UTM UNIVERSITI TEKNOLOGI MALAYSIA

SCHOOL OF COMPUTING
Faculty of Engineering

GROUP PROJECT

MCSD1123 – 01
BIG DATA MANAGEMENT

Lecturer's Name: Dr. Nor Haizan Mohamed Radzi

Student's Name:

Ahmed Hashim Taha Salim	(MCS211041)
Divyasini A.Konasegar	(MCS222002)
Ibrahim Sunusi Alkali	(MCS222003)

TABLE OF CONTENTS

1.0 Executive Summary

1.1 Introduction

After completed the Adventure works database in previous project, and applied SQL queries based on a given scenario that was proposed, now we'd like to develop a No-SQL document-based database using MongoDB Compass application and apply similar aggregations.

The very first step was exporting our tables as a JSON format from 'adventureworks' database stored in MySQL Workbench graphical tool.

2.0 Document design

The choice between embedded and referenced document design in MongoDB compass would depend on the relationships between the tables and the data access patterns in our application.

In general, embedded documents are useful for modeling one-to-many relationships where the related data will always be retrieved together with the parent document. While reference design is useful for modeling many-to-many relationships, where the related data is not always needed. In such cases, references (ObjectIDs) are used to link documents.

However, in our case as we using the AdventureWork database from SQL, it seems like a combination of both embedded and referenced document design would be appropriate, based on the relationships between our tables and the data access patterns in our application.

The tables "sales_2015", "sales_2016" and "sales_2017" have been stored as separate documents in a "sales" collection and referenced from the "customers" and "products" collections.

The tables “products_categories”, “products_subcategories” and “products” were embedded in a separate document in “products” collection.

The "returns", “customers” and "territories" tables have been stored as a separate collections and referenced from the "sales" collection. While the table “calendar” was excluded as now we have the dates in sales collection.

However, as we had ten different tables in SQL database, now we only have five collections in our document-based database named “adventureworks” that involves our nine tables we discussed above. These collections can be summarized in the Table 1. And Figure 1. Below

No.	**Collection**	**Contains objects related to**	**JSON Tables**
1	Customers	**_id**, **CustomerKey**, Prefix, FirstName, LastName, BirthDate, MaritalStatus, Gender, EmailAddress, AnnualIncome, TotalChildren, EducationLevel, Occupation, HomeOwner	Customers table
2	Products	**_id, ProductKey**, ProductSKU, ProductName, ModelName, ProductDescription, ProductColor, ProductSize, ProductStyle, ProductCost, ProductPrice, **ProductSubcategoryKey**, SubcategoryName, **ProductCategoryKey**, CategoryName	Products table Subcategories table Categories table

3	Sales	**_id**, OrderDate, StockDate, **OrderNumber**, ProductKey, CustomerKey, TerritoryKey, OrderLineItem, OrderQuantity	Sales_2015 table Sales_2016 table Sales_2017 table
4	Returns	**_id**, ReturnDate, **TerritoryKey**, ProductKey, ReturnQuantity	Returns table
5	Territories	**_id, TerritoryKey,** Region**,** Country**,** Continent	Territories table

Table 1: Collections

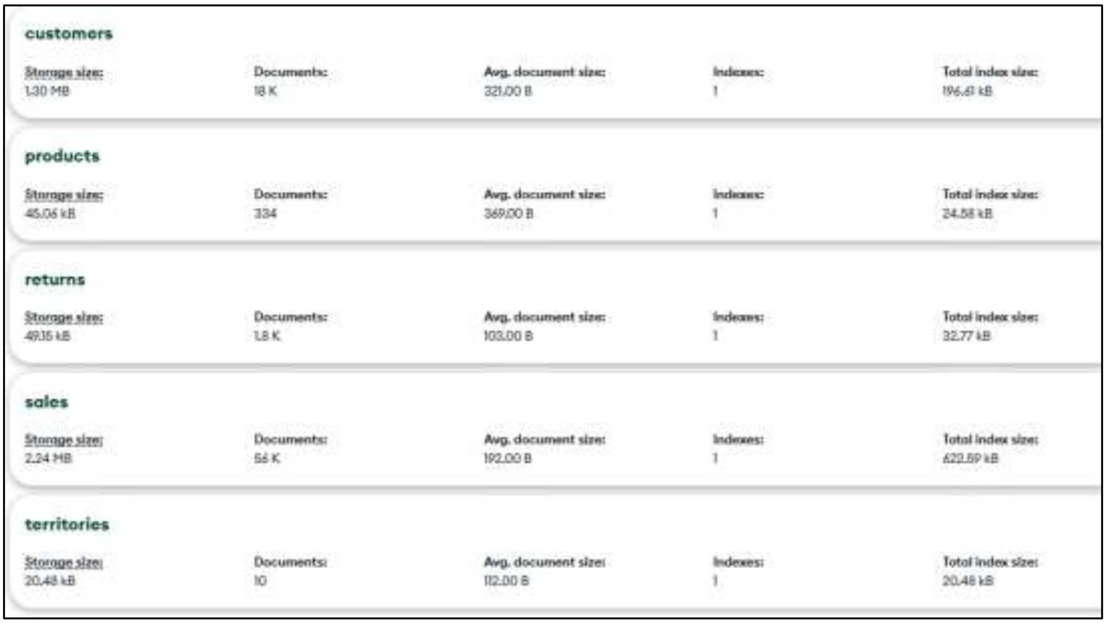


Figure 1. Collections

3.0 Queries

While SQL queries are written in Structured Query Language (SQL), which is used for relational databases. MongoDB is a NoSQL database and uses a different query language, MongoDB Query Language (MQL).

Before obtaining any aggregation stages, we created indexes on fields that we will frequently search, sort, or group by within our collections. By creating an index on these fields, MongoDB will use the index to locate the relevant documents for the query, which will result in faster query execution times. As we are using similar queries to those in SQL project, these fields are equivalent to Primary Keys.

3.1 Find all the products profit and identify them by their names in ascending order

```
[
  {
    $project: {
      ProductName: 1,
      ProductCost: 1,
      ProductPrice: 1,
      Profit: {
        $subtract: [
          "$ProductPrice",
          "$ProductCost",
        ],
      },
    },
  },
  {
    $sort: {
      Profit: -1,
    },
  },
]
```

Figure 1: MQL Output Query 1

```sql
/* 2. Find all the products profit and identify th
SELECT ProductName, ProductCost, ProductPrice,
ProductPrice-ProductCost AS Profit
FROM products ORDER BY profit DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

ProductName	ProductCost	ProductPrice	Profit
"Mountain-100 Silver, 38"	1912.1544	3399.9900	1487.8356
"Mountain-100 Silver, 42"	1912.1544	3399.9900	1487.8356
"Mountain-100 Silver, 44"	1912.1544	3399.9900	1487.8356
"Mountain-100 Silver, 48"	1912.1544	3399.9900	1487.8356
"Mountain-100 Black, 38"	1898.0944	3374.9900	1476.8956
"Mountain-100 Black, 42"	1898.0944	3374.9900	1476.8956
"Mountain-100 Black, 44"	1898.0944	3374.9900	1476.8956
"Mountain-100 Black, 48"	1898.0944	3374.9900	1476.8956
"Road-150 Red, 62"	2171.2942	3578.2700	1406.9758
"Road-150 Red, 44"	2171.2942	3578.2700	1406.9758
"Road-150 Red, 48"	2171.2942	3578.2700	1406.9758
"Road-150 Red, 52"	2171.2942	3578.2700	1406.9758
"Road-150 Red, 56"	2171.2942	3578.2700	1406.9758
"Mountain-200 Silver, 38"	1117.8559	2071.4196	953.5637
"Mountain-200 Silver, 42"	1117.8559	2071.4196	953.5637
"Mountain-200 Silver, 46"	1117.8559	2071.4196	953.5637
"Mountain-200 Black, 38"	1105.8100	2049.0982	943.2882
"Mountain-200 Black, 42"	1105.8100	2049.0982	943.2882
"Mountain-200 Black, 46"	1105.8100	2049.0982	943.2882
"Road-250 Red, 44"	1518.7864	2443.3500	924.5636
"Road-250 Red, 48"	1518.7864	2443.3500	924.5636
"Road-250 Red, 52"	1518.7864	2443.3500	924.5636
"Touring-1000 Yellow, 46"	1481.9379	2384.0700	902.1321
"Touring-1000 Yellow, 50"	1481.9379	2384.0700	902.1321

Figure 2: SQL Output Query 1

3.2 List all the customers that their annual income is less than 20,000 and bought products in 2015.

```
[
  {
    $lookup: {
      from: "products",
      localField: "ProductKey",
      foreignField: "ProductKey",
      as: "products",
    },
  },
  {
    $unwind: "$products",
  },
  {
    $lookup: {
      from: "customers",
      localField: "CustomerKey",
      foreignField: "CustomerKey",
      as: "customers",},},
  {
    $unwind: "$customers",
  },
  {
    $match: {
      "customers.AnnualIncome": {
        $lt: 20000},},},
  {
    $addFields: {
      OrderDate: {
        $toDate: "$OrderDate",
      },
    },
  },
  {
    $project: {
      FirstName: "$customers.FirstName",
      LastName: "$customers.LastName",
      AnnualIncome: "$customers.AnnualIncome",
      ProductName: "$products.ProductName",
      Year: {
        $year: "$OrderDate",
      },},},]
```

PIPELINE OUTPUT
Sample of 10 documents

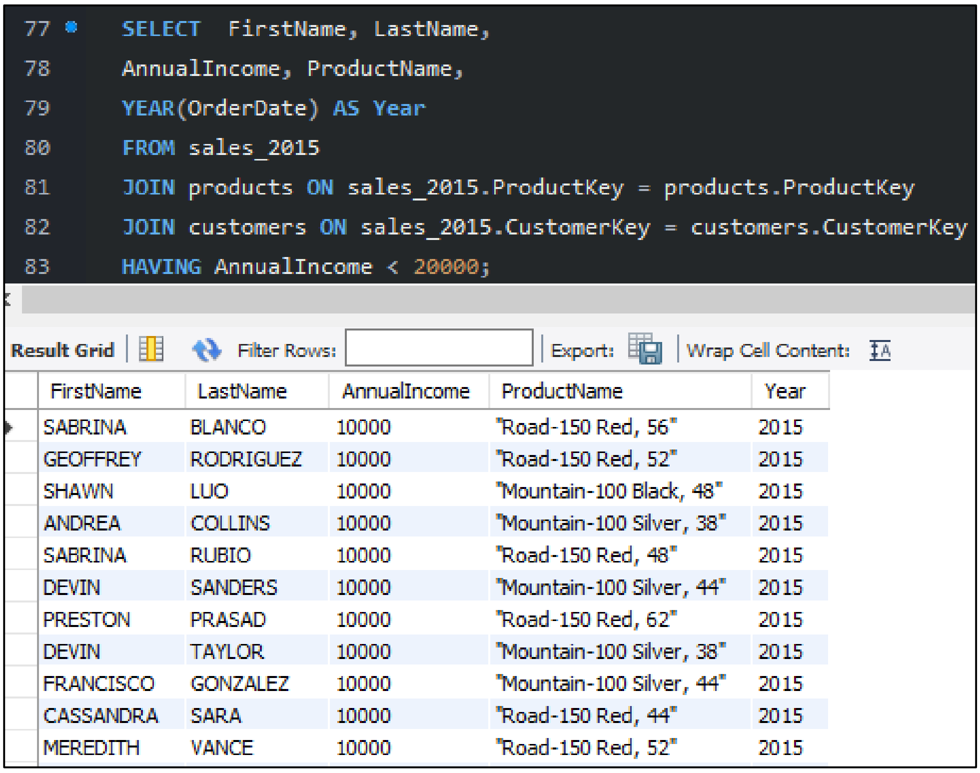
```
_id: ObjectId('63dcb07e6b82cb33bce3c407')
FirstName: "SABRINA"
LastName: "BLANCO"
AnnualIncome: 10000
ProductName: ""Road-150 Red, 56""
Year: 2015
```

```
_id: ObjectId('63dcb07e6b82cb33bce3c40c')
FirstName: "GEOFFREY"
LastName: "RODRIGUEZ"
AnnualIncome: 10000
ProductName: ""Road-150 Red, 52""
Year: 2015
```

```
_id: ObjectId('63dcb07e6b82cb33bce3c411')
FirstName: "SHAWN"
LastName: "LUO"
AnnualIncome: 10000
ProductName: ""Mountain-100 Black, 48""
Year: 2015
```

```
_id: ObjectId('63dcb07e6b82cb33bce3c413')
FirstName: "ANDREA"
LastName: "COLLINS"
AnnualIncome: 10000
ProductName: ""Mountain-100 Silver, 38""
```

Figure 3: MQL Output Query 2



```sql
SELECT  FirstName, LastName,
AnnualIncome, ProductName,
YEAR(OrderDate) AS Year
FROM sales_2015
JOIN products ON sales_2015.ProductKey = products.ProductKey
JOIN customers ON sales_2015.CustomerKey = customers.CustomerKey
HAVING AnnualIncome < 20000;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

FirstName	LastName	AnnualIncome	ProductName	Year
SABRINA	BLANCO	10000	"Road-150 Red, 56"	2015
GEOFFREY	RODRIGUEZ	10000	"Road-150 Red, 52"	2015
SHAWN	LUO	10000	"Mountain-100 Black, 48"	2015
ANDREA	COLLINS	10000	"Mountain-100 Silver, 38"	2015
SABRINA	RUBIO	10000	"Road-150 Red, 48"	2015
DEVIN	SANDERS	10000	"Mountain-100 Silver, 44"	2015
PRESTON	PRASAD	10000	"Road-150 Red, 62"	2015
DEVIN	TAYLOR	10000	"Mountain-100 Silver, 38"	2015
FRANCISCO	GONZALEZ	10000	"Mountain-100 Silver, 44"	2015
CASSANDRA	SARA	10000	"Road-150 Red, 44"	2015
MEREDITH	VANCE	10000	"Road-150 Red, 52"	2015

Figure 4: SQL Output Query 2

3.3 List all customers and their order quantities in the year 2017

```
[
  {
    $lookup: {
      from: "customers",
      localField: "CustomerKey",
      foreignField: "CustomerKey",
      as: "customer_info",}},
  {
    $lookup: {
      from: "products",
      localField: "ProductKey",
      foreignField: "ProductKey",
      as: "product_info",
    },
  },
  {
    $unwind: "$customer_info",
  },
  {
    $unwind: "$product_info",
  },
  {
    $addFields: {
      OrderDate: {
        $toDate: "$OrderDate",
      },
    },
  },
  {
    $match: {
      OrderDate: {
        $gte: Date("2017-01-01"),
        $lt: Date("2018-01-01"),
      },
    },
  },
  {
    $group: {
      _id: "$customer_info.CustomerKey",
      FirstName: {
        $first: "$customer_info.FirstName",
      },
      LastName: {
        $first: "$customer_info.LastName",
      },
      ProductName: {
        $first: "$product_info.ProductName",
      },
      OrderQuantity: {
        $sum: "$OrderQuantity",
      },
      OrderDate: {
        $first: "$OrderDate"}}},
  {$sort: {OrderQuantity: -1}},
  {
    $project: {
      FirstName: 1,
      LastName: 1,
      ProductName: 1,
      OrderQuantity: 1,
      Year: {
        $year: "$OrderDate",},
      _id: 0}}
]
```

Figure 5: MQL Output Query 3

3.4 Count the products that purchased the same item in all years.

```
[{$lookup: {
      from: "customers",
      localField:
"CustomerKey",
      foreignField:
"CustomerKey",
      as: "customer_info",
    },
  },
  {
    $lookup: {
      from: "products",
      localField:
"ProductKey",
      foreignField:
"ProductKey",
      as: "product_info",
    },
  },
  {
    $unwind:
"$customer_info",
  },
  {
    $unwind:
"$product_info",
  },
  {
    $group: {
      _id:
"$product_info.ProductName"
,
      quantity_sold: {
        $sum:
"$OrderQuantity",
      },
    },
  },
  {
    $project: {
      _id: 0,
      ProductName: "$_id",
      quantity_sold: 1,
    },
  },
  {
    $sort: {
      quantity_sold: -1,
    },
  },
]
```

Figure 6: MQL Output Query 4

3.5 Count the returned products group by region.

```
[
  {
    $lookup: {
      from: "territories",
      localField:
"TerritoryKey",
      foreignField:
"TerritoryKey",
      as: "territory_info",
    },
  },
  {
    $unwind:
"$territory_info",
  },
  {
    $group: {
      _id:
"$territory_info.Region",
      Total_Return: {
        $sum: 1,
      },
    },
  },
  {
    $sort: {
      Total_Return: -1,
    },
  },
  {
    $project: {
      Total_Return: 1,
      Region: "$_id",
      _id: 0,
    },
  },
]
```

PIPELINE OUTPUT

Sample of 8 documents

```
Total_Return: 400
Region: "Australia"

Total_Return: 354
Region: "Southwest"

Total_Return: 269
Region: "Northwest"

Total_Return: 234
Region: "Canada"

Total_Return: 204
Region: "United Kingdom"

Total_Return: 186
Region: "France"
```

Figure 7: MQL Output Query 5

```sql
SELECT count(*) AS Total_Return, Region
FROM returns
JOIN territories ON
returns.TerritoryKey = territories.TerritoryKey
GROUP BY region;
```

sult Grid | Filter Rows: | Export: | Wrap

Total_Return	Region
269	Northwest
354	Southwest
1	Southeast
234	Canada
186	France
161	Germany
400	Australia
204	United Kingdom

Figure 8: SQL Output Query 6

3.6 Find out the profit of the top 5 products for 2017.

```
[
  {
    $addFields: {
      OrderDate: {
        $toDate: "$OrderDate",
      },
    },
  },
  {
    $match: {
      OrderDate: {
        $gte: Date("2017-01-01"),
        $lt: Date("2018-01-01"),
      },
    },
  },
  {
    $lookup: {
      from: "products",
      localField: "ProductKey",
      foreignField: "ProductKey",
      as: "product_info",
    },
  },
  {
    $unwind: "$product_info",
  },
  {
    $addFields: {
      Profit: {
        $subtract: [
     "$product_info.ProductPrice",
          "$product_info.ProductCost"],
      },
    },
  },
  {
    $project: {
      ProductKey:
"$product_info.ProductKey",
      ProductName:
"$product_info.ProductName",
      ProductCost:
"$product_info.ProductCost",
      ProductPrice:
"$product_info.ProductPrice",
      Profit: 1,
      Year: {
        $year: "$OrderDate",
      },
      _id: 0,
    },
  },
  {
    $limit: 5,
  },
]
```

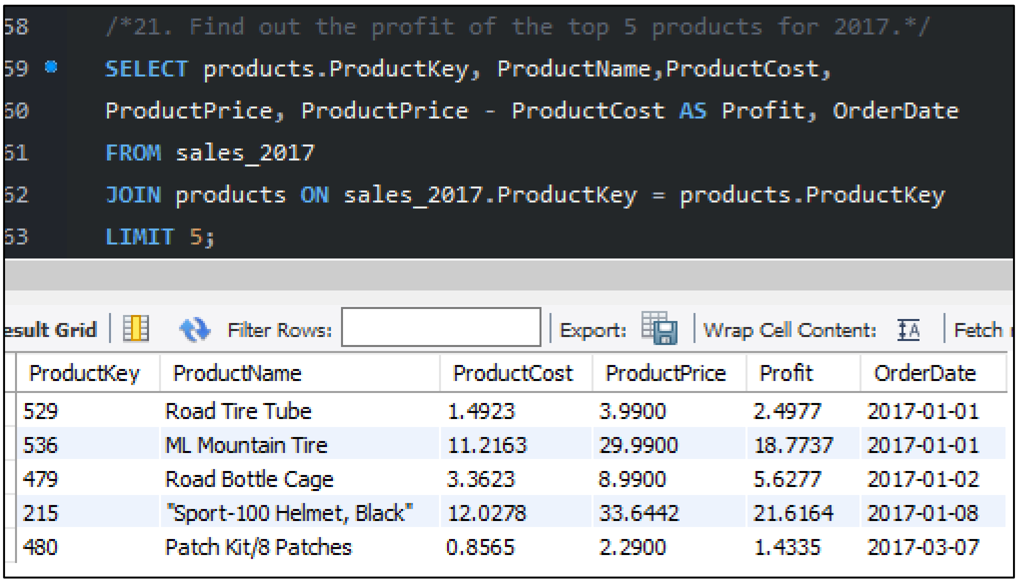
PIPELINE OUTPUT
Sample of 5 documents

```
Profit: 2.4977
ProductKey: 529
ProductName: "Road Tire Tube"
ProductCost: 1.4923
ProductPrice: 3.99
Year: 2017
```

```
Profit: 18.773699999999998
ProductKey: 536
ProductName: "ML Mountain Tire"
ProductCost: 11.2163
ProductPrice: 29.99
Year: 2017
```

```
Profit: 5.6277000000000001
ProductKey: 479
ProductName: "Road Bottle Cage"
ProductCost: 3.3623
ProductPrice: 8.99
Year: 2017
```

Figure 9: MQL Output Query 6



```sql
/*21. Find out the profit of the top 5 products for 2017.*/
SELECT products.ProductKey, ProductName,ProductCost,
ProductPrice, ProductPrice - ProductCost AS Profit, OrderDate
FROM sales_2017
JOIN products ON sales_2017.ProductKey = products.ProductKey
LIMIT 5;
```


ProductKey	ProductName	ProductCost	ProductPrice	Profit	OrderDate
529	Road Tire Tube	1.4923	3.9900	2.4977	2017-01-01
536	ML Mountain Tire	11.2163	29.9900	18.7737	2017-01-01
479	Road Bottle Cage	3.3623	8.9900	5.6277	2017-01-02
215	"Sport-100 Helmet, Black"	12.0278	33.6442	21.6164	2017-01-08
480	Patch Kit/8 Patches	0.8565	2.2900	1.4335	2017-03-07

Figure 10: SQL Output Query 6

3.7 Find the total returns in each year (2015, 2016, 2017

```
[{$addFields: {
      ReturnDate: {
        $toDate: "$ReturnDate"}}},
  {$facet: {
      2015: [{
          $match: {
            ReturnDate: {
              $gte: new Date("2015-01-01"),
              $lte: new Date("2015-12-31")}}}},
        {$group: {
            _id: null,
            Total_Returns: {
              $sum: "$ReturnQuantity"}}},
        {$project: {
            Year: {
              $literal: "2015",
            },
            Total_Returns: 1,
            _id: 0}}],
      2016: [{
          $match: {
            ReturnDate: {
              $gte: new Date("2016-01-01"),
              $lte: new Date("2016-12-31")}}}},
        {
          $group: {
            _id: null,
            Total_Returns: {
              $sum: "$ReturnQuantity",
            },
          },
        },
        {
          $project: {
            Year: {
              $literal: "2016",
            },
            Total_Returns: 1,
            _id: 0}}],
      2017: [
        {
          $match: {
            ReturnDate: {
              $gte: new Date("2017-01-01"),
              $lte: new Date("2017-12-31")}}}},
        {
          $group: {
            _id: null,
            Total_Returns: {
              $sum: "$ReturnQuantity"}}},
        {
          $project: {Year: {$literal: "2017"},
            Total_Returns: 1,
            _id: 0,}}]}},
  {$project: {
      results: {
        $concatArrays: [
          "$2015",
          "$2016",
          "$2017"]}}},
  {
    $unwind: "$results",
  },
  {$replaceRoot: {newRoot: "$results"}}]
```

```
Total_Returns: 86
Year: "2015"
```

```
Total_Returns: 770
Year: "2016"
```

```
Total_Returns: 972
Year: "2017"
```

Figure 11: MQL Output Query 7

```sql
SELECT '2015' AS Year, SUM(ReturnQuantity) AS Total_Returns FROM returns
WHERE ReturnDate BETWEEN '2015-01-01' AND '2015-12-31'
UNION ALL
SELECT '2016' AS Year, SUM(ReturnQuantity) AS Total_Returns FROM returns
WHERE ReturnDate BETWEEN '2016-01-01' AND '2016-12-31'
UNION ALL
SELECT '2017' AS Year, SUM(ReturnQuantity) AS Total_Returns FROM returns
WHERE ReturnDate BETWEEN '2017-01-01' AND '2017-12-31';
```

esult Grid | Filter Rows: | Export: | Wrap Cell Content:

Year	Total_Returns
2015	86
2016	770
2017	972

Figure 12: SQL Output Query 7

4.0 Data-Models Discussion

Relational data models such as MySQL workbench are good at storing structured and related data, where data is organized into tables with relationships defined between them. They provide a number of advantages, such as enforcing data integrity and consistency through foreign keys, and enabling complex queries and transactions.

On the other hand, document-based data models, such as MongoDB compass, store data in semi-structured or unstructured format, in the form of documents (key-value pairs), which can be nested and embedded. These data models are more flexible and scalable, as they can store any kind of data without having to define a fixed schema beforehand, and can handle large amounts of unstructured data.

When considering which model is more applicable, it's important to think about the specific needs of the case study. In our case, the dataset “Adventure Works” is highly structured, with well-defined relationships thus, a relational SQL data model may be a better choice.

In summary, the choice between a relational and a document-based data model should be based on the specific needs of the case study, and the strengths and limitations of each model should be taken into account. Relational data model is advisable to dataset’s that are structured and well defined relationships same to this case study, while data that is semi-structured or unstructured, and requires scalability and flexibility, a document-based data model may be a better option.