MongoDB-Calculate Total (Using Referencing)

Task goal:

To analyze orders regardless of the customer using the referencing design technique. With this technique, we'll create one document for each customer. In the customer document, we need to have all the sales order ids of the customer but not details about an order, we can use CustomerID as the PK of customer document by putting it in the _id field. We'll also create a document for each order. In the order document, there will be details about the order. We can use SalesOrderID as the PK of the order document by putting it in the _id field.

Example data

CustomerID	LastName	FirstName	EmailAddress	SalesOrder	TotalOrder	OrderValue
30010	Mew	Stephen	stephen4@adventure-works.com	44529	1	472
30010	Mew	Stephen	stephen4@adventure-works.com	46063	1	985
30020	Miller	Emilo	emilo0@adventure-works.com	53525	80	592223
30020	Miller	Emilo	emilo0@adventure-works.com	58968	16	141240
30020	Miller	Emilo	emilo0@adventure-works.com	65286	21	134777
30020	Miller	Emilo	emilo0@adventure-works.com	71838	17	103238
30083	Track	Glenn	glenn0@adventure-works.com	47398	152	3597824
30083	Track	Glenn	glenn0@adventure-works.com	48339	124	2574849
30083	Track	Glenn	glenn0@adventure-works.com	49496	72	1428758
30083	Track	Glenn	glenn0@adventure-works.com	50689	120	2343950

1. MongoDB database design and create documents.

Approach1:

Manually design

- **Understand the requirements, design the schema**
- **Process your raw data, clean it, and transform it into the format that matches your MongoDB schema. For example, using Excel or Google Spreadsheets**
- **Export the processed data to a JSON or CSV file.**
- **Import your JSON or CSV file into MongoDB compass.**
- **After designing manually, go directly to Step 4, skipping Steps 2-6 which are for Approach2 only**

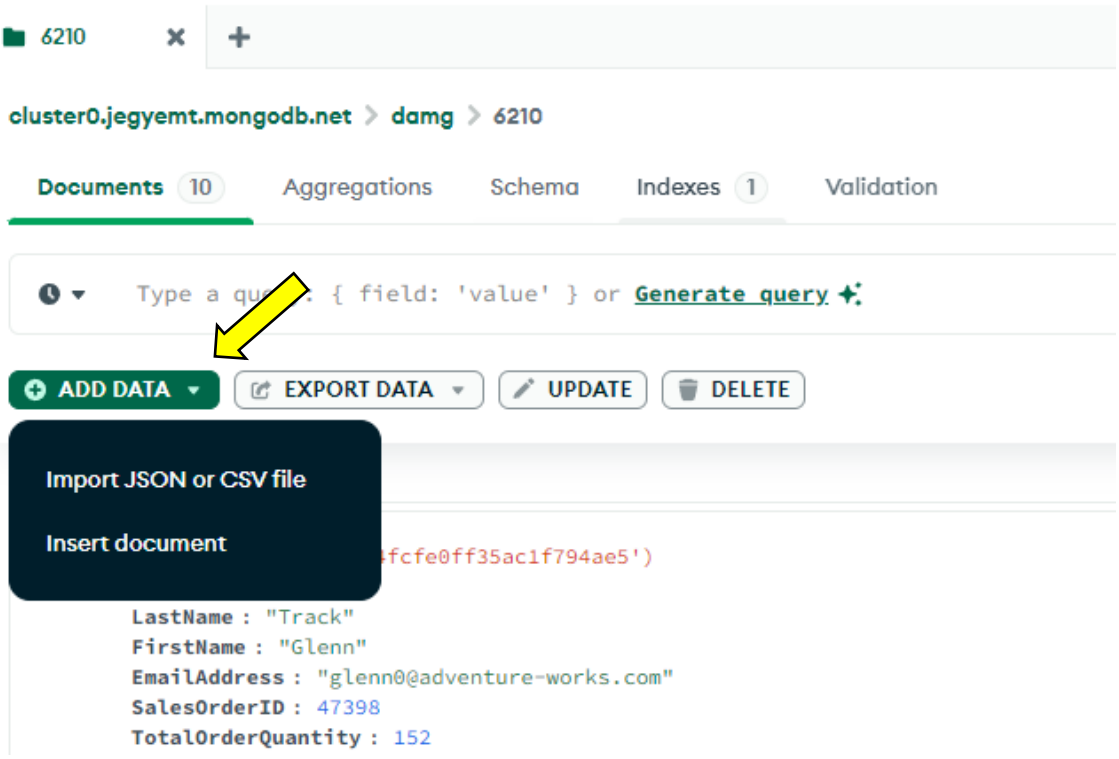
Approach2:

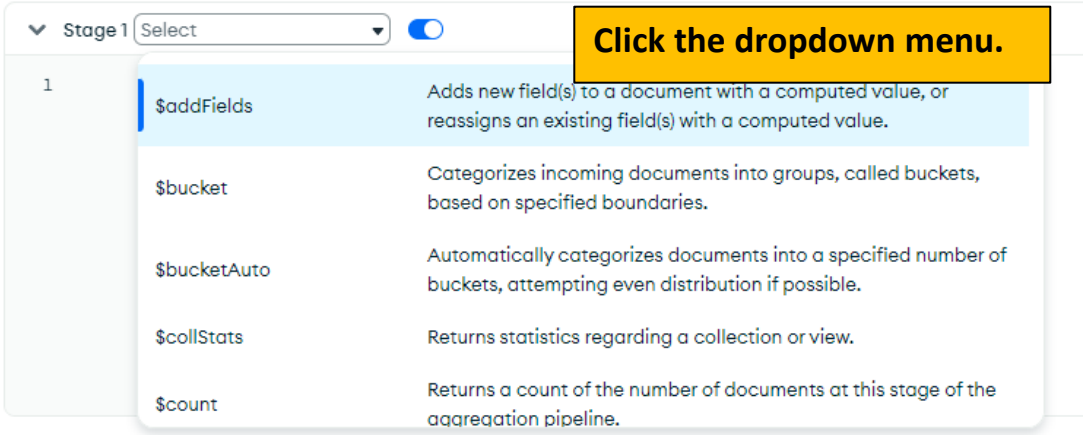
Using Aggregation Pipeline to process your data

- **See Step2-6**

2. Importing Data

2.1 Open MongoDB Compass and connect to the database and import the raw data (JSON or CSV file), my example database is named "damg" and the collection I inserted the data into is "6210".





(Navigate to the "Aggregations" tab within your selected collection, this will allow you to start creating an aggregation pipeline. Click on the "Create New" button to start building a new aggregation pipeline. Click on the dropdown menu where you can find various options.)

3. Using Aggregation Pipeline to build "Customer "collection because the original collection doesn't currently have an array of SalesOrderIDs but rather individual documents for each order.

3.1 Define $group stage

_id: Set this to "$CustomerID" to group by CustomerID.

Aggregate Fields:

LastName: Use $first to take the first occurrence of LastName.

FirstName: Use $first to take the first occurrence of FirstName.

EmailAddress: Use $first to take the first occurrence of EmailAddress.

SalesOrderIDs: Use $push to aggregate all SalesOrderID values into an array.

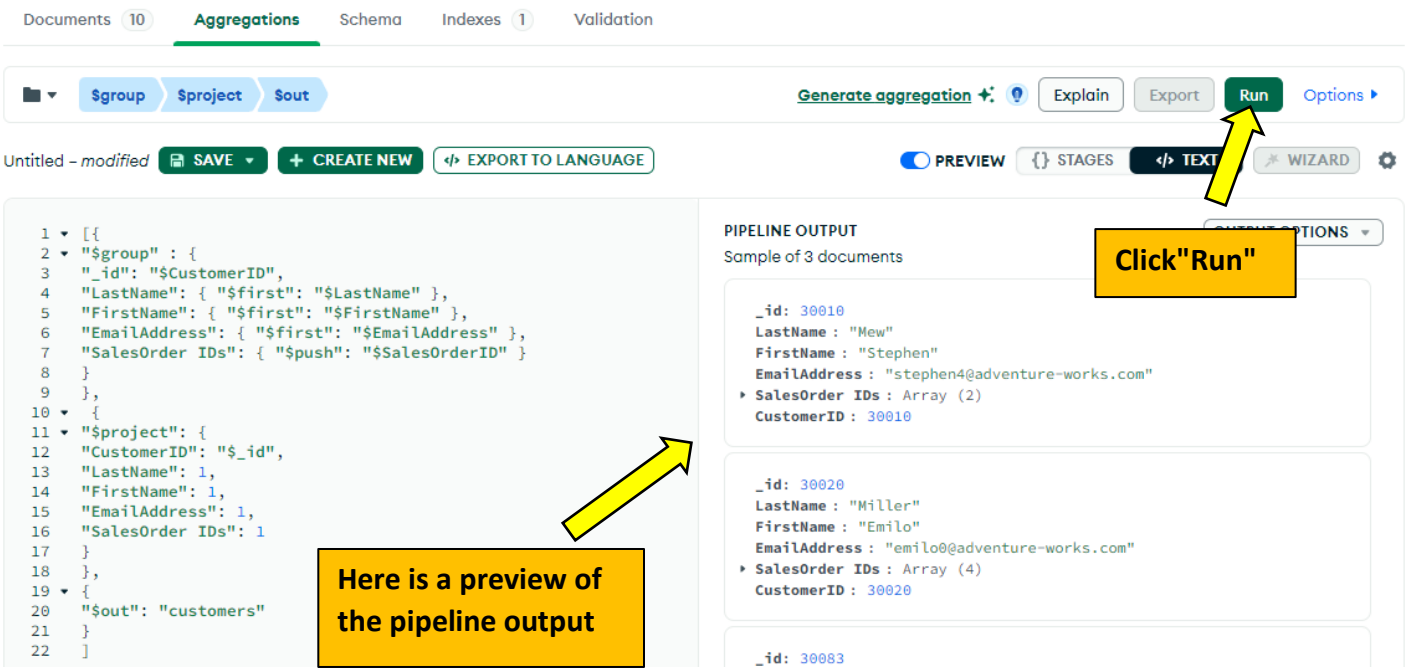
3.2 Define $project stage

Project the Fields:

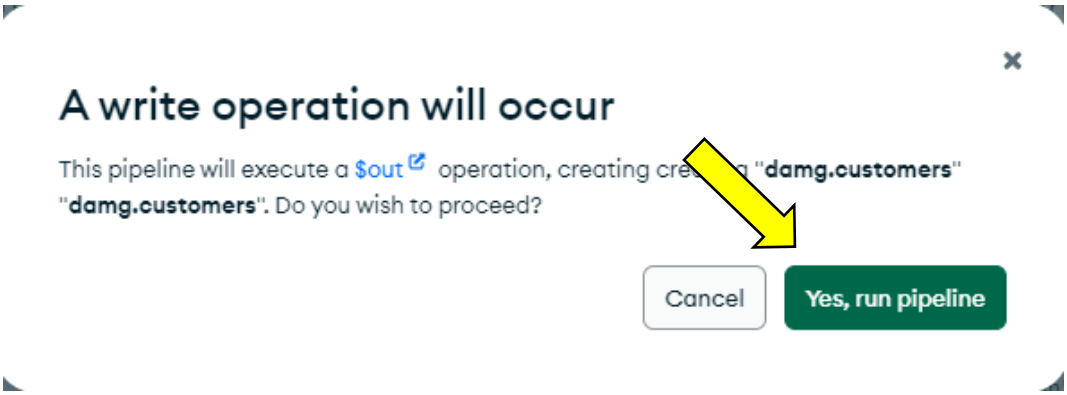
Include CustomerID, LastName, FirstName, EmailAddress, and SalesOrderIDs.

3.3 Define $out Stage, set the output collection to customers.

3.4 Click on "Run" to execute the pipeline and create the customers collection.

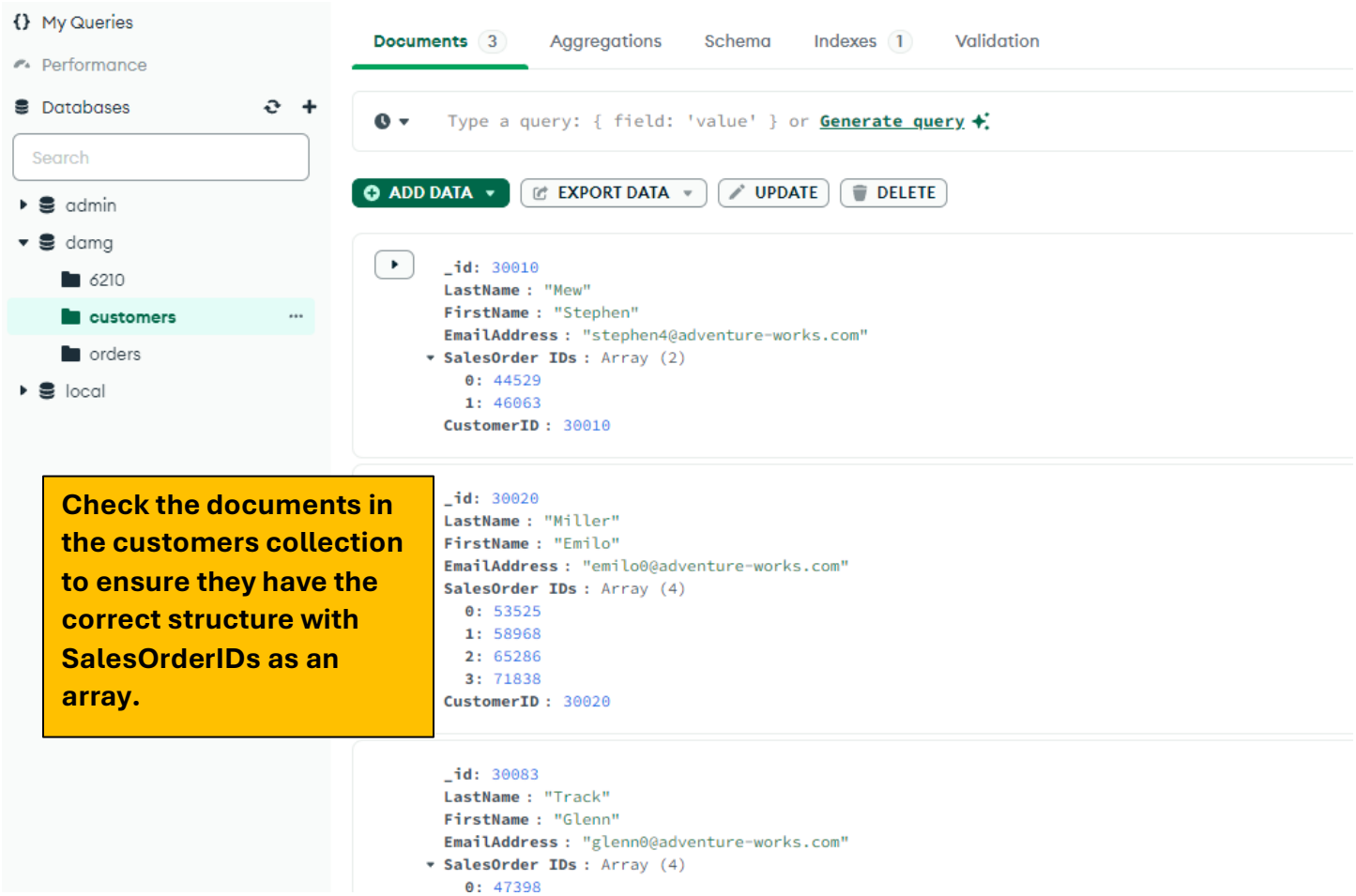


3.5 A confirmation dialog will appear, indicating that a write operation will occur. Click "Yes, run pipeline" to proceed.



4. Verifying the "Customer" collection

4.1 Verify that the customers collection has been created under your database.Check the documents in the customers collection to ensure they have the correct structure with SalesOrderIDs as an array.



Check the documents in the customers collection to ensure they have the correct structure with SalesOrderIDs as an array.

5.Use Aggregation Pipeline to create the "orders" collection

5.1 Navigate to the 6210 Collection (original data collection), click on the "Aggregations" Tab and add the $project Stage.

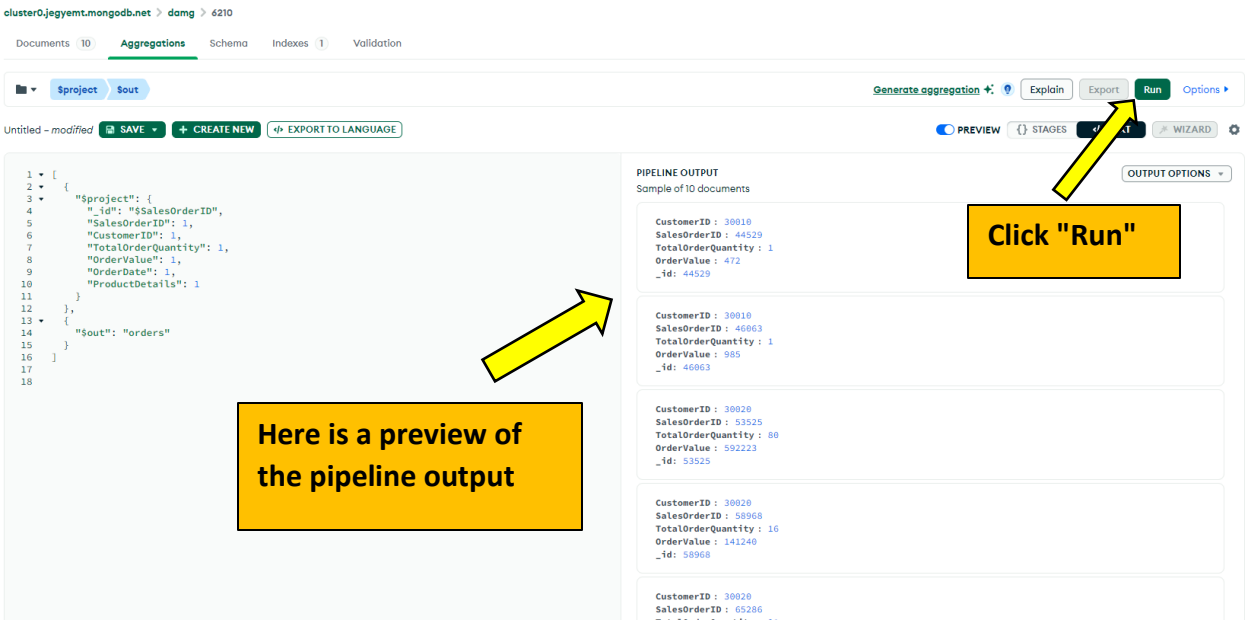
Project the Fields:

Include SalesOrderID, CustomerID, TotalOrderQuantity, OrderValue, OrderDate, and ProductDetails.

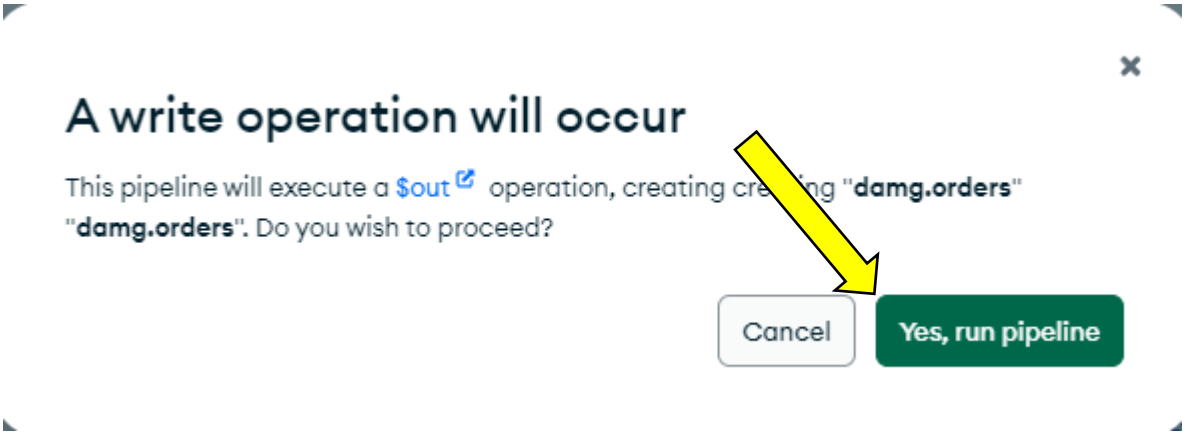
Set _id to $SalesOrderID.

5.2 Add the $out Stage, set the output collection to orders.

5.3 Click on "Run" to execute the pipeline and create the orders collection.



5.4 A confirmation dialog will appear, indicating that a write operation will occur. Click "Yes, run pipeline" to proceed.



6. Verifying the "orders" collection

6.1 Verify that the orders collection has been created under your database. Check the documents in the orders collection to ensure they have the correct structure with the necessary fields.

Documents 10 | Aggregations | Schema | Indexes 1 | Validation

Type a query: { field: 'value' } or Generate query

ADD DATA | EXPORT DATA | UPDATE | DELETE

```
_id: 44529
CustomerID : 30010
SalesOrderID : 44529
TotalOrderQuantity : 1
OrderValue : 472
```

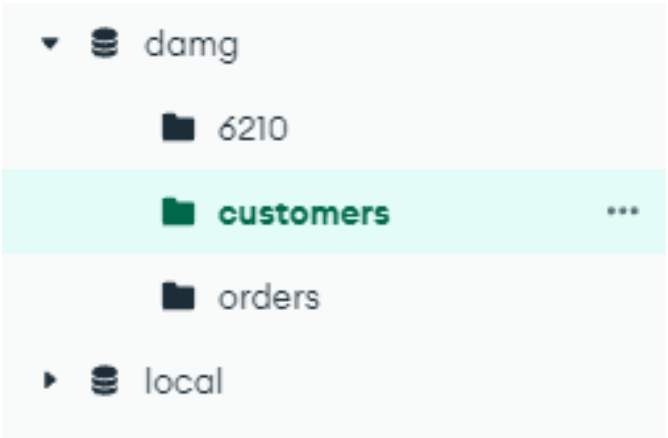
```
_id: 46063
CustomerID : 30010
SalesOrderID : 46063
TotalOrderQuantity : 1
OrderValue : 985
```

Check the documents in the orders collection to ensure they have the correct structure with the necessary fields.

```
_id: 53525
CustomerID : 30020
SalesOrderID : 53525
TotalOrderQuantity : 80
OrderValue : 592223
```

```
_id: 58968
CustomerID : 30020
SalesOrderID : 58968
TotalOrderQuantity : 16
OrderValue : 141240
```

```
_id: 65286
CustomerID : 30020
SalesOrderID : 65286
TotalOrderQuantity : 21
OrderValue : 134777
```



Now we have "customers" collection and "orders" collection

7. Before Calculation, verify your processed data again (Common to both Approach1 and Approach 2)

"customers" Collection in JSON format


```
[{
  "_id": 30010,
  "LastName": "Mew",
  "FirstName": "Stephen",
  "EmailAddress": "stephen4@adventure-works.com",
  "SalesOrder IDs": [
    44529,
    46063
  ],
  "CustomerID": 30010
},{
  "_id": 30020,
  "LastName": "Miller",
  "FirstName": "Emilo",
  "EmailAddress": "emilo0@adventure-works.com",
  "SalesOrder IDs": [
    53525,
    58968,
    65286,
    71838
  ],
  "CustomerID": 30020
},{
  "_id": 30083,
  "LastName": "Track",
  "FirstName": "Glenn",
  "EmailAddress": "glenn0@adventure-works.com",
  "SalesOrder IDs": [
    47398,
    48339,
    49496,
    50689
  ],
  "CustomerID": 30083
}]
```

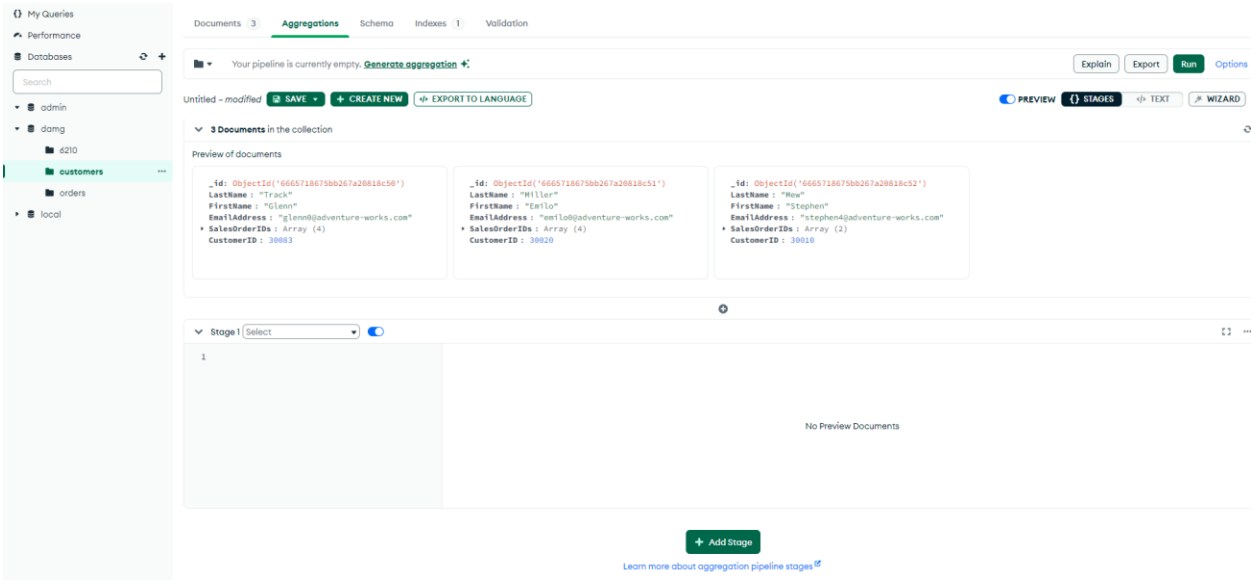

"orders" Collection in JSON format

```
[{
  "_id": 44529,
  "CustomerID": 30010,
  "SalesOrderID": 44529,
  "TotalOrderQuantity": 1,
  "OrderValue": 472
},{
  "_id": 46063,
  "CustomerID": 30010,
  "SalesOrderID": 46063,
  "TotalOrderQuantity": 1,
  "OrderValue": 985
},{
  "_id": 53525,
  "CustomerID": 30020,
  "SalesOrderID": 53525,
  "TotalOrderQuantity": 80,
  "OrderValue": 592223
},{
  "_id": 58968,
  "CustomerID": 30020,
  "SalesOrderID": 58968,
  "TotalOrderQuantity": 16,
  "OrderValue": 141240
},{
  "_id": 65286,
  "CustomerID": 30020,
  "SalesOrderID": 65286,
  "TotalOrderQuantity": 21,
  "OrderValue": 134777
},{
  "_id": 71838,
  "CustomerID": 30020,
  "SalesOrderID": 71838,
  "TotalOrderQuantity": 17,
  "OrderValue": 103238
},{
  "_id": 47398,
  "CustomerID": 30083,
  "SalesOrderID": 47398,
  "TotalOrderQuantity": 152,
  "OrderValue": 3597824
```

```
},{
  "_id": 48339, ···
},{
  "_id": 49496,
  "CustomerID": 30083,
  "SalesOrderID": 49496,
  "TotalOrderQuantity": 72,
  "OrderValue": 1428758
},{
  "_id": 50689,
  "CustomerID": 30083,
  "SalesOrderID": 50689,
  "TotalOrderQuantity": 120,
  "OrderValue": 2343950
}]
```


8. Calculate total using Aggregation Pipeline (Common to both Approach1 and Approach 2)

8.1 Navigate to the customers collection, navigate to the "Aggregations" tab within your selected collection.



8.2 Add the $lookup Stage:

Define the Lookup Parameters:

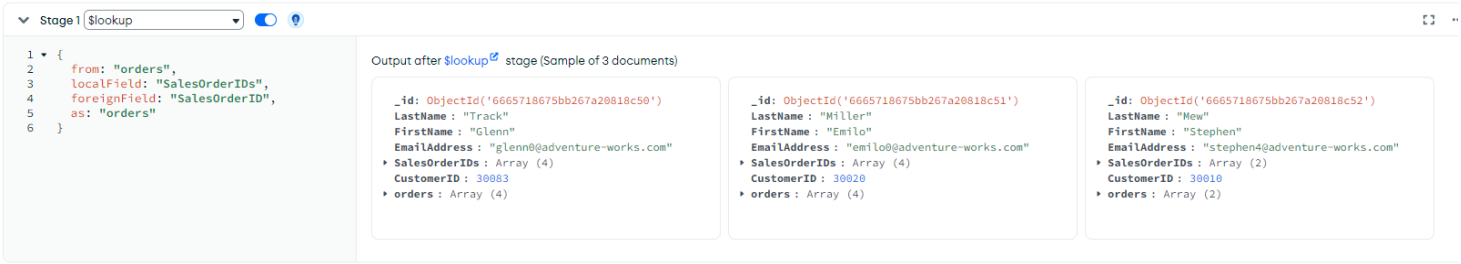
from: Set to "orders".

localField: Set to "SalesOrderIDs".

foreignField: Set to "SalesOrderID".

as: Set to "orders".

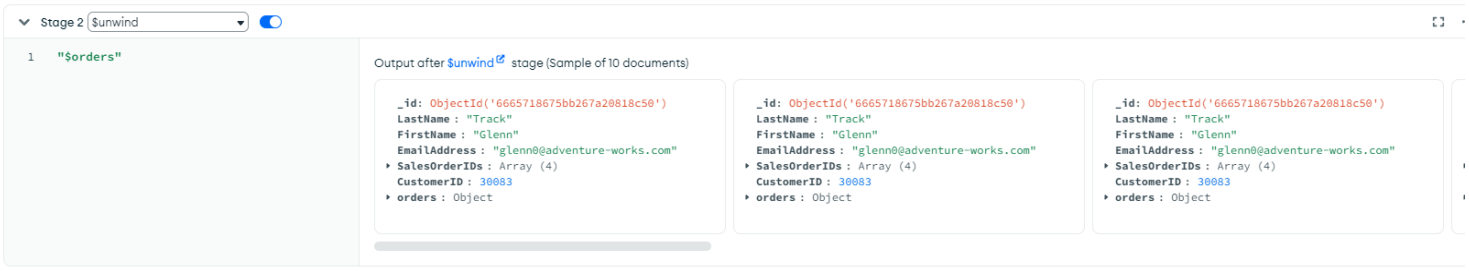
click on "Run" to execute the pipeline and verify that the output includes the orders array within each customer document.



8.3 Add the $unwind Stage:

Set the path to "$orders".

Click on "Run" to execute the pipeline and verify that the output includes each order as a separate document.



8.4 Adding the $group Stage

Define the Grouping Key:

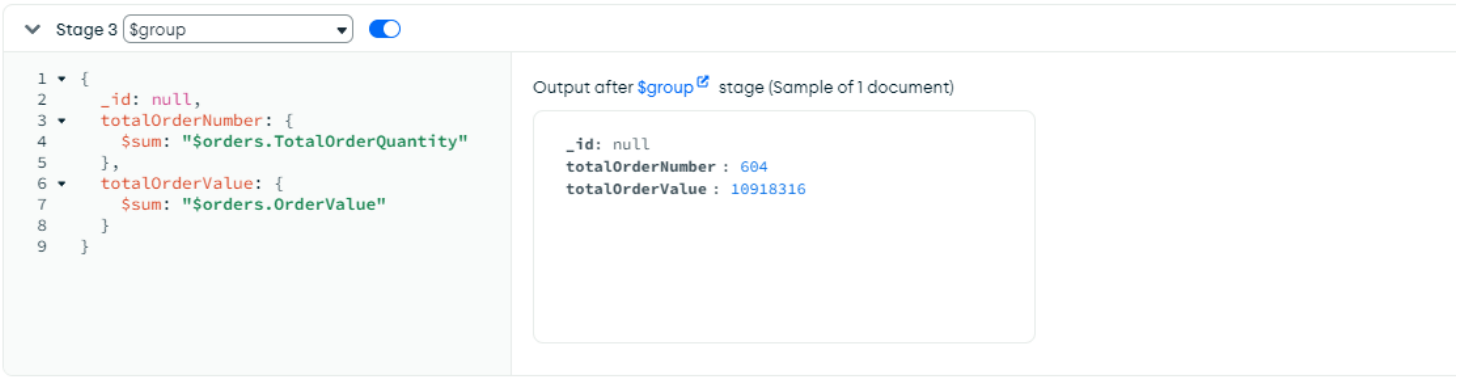
_id: null

Aggregate Fields:

totalOrderQuantity: Use $sum to calculate the total of TotalOrderQuantity.

totalOrderValue: Use $sum to calculate the total of OrderValue.

Click on "Run" to execute the pipeline and check the output to ensure it includes the aggregated data for each customer.



9. Final Output

9.1 Examine the Aggregation Results, check the final output to ensure that it includes all the required fields. Verify that the results achieve the task goal

PIPELINE OUTPUT

OUTPUT OPTIONS

Sample of 1 document

```
_id: null
totalOrderNumber : 604
totalOrderValue : 10918316
```

Final Output. Verify that the results achieve the task goal