date IS NOT NULL
GROUP BY Purchase_Month, Purchase_Year
ORDER BY Purchase_Month ASC;
```

Purchase_Month	Purchase_Year	AVG_Delivery_Days
1	2017	12.7487
1	2018	14.0120
2	2017	13.3675
2	2018	16.8007
3	2017	13.0997
3	2018	15.9345
4	2017	14.9047

b. 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).

11. ¿Cuántos ítems promedio tiene cada orden?
    a. MySQL

```
SELECT AVG(items) FROM
(SELECT order_id, COUNT(*) AS items
FROM order_items
GROUP BY order_id) AS sq;
```

AVG(items)
1.1417

    b. MongoDB

    Añadiré pregunta extra 1 para reponer.

12. ¿Cuáles son las órdenes que tienen más de 1 item?
    a. 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);
```

	id	order_id	order_item_id
▶	14	0008288aa423d2a3f00fcb17cd7d8719	1
	15	0008288aa423d2a3f00fcb17cd7d8719	2
	33	00143d0f86d6fbd9f9b38ab440ac16f5	1
	34	00143d0f86d6fbd9f9b38ab440ac16f5	2
	35	00143d0f86d6fbd9f9b38ab440ac16f5	3
	43	001ab0a7578dd66cd4b0a71f5b6e1e41	1
	44	001ab0a7578dd66cd4b0a71f5b6e1e41	2

b. MongoDB

```
[{$match: {
  order_item_id: {$gt: 1}
}}, {$group: {
  _id: "$order_id"
}}]
```

13. ¿Cuál es la orden con más items y cuántos tiene?
    a. MySQL

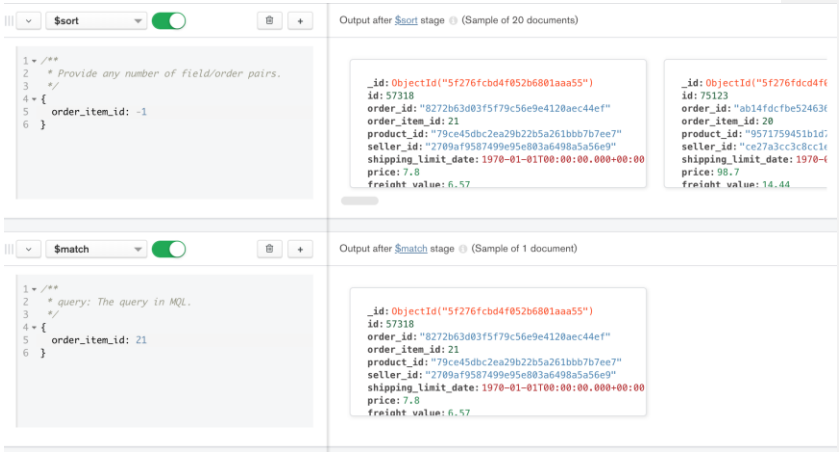
```
SELECT order_id, items FROM
        (SELECT order_id, COUNT(*) AS items
        FROM order_items
        GROUP BY order_id) AS sq
WHERE items =
        (SELECT MAX(items) FROM
                (SELECT order_id, COUNT(*) AS items
                FROM order_items
                GROUP BY order_id) AS sq2
);
```

	order_id	items
▶	8272b63d03f5f79c56e9e4120aec44ef	21

    b. 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
}}]
```



14. ¿Cuál es el tiempo promedio de entrega por Seller?
    a. 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;

seller	tpe
101921376b577a4540dc30e9009133ca	NULL
139157dd4daa45c25b0807ffff348363	1.0000
6561d6bf844e464b4019442692b40e02	1.0000
5e063e85d44b0f5c3e6ec3131103a57e	1.0000
702835e4b785b67a084280efca355756	2.0000
26562f211dd1364af2a8596dc945e8ae	2.0000

    (Trae registros en NULL porque hay sellers que no han registrado entregas)
    b. MongoDB
    Añadiré pregunta extra 2 para reponer.
15. ¿Cuántos productos tiene cada seller?
    a. MySQL
    SELECT COUNT(DISTINCT(product_id)) AS Dist_Products, seller_id
    FROM order_items
    GROUP BY seller_id
    ORDER BY 1 DESC;

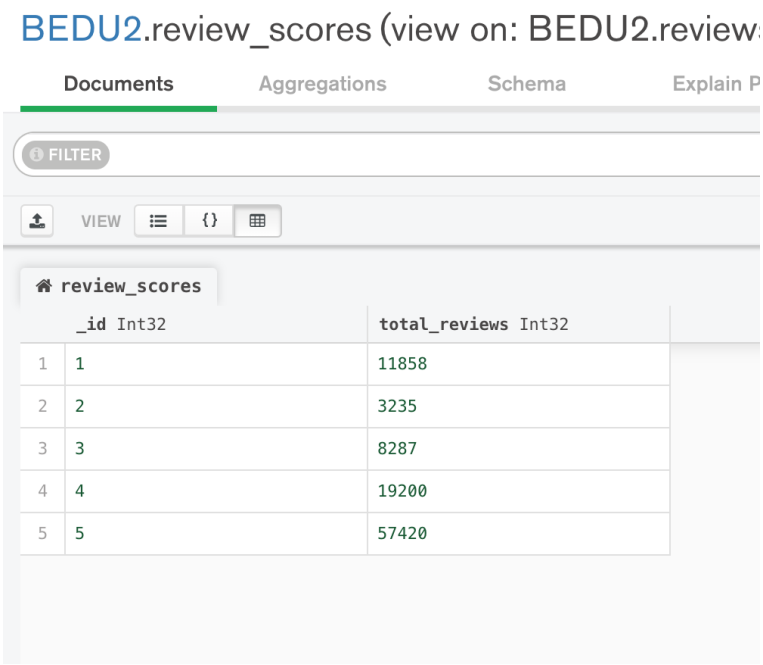
	Dist_Products	seller_id
▶	399	4a3ca9315b744ce9f8e9374361493884
	322	cca3071e3e9bb7d12640c9fbe2301306
	315	d91fb3b7d041e83b64a00a3edfb37e4f
	289	fa1c13f2614d7b5c4749cbc52fecda94
	266	7142540dd4c91e2237acb7e911c4eba2
	256	6560211a19b47992c3666cc44a7e94c0
	222	da8622b14eb17ae2831f4ac5b9dab84a

b. MongoDB
Añadiré pregunta extra 3 para reponer.

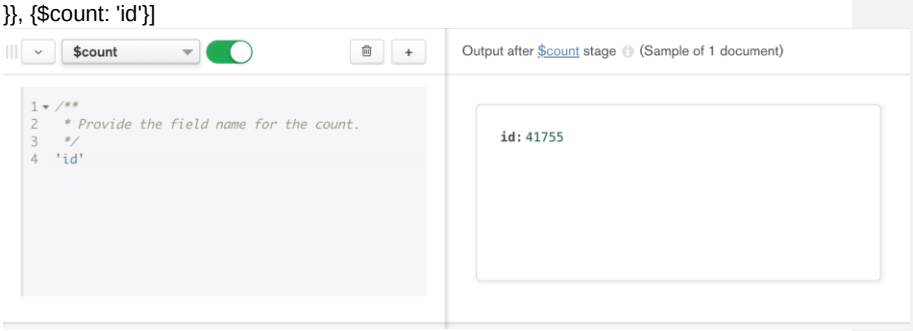
Preguntas Extra MongoDB

1. ¿Cuántas reviews tiene la orden con más reviews?
```
[{$group: {
  _id: "$order_id",
  reviews: {
    $sum: 1
  }
}}, {$sort: {
  reviews: -1
}}]
```


2. ¿Cuántas reviews hay con cada calificación? Crear vista.
```
[{$group: {
  _id: "$review_score",
  total_reviews: {
    $sum: 1
  }
}}]
```



3. ¿Cuántas reviews hay con un mensaje escrito?
[{$match: {
review_comment_message: {
$ne: ""
}
}}, {$count: 'id'}]

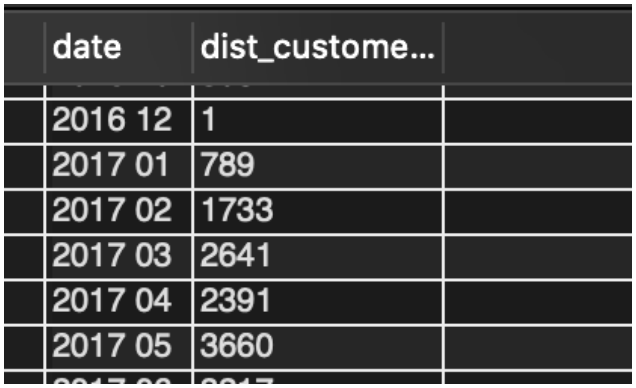


Vistas

1. Creando una vista "orden", ¿cuántos usuarios distintos realizan una compra cada mes?
    a. 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;
```



date	dist_custome...
2016 12	1
2017 01	789
2017 02	1733
2017 03	2641
2017 04	2391
2017 05	3660

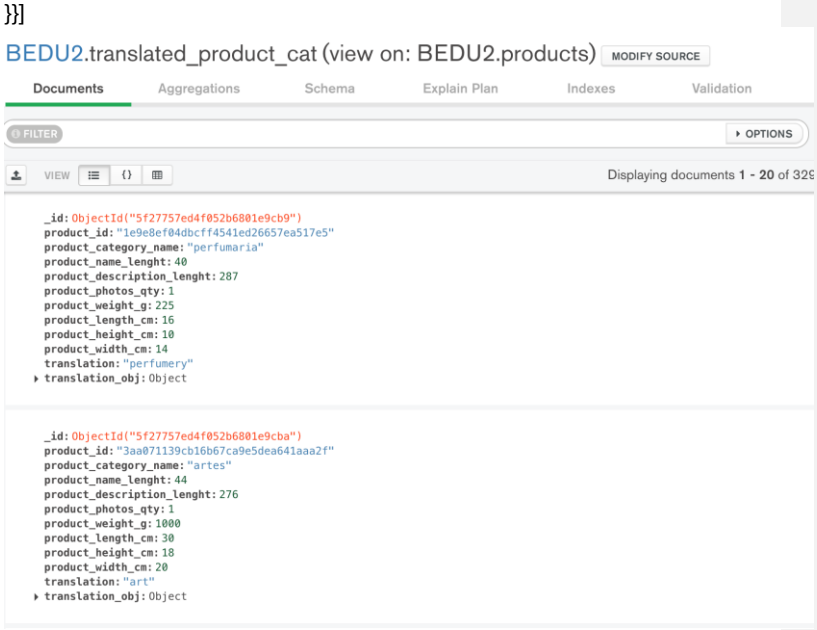
2. Crear vista para agregar traducción a la tabla de products
    a. MySQL

```
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);
```

product_id	product_category_name	product_category_name_engli...	product_photos_...	p
0b2a1288e8ba64c797e7586c8df75602	agro_industria_e_comercio	agro_industry_and_commerce	1	3
8a67e4e6cc6f2abf65fe2164ea51b219	agro_industria_e_comercio	agro_industry_and_commerce	1	2
2b69866f22de8dad69c976771daba91c	agro_industria_e_comercio	agro_industry_and_commerce	1	3
a0fe1efb855f3e786f0650268cd77f44	agro_industria_e_comercio	agro_industry_and_commerce	1	1
674005996f7500049a539559eeec5b76	agro_industria_e_comercio	agro_industry_and_commerce	4	6
e1834e5158e392bab21d691ac7b4eccd	agro_industria_e_comercio	agro_industry_and_commerce	1	1
018ca97302e4293050cc41413194bb19	agro_industria_e_comercio	agro_industry_and_commerce	1	3

b. 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:
    a. Sao Paulo
    b. Rio de Janeiro
    c. 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.