POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Systems and Methods for Big and Unstructured Data Project

Author(s): **Ferraù Francesco (10709375)**
Gigante Davide (11018245)

Academic Year: 2023-2024

Contents

1 | Introduction

1.1. Problem Statement

In the rapidly evolving digital landscape, e-commerce platforms generate vast amounts of diverse data, encompassing customer profiles, transaction histories, product details, and purchasing behaviors. This wealth of information, if harnessed correctly, can provide invaluable insights into customer preferences, sales trends, and inventory management, driving informed business decisions and personalized marketing strategies. However, the challenge lies in effectively managing and analyzing this heterogeneous and voluminous data. Our objective is to develop a robust system that can store, process, and analyze the complex and varied data generated by an online shopping platform, enabling real-time insights and data-driven decision-making.

1.2. Dataset Overview

The chosen dataset for this endeavor is the Online Shopping Dataset. This dataset encapsulates a comprehensive range of e-commerce data points that will be explored in the Chapter 3. Its attributes vary significantly in type and structure, ranging from simple numeric values (like CustomerID) to more complex text descriptions (like Product_Description). The dataset's intrinsic diversity in data types and the potential for high-volume transactional data make it an ideal candidate for demonstrating the effectiveness of our chosen technology solution.

1.3. Selection of Database Technology: NoSQL Approach

After careful consideration of the dataset's characteristics, a Non-Relational Database (NoSQL) was chosen, specifically MongoDB, for the following reasons.

1.3.1. Flexibility in Data Modeling

The schema-less nature of MongoDB is particularly advantageous for our dataset. Unlike traditional relational databases, MongoDB can handle varied and evolving data structures. This flexibility is essential for accommodating the diverse data types and complex structures inherent in the dataset.

1.3.2. Scalability and Performance

With the anticipated high volume of data and the need for rapid querying and real-time analysis, MongoDB offers superior scalability and performance. It efficiently handles large volumes of data and complex queries, which is crucial for a dynamic e-commerce environment.

1.3.3. Handling Unstructured and Semi-Structured Data

The dataset includes a mix of structured, unstructured, and semi-structured data. MongoDB's document-oriented approach is well-suited for such diversity, allowing for the integration of all these data types in a more unified and accessible format.

1.3.4. Agile Development and Iterative Changes

The e-commerce domain requires agility in development and the ability to adapt to changes quickly. MongoDB's flexible data model allows for rapid iteration and evolution of the database structure, aligning well with the agile development methodologies.

2 | Data Wrangling

Data wrangling, an integral part of data science, involves cleaning and unifying messy and complex data sets for easy access and analysis. A critical step in this process is the handling of null values. Null values, often representing missing, undefined, or inappropriate data, can significantly impact the quality and integrity of datasets. This chapter delves into why eliminating or appropriately managing null values is essential for reliable data analysis.

2.1. Impact of Null Values

2.1.1. Statistical Analysis Complications

Null values can skew and bias statistical analyses. Most statistical models assume that the data is complete and distributions are correctly represented. Null values can distort these assumptions, leading to inaccurate conclusions.

2.1.2. Data Integrity Concerns

Null values can indicate deeper issues such as data corruption, loss during transfer, or improper data entry. Their presence can raise questions about the overall integrity and reliability of the dataset.

2.1.3. Challenges in Data Aggregation

Aggregating data with null values can be problematic. Operations like sum, average, or count might return inaccurate results, leading to flawed business insights and decisions.

2.2. Strategy Applied for Handling Null Values

2.3. Practical Application: Null Value Removal in MongoDB

The provided Python script has been used to connect to our MongoDB database and remove documents that contain null values in specified fields.

```
# Connect to db
client = MongoClient('mongodb://localhost:27017')
db = client.shopping_cart
collection = db.shopping_cart_connection

# Print info to verify the server is on
print(client.server_info())

# Optionally, to check the number of documents before deletion
initial_docs_count = collection.count_documents({})
print(f"Documents remaining after deletion:
   {initial_docs_count}") # total: 52955

# List of fields to check for null values
fields_to_check = ['CustomerID', 'Gender', 'Location',
   'Tenure_Months',
                   'Transaction_ID', 'Transaction_Date',
                      'Product_SKU',
                   'Product_Description', 'Product_Category',
                      'Quantity',
                   'Avg_Price', 'Delivery_Charges',
                      'Coupon_Status', 'GST',
                   'Date', 'Offline_Spend', 'Online_Spend',
                      'Month',
                   'Coupon_Code', 'Discount_pct']

# Iterate through each field and remove documents with null values
for field in fields_to_check:
    # This query finds documents where the field is either
       missing, is None, or is an empty string
    query = {"$or": [{field: {"$exists": False}}, {field: None},
       {field: ""}]}
    # Deletes the documents matching the query
```

```
    collection.delete_many(query)

# Optionally, to check the number of documents after deletion
remaining_docs_count = collection.count_documents({})
print(f"Documents remaining after deletion:
   {remaining_docs_count}") # total: 52524
```

2.3.1. Database Connection

The script begins by establishing a connection to a local MongoDB instance and accessing a specific database and collection.

2.3.2. Fields Specification

A list of fields (`fields_to_check`) is defined, representing the dataset's columns. These are the fields that will be checked for null values.

2.3.3. Deletion Process

The script iterates through each field in the list. For each field, it executes a query that identifies documents where the field:

- is missing;
- is None;
- Contains an empty string.

These documents are then deleted from the collection.

2.3.4. Post-Deletion Verification

After the deletion process, the script counts the remaining documents in the collection, providing a quick verification of the operation's impact.

3 | Dataset

This chapter provides a comprehensive overview of the Online Shopping Dataset (downloadable at this link), a rich collection of data points that encapsulate various aspects of customer transactions in an e-commerce setting.

3.1. Dataset Description

The Online Shopping Dataset contains detailed information about customer transactions in an e-commerce environment. This dataset includes data points such as customer details, transaction information, product specifics, and purchase metrics. Given the diverse nature of this information, which ranges from structured data like numeric IDs and dates to unstructured data like product descriptions, a NoSQL database is apt for efficient storage, retrieval, and querying of this data.

3.2. Non-Relational Database Implementation

In MongoDB, the data would be stored in a document-oriented manner. Each document in the collection would represent a single transaction with nested documents and arrays to handle complex data structures. Below is an example of how a document in this dataset might be structured:


```
{
"_id": "65957947d67f9e740096c093",
"CustomerID": 17850,
"Gender": "M",
"Location": "Chicago",
"Tenure_Months": 12,
"Transaction": {
    "Transaction_ID": 16679,
    "Transaction_Date": "2019-01-01T00:00:00.000+00:00",
```

```
        "Product": {
            "Product_SKU": "GGOENEBJ079499",
            "Product_Description": "Nest Learning Thermostat 3rd
               Gen-USA - Stainless Steel",
            "Product_Category": "Nest-USA",
            "Quantity": 1,
            "Avg_Price": 153.71
        },
        "Delivery_Charges": 6.5,
        "Coupon_Status": "Used",
        "GST": 0.1,
        "Date": "1/1/2019",
        "Offline_Spend": 4500,
        "Online_Spend": 2424.5,
        "Month": 1,
        "Coupon_Code": "ELEC10",
        "Discount_pct": 10
    }
}
```


3.3. Dataset Attributes

1. `CustomerID`
    - *Description*: Unique identifier for each customer.
    - *Data Type*: Numeric
2. `Gender`
    - *Description*: Gender of the customer (e.g., Male, Female).
    - *Data Type*: Categorical
3. `Location`
    - *Description*: Location or address information of the customer.
    - *Data Type*: Text
4. `Tenure_Months`
    - *Description*: Number of months the customer has been associated with the

platform.

- *Data Type*: Numeric

5. `Transaction_ID`
    - *Description*: Unique identifier for each transaction.
    - *Data Type*: Numeric
6. `Transaction_Date`
    - *Description*: Date of the transaction.
    - *Data Type*: Date
7. `Product_SKU`
    - *Description*: Stock Keeping Unit (SKU) identifier for the product.
    - *Data Type*: Text
8. `Product_Description`
    - *Description*: Description of the product.
    - *Data Type*: Text
9. `Product_Category`
    - *Description*: Category to which the product belongs.
    - *Data Type*: Categorical
10. `Quantity`
    - *Description*: Quantity of the product purchased in the transaction.
    - *Data Type*: Numeric
11. `Avg_Price`
    - *Description*: Average price of the product.
    - *Data Type*: Numeric
12. `Delivery_Charges`
    - *Description*: Charges associated with the delivery of the product.
    - *Data Type*: Numeric

13. `Coupon_Status`
    - *Description*: Status of the coupon associated with the transaction.
    - *Data Type*: Categorical
14. `GST`
    - *Description*: Goods and Services Tax associated with the transaction.
    - *Data Type*: Numeric
15. `Date`
    - *Description*: Date of the transaction (potentially redundant with Transaction_Date).
    - *Data Type*: Date
16. `Offline_Spend`
    - *Description*: Amount spent offline by the customer.
    - *Data Type*: Numeric
17. `Online_Spend`
    - *Description*: Amount spent online by the customer.
    - *Data Type*: Numeric
18. `Month`
    - *Description*: Month of the transaction.
    - *Data Type*: Categorical
19. `Coupon_Code`
    - *Description*: Code associated with a coupon, if applicable.
    - *Data Type*: Text
20. `Discount_pct`
    - *Description*: Percentage of discount applied to the transaction.
    - *Data Type*: Numeric

4 | Queries

4.1. Queries level easy

4.1.1. Retrieving Customer-Specific Transaction Data

This query is designed to fetch all transaction records for a specific customer from an online shopping database. By specifying `{'CustomerID' : 14457}`, the query filters out all transactions that are associated with the customer whose `CustomerID` is 14457.

Code

```
db.shoppingCartCollection.find({
    "CustomerID" : 14457
})
```

Output


```
{
  _id: ObjectId("65957948d67f9e740096d753"),
  '': 5824,
  CustomerID: 14457,
  Gender: 'F',
  Location: 'New Jersey',
  Tenure_Months: 4,
  Transaction_ID: 46392,
  Transaction_Date: 2019-12-11T00:00:00.000Z,
  Product_SKU: 'GGOEGOCL087799',
  Product_Description: 'Google Large Standard Journal Navy',
  Product_Category: 'Office',
  Quantity: 76,
  Avg_Price: 12.87,
  Delivery_Charges: 37.51,
  Coupon_Status: 'Clicked',
```

```
  GST: 0.1,
  Date: '12/11/2019',
  Offline_Spend: 4000,
  Online_Spend: 1612.93,
  Month: 12,
  Coupon_Code: 'OFF30',
  Discount_pct: 30
}
{
  _id: ObjectId("65957948d67f9e740096daf2"),
  '': 6751,
  CustomerID: 14457,
  Gender: 'F',
  Location: 'New Jersey',
  Tenure_Months: 4,
  Transaction_ID: 46392,
  Transaction_Date: 2019-12-11T00:00:00.000Z,
  Product_SKU: 'GGOEYAEB091814',
  Product_Description: 'YouTube Tee Black',
  Product_Category: 'Apparel',
  Quantity: 10,
  Avg_Price: 17.93,
  Delivery_Charges: 37.51,
  Coupon_Status: 'Used',
  GST: 0.18,
  Date: '12/11/2019',
  Offline_Spend: 4000,
  Online_Spend: 1612.93,
  Month: 12,
  Coupon_Code: 'SALE30',
  Discount_pct: 30
}
continues...
```


4.1.2. Low Delivery Charges

This query retrieves products where the delivery charges are less than a specified amount, in this case, less than 5.

Code

```
db.shoppingCartCollection.find({
        Delivery_Charges: { $lt: 5 }
        })
```

Output


```
{
  _id: ObjectId("65957947d67f9e740096c7d3"),
  '': 1856,
  CustomerID: 14849,
  Gender: 'F',
  Location: 'New Jersey',
  Tenure_Months: 45,
  Transaction_ID: 17207,
  Transaction_Date: 2019-01-05T00:00:00.000Z,
  Product_SKU: 'GGOEGFKA022299',
  Product_Description: 'Keyboard DOT Sticker',
  Product_Category: 'Office',
  Quantity: 3,
  Avg_Price: 1.55,
  Delivery_Charges: 0,
  Coupon_Status: 'Used',
  GST: 0.1,
  Date: '1/5/2019',
  Offline_Spend: 4500,
  Online_Spend: 4055.3,
  Month: 1,
  Coupon_Code: 'OFF10',
  Discount_pct: 10
}
{
  _id: ObjectId("65957947d67f9e740096c7d4"),
  '': 1857,
  CustomerID: 14849,
  Gender: 'F',
  Location: 'New Jersey',
  Tenure_Months: 45,
  Transaction_ID: 17207,
```

```
  Transaction_Date: 2019-01-05T00:00:00.000Z,
  Product_SKU: 'GGOEGOAB015899',
  Product_Description: 'Rainbow Stylus Pen',
  Product_Category: 'Office',
  Quantity: 3,
  Avg_Price: 1.55,
  Delivery_Charges: 0,
  Coupon_Status: 'Clicked',
  GST: 0.1,
  Date: '1/5/2019',
  Offline_Spend: 4500,
  Online_Spend: 4055.3,
  Month: 1,
  Coupon_Code: 'OFF10',
  Discount_pct: 10
}
continues...
```

4.1.3. Counting Transactions by CustomerID

This query counts the number of transactions per customer using the $group and $sum operators.

Code

```
db.shoppingCartCollection.aggregate({
    "$group": {
        "_id": "$CustomerID", "count": {"$sum": 1}}
        })
```

Output

```
{
  _id: 12359,
  count: 16
}
{
  _id: 14589,
  count: 3
}
```

```
{
  _id: 17742,
  count: 13
}
continues...
```

4.1.4. Total Customer Spend Calculation

This query calculates the total expenditure of each customer on the platform by summing their online and offline spending.

Code

```
db.shoppingCartCollection.aggregate({"$group": {
        "_id": "$CustomerID",
        "totalSpend": {"$sum": {"$add": ["$Online_Spend",
           "$Offline_Spend"]}}
    }})
```

Output

```
{
  _id: 17978,
  totalSpend: 78317.88
}
{
  _id: 15998,
  totalSpend: 553473.12
}
{
  _id: 16110,
  totalSpend: 698725.88
}
continues...
```

4.1.5. Most Popular Products

This query aims to identify the most popular products by calculating the total quantity sold for each product and sorting them in descending order based on the total quantity

sold.

Code

```
db.shoppingCartCollection.aggregate({"$group": {
        "_id": "$Product_SKU",
        "totalQuantitySold": {"$sum": "$Quantity"}
    }},
    {"$sort": {"totalQuantitySold": -1}})
```

Output

```
{
  _id: 'GGOEGGOA017399',
  totalQuantitySold: 16234
}
{
  _id: 'GGOEGDHC018299',
  totalQuantitySold: 9728
}
{
  _id: 'GGOEGBMJ013399',
  totalQuantitySold: 7321
}
continues...
```

4.1.6. Average Transaction Value by Month

This query calculates the average value of transactions for each month. It considers the sum of average product price, delivery charges, and GSTs for each transaction, grouping the results by month and sorting them in ascending order by month.

Code

```
db.shoppingCartCollection.aggregate({"$group": {
        "_id": "$Month",
        "averageTransactionValue": {"$avg": {"$add":
           ["$Avg_Price", "$Delivery_Charges", "$GST"]}}
    }},
    {"$sort": {"_id": 1}})
```

Output

```
{
  _id: 1,
  averageTransactionValue: 76.83856077554064
}
{
  _id: 2,
  averageTransactionValue: 68.57440677966102
}
{
  _id: 3,
  averageTransactionValue: 59.298759906759905
}
continues...
```

4.2. Queries level intermediate

4.2.1. Average Spend per Transaction by Location

This query calculates the average spend per transaction for each location by summing the online and offline spend for each transaction and then grouping the results by location to compute the average spend.

Code

```
db.shoppingCartCollection.aggregate({
        "$addFields": {
            "Total_Spend": {"$add": ["$Online_Spend",
               "$Offline_Spend"]}
        }
    },
    {
        "$group": {
            "_id": "$Location",
            "Average_Spend": {"$avg": "$Total_Spend"}
        }
    })
```

Output

```
{
  _id: 'Washington DC',
  Average_Spend: 4635.949235880399
}
{
  _id: 'Chicago',
  Average_Spend: 4726.329525219298
}
{
  _id: 'California',
  Average_Spend: 4807.530675912044
}
{
  _id: 'New Jersey',
  Average_Spend: 4653.534100671141
}
{
  _id: 'New York',
  Average_Spend: 4662.074217356043
}
```

4.2.2. Sales Performance by Product Category

This query calculates the total quantity of products sold and the total revenue generated, grouped by each product category. The results are sorted by the total quantity sold.

Code

```
db.shoppingCartCollection.aggregate([{
        $group: {
            _id: "$Product_Category",
            total_quantity_sold: { $sum: "$Quantity" },
            total_revenue: { $sum: { $multiply: ["$Quantity",
               "$Avg_Price"] } }
        }
    },
    {
        $sort: { total_quantity_sold: -1 }
```

```
    })
```

Output

```
{
  _id: 'Office',
  total_quantity_sold: 88383,
  total_revenue: 276794.4
}
{
  _id: 'Apparel',
  total_quantity_sold: 32438,
  total_revenue: 591145.8
}
{
  _id: 'Drinkware',
  total_quantity_sold: 30501,
  total_revenue: 200707.83
}
continues...
```

4.2.3. Impact of Discounts on Sales Volume and Value Code

```
db.shoppingCartCollection.aggregate([
  { $match: { Coupon_Status: "Used" } },
  { $group: {
      _id: "$Product_SKU",
      Used_Coupons_Count: { $sum: "$Quantity" }
    }
  },
  { $sort: { Used_Coupons_Count: -1 } }
])
```

Output

```
{
  _id: 'GGOEGGOA017399',
  Used_Coupons_Count: 8184
```

```
}
{
  _id: 'GGOEGDHC018299',
  Used_Coupons_Count: 3405
}
{
  _id: 'GGOEGFKQ020399',
  Used_Coupons_Count: 2416
}
{
  _id: 'GGOEGOLC014299',
  Used_Coupons_Count: 2206
}
continues...
```

4.2.4. Customer Loyalty Analysis

This query is designed to explore customer spending behavior in relation to their tenure with the online platform. It calculates the average offline and online spending for customers grouped by the number of months they have been associated with the platform. The query results are sorted by tenure,

Code

```
db.shoppingCartCollection.aggregate([
    {
        $group: {
            _id: "$Tenure_Months",
            Average_Offline_Spend: { $avg: "$Offline_Spend" },
            Average_Online_Spend: { $avg: "$Online_Spend" }
        }
    },
    {
        $sort: { _id: 1 }
    }
])
```

Output

```
{
```

```
  _id: 2,
  Average_Offline_Spend: 3042.2535211267605,
  Average_Online_Spend: 1964.1502660406886
}
{
  _id: 3,
  Average_Offline_Spend: 3100.8064516129034,
  Average_Online_Spend: 1883.6242903225807
}
{
  _id: 4,
  Average_Offline_Spend: 2823.262548262548,
  Average_Online_Spend: 1838.894691119691
}
continues...
```

4.2.5. Monthly Expenditure Analysis for a specific Customer

This query provides a detailed analysis of the monthly spending habits of a specific customer, identified by CustomerID 12347 (the CustomerID can be changed). The query begins by filtering the shopping cart database to include only the transactions of this customer. It then aggregates the data by year and month, calculating the total spend for each month. This calculation multiplies the average price of each item (Avg_Price) by the quantity purchased in each transaction, providing the monthly expenditure. The results are sorted chronologically, starting with the earliest year and month, offering a clear, time-sequenced view of the customer's spending behavior.

Code

```
db.shoppingCartCollection.aggregate([
  { $match: { CustomerID: 12347 } },
  { $group: {
      _id: { Year: { $year: "$Transaction_Date" }, Month:
         "$Month" },
      Total_Spend: { $sum: { $multiply: ["$Avg_Price",
         "$Quantity"] } }
    }
  },
  { $sort: { "_id.Year": 1, "_id.Month": 1 } }
```

```
])
```

Output

```
{
  _id: {
    Year: 2019,
    Month: 3
  },
  Total_Spend: 11216.050000000001
}
{
  _id: {
    Year: 2019,
    Month: 11
  },
  Total_Spend: 2618.85
}
```

4.2.6. Gender-Based Spending Analysis with Discounted Prices

This query delves into the spending habits of customers based on gender, accounting for discounts applied through coupon usage. It begins by adjusting the average price of products to reflect any discounts if a coupon was used, thereby calculating the 'Effective_Price'. The query then aggregates this effective spending per gender, summing up the total expenditures by multiplying the effective price by the quantity of products purchased. The final step sorts the results by total spending, providing a comparative view of which gender spends more when considering discounted prices.

Code

```
db.shoppingCartCollection.aggregate([
  { $addFields: {
      Effective_Price: {
        $cond: {
          if: { $eq: ["$Coupon_Status", "Used"] },
          then: { $multiply: ["$Avg_Price", { $subtract: [1, {
             $divide: ["$Discount_pct", 100] }] }] },
          else: "$Avg_Price"
```

```
        }
      }
    }
  },
  { $group: {
      _id: "$Gender",
      Total_Spend: { $sum: { $multiply: ["$Effective_Price",
         "$Quantity"] } }
    }
  },
  { $sort: { Total_Spend: -1 } }
])
```

Output

```
{
  _id: 'F',
  Total_Spend: 2708669.751
}
{
  _id: 'M',
  Total_Spend: 1642938.491
}
```

4.3. Queries level hard

4.3.1. Top 5 Products by Location

This query starts by adding a new field, Effective_Price, which adjusts the Avg_Price based on whether a coupon was used, applying the discount percentage if applicable. The data is then grouped by Location and Product_SKU, calculating the total sales (as the product of Effective_Price and Quantity) and the sales count for each product at each location. The results are sorted by location and sales count. Subsequently, the data is regrouped by location alone, with the top 5 selling products (based on sales count) at each location being pushed into a Top_Products list. Finally, the results are projected to show the Location and the Top_Products, and sorted by location, providing a clear view of the best-selling products at each location in the database.

Code

```
db.shoppingCartCollection.aggregate([
  { $addFields: {
      Effective_Price: {
        $cond: {
          if: { $eq: ["$Coupon_Status", "Used"] },
          then: { $multiply: ["$Avg_Price", { $subtract: [1, {
             $divide: ["$Discount_pct", 100] }] }] },
          else: "$Avg_Price"
        }
      }
    }
  },
  { $group: {
      _id: { Location: "$Location", Product_SKU: "$Product_SKU" },
      Total_Sales: { $sum: { $multiply: ["$Effective_Price",
         "$Quantity"] } },
      Sales_Count: { $sum: "$Quantity" }
    }
  },
  { $sort: { "_id.Location": 1, "Sales_Count": -1 } },
  { $group: {
      _id: "$_id.Location",
      Products: { $push: { Product_SKU: "$_id.Product_SKU",
         Total_Sales: "$Total_Sales", Sales_Count: "$Sales_Count"
         } }
    }
  },
  { $project: {
      Location: "$_id",
      Top_Products: { $slice: ["$Products", 5] },
      _id: 0
    }
  },
  { $sort: { "Location": 1 } }
])
```

Output

```
Location: 'California',
  Top_Products: [
    {
      Product_SKU: 'GGOEGGOA017399',
      Total_Sales: 4177.284,
      Sales_Count: 4799
    },
    {
      Product_SKU: 'GGOEGDHC018299',
      Total_Sales: 7489.9580000000005,
      Sales_Count: 2884
    },
    {
      Product_SKU: 'GGOEGBMJ013399',
      Total_Sales: 10780.096,
      Sales_Count: 2450
    },
    {
      Product_SKU: 'GGOEGOLC014299',
      Total_Sales: 9690.973,
      Sales_Count: 2139
    },
    {
      Product_SKU: 'GGOEGFKQ020399',
      Total_Sales: 4113.067,
      Sales_Count: 1791
    }
  ]
}
{
Location: 'Chicago',
  Top_Products: [
    {
      Product_SKU: 'GGOEGDHC018299',
      Total_Sales: 10383.139000000001,
      Sales_Count: 4065
    },
```

```
    {
      Product_SKU: 'GGOEGGOA017399',
      Total_Sales: 3080.7,
      Sales_Count: 4011
    },
    {
      Product_SKU: 'GGOEGFKQ020399',
      Total_Sales: 6244.606,
      Sales_Count: 2746
    },
    {
      Product_SKU: 'GGOEGBMJ013399',
      Total_Sales: 10152.623000000001,
      Sales_Count: 2544
    },
    {
      Product_SKU: 'GGOEGOAQ012899',
      Total_Sales: 4829.612,
      Sales_Count: 2372
    }
  ]
}
continues...
```


4.3.2. Analysis of Average Spending Based on Membership Duration

This query first calculates two new fields for each transaction: Discounted_Price, which applies a discount to the average price if a coupon was used, and Total_Spend, the total expenditure for that transaction. Next, the transactions are grouped by CustomerID, where it calculates the average spend per customer and counts their transactions, while also retrieving their tenure in months. Following this, a second grouping is done based on Tenure_Months to find the average spend per tenure and count the total number of customers for each tenure duration. The resulting data is then projected to highlight the tenure in months, average spend per tenure, and total customers in each tenure group. Finally, the data is sorted by tenure in months.

Code

```
db.shoppingCartCollection.aggregate([
  { $addFields: {
      Discounted_Price: {
        $cond: {
          if: { $eq: ["$Coupon_Status", "Used"] },
          then: { $multiply: ["$Avg_Price", { $subtract: [1, {
             $divide: ["$Discount_pct", 100] }] }] },
          else: "$Avg_Price"
        }
      },
      Total_Spend: { $multiply: ["$Quantity", { $multiply:
         ["$Avg_Price", { $subtract: [1, { $divide:
         ["$Discount_pct", 100] }] }] }] }
    }
  },
  { $group: {
      _id: "$CustomerID",
      Tenure_Months: { $first: "$Tenure_Months" },
      Average_Spend: { $avg: "$Total_Spend" },
      Transaction_Count: { $sum: 1 }
    }
  },
  { $group: {
      _id: "$Tenure_Months",
      Average_Spend_Per_Tenure: { $avg: "$Average_Spend" },
      Total_Customers: { $sum: 1 }
    }
  },
  { $project: {
      Tenure_Months: "$_id",
      _id: 0,
      Average_Spend_Per_Tenure: 1,
      Total_Customers: 1
    }
  },
  { $sort: { "Tenure_Months": 1 } }
])
```

Output


```
{
  Average_Spend_Per_Tenure: 63.015650809180634,
  Total_Customers: 26,
  Tenure_Months: 2
}
{
  Average_Spend_Per_Tenure: 57.230952371921546,
  Total_Customers: 23,
  Tenure_Months: 3
}
{
  Average_Spend_Per_Tenure: 74.83638052481064,
  Total_Customers: 38,
  Tenure_Months: 4
}
{
  Average_Spend_Per_Tenure: 81.02068384191284,
  Total_Customers: 30,
  Tenure_Months: 5
}
continues...
```


4.3.3. Comprehensive Customer Purchase Analysis by Product Category

This aggregation query processes data in the shoppingCartCollection to analyze customer spending habits across different product categories. It begins by grouping the data by both CustomerID and Product_Category, calculating for each group the total spend (product of average price and quantity), average unit price, total quantity of products purchased, and the number of transactions. This initial grouping provides a detailed breakdown of spending per customer for each product category. Afterward, the data is sorted by CustomerID in ascending order and by Total_Spend in descending order within each customer. This sorting allows for a clear view of each customer's spending preference by category. Finally, the data is regrouped by CustomerID, where the details of each product category are consolidated into a Category_Details list. This list includes the product category, total spend, average unit price, total quantity, and transaction count

for each category.

Code

```
db.shoppingCartCollection.aggregate([
  { $group: {
      _id: { CustomerID: "$CustomerID", Product_Category:
         "$Product_Category" },
      Total_Spend: { $sum: { $multiply: ["$Avg_Price",
         "$Quantity"] } },
      Average_Unit_Price: { $avg: "$Avg_Price" },
      Total_Quantity: { $sum: "$Quantity" },
      Transaction_Count: { $sum: 1 }
    }
  },
  { $sort: { "_id.CustomerID": 1, "Total_Spend": -1 } },
  { $group: {
      _id: "$_id.CustomerID",
      Category_Details: {
        $push: {
          Category: "$_id.Product_Category",
          Total_Spend: "$Total_Spend",
          Average_Unit_Price: "$Average_Unit_Price",
          Total_Quantity: "$Total_Quantity",
          Transaction_Count: "$Transaction_Count"
        }
      }
    }
  }
])
```

Output

```
{
  _id: 12626,
  Category_Details: [
    {
      Category: 'Nest-USA',
      Total_Spend: 1755,
```

```
    Average_Unit_Price: 123.54545454545455,
    Total_Quantity: 15,
    Transaction_Count: 11
  },
  {
    Category: 'Drinkware',
    Total_Spend: 1131.35,
    Average_Unit_Price: 7.296666666666667,
    Total_Quantity: 225,
    Transaction_Count: 6
  },
  {
    Category: 'Nest',
    Total_Spend: 1052.65,
    Average_Unit_Price: 190.73,
    Total_Quantity: 6,
    Transaction_Count: 5
  },
  {
    Category: 'Apparel',
    Total_Spend: 438.54,
    Average_Unit_Price: 16.86692307692308,
    Total_Quantity: 26,
    Transaction_Count: 26
  },
  {
    Category: 'Notebooks & Journals',
    Total_Spend: 245.8,
    Average_Unit_Price: 8.995000000000001,
    Total_Quantity: 21,
    Transaction_Count: 2
  },
  {
    Category: 'Lifestyle',
    Total_Spend: 175.8,
    Average_Unit_Price: 1.635,
    Total_Quantity: 110,
    Transaction_Count: 2
  },
```

```
    {
      Category: 'Office',
      Total_Spend: 144.34,
      Average_Unit_Price: 1.9785714285714289,
      Total_Quantity: 162,
      Transaction_Count: 7
    },
    {
      Category: 'Bags',
      Total_Spend: 129.86,
      Average_Unit_Price: 18.79,
      Total_Quantity: 14,
      Transaction_Count: 5
    },
    {
      Category: 'Headgear',
      Total_Spend: 23.92,
      Average_Unit_Price: 11.96,
      Total_Quantity: 2,
      Transaction_Count: 2
    },
    {
      Category: 'Bottles',
      Total_Spend: 2.99,
      Average_Unit_Price: 2.99,
      Total_Quantity: 1,
      Transaction_Count: 1
    }
  ]
}
{
  _id: 14457,
  Category_Details: [
    {
      Category: 'Office',
      Total_Spend: 978.1199999999999,
      Average_Unit_Price: 12.87,
      Total_Quantity: 76,
      Transaction_Count: 1
```

```
    },
    {
      Category: 'Apparel',
      Total_Spend: 537.9000000000001,
      Average_Unit_Price: 17.93,
      Total_Quantity: 30,
      Transaction_Count: 3
    }
  ]
}
continues...
```

4.3.4. Customer Expenditure and Coupon Usage Analysis

This aggregation query in the shoppingCartCollection is designed to analyze customer spending and savings based on coupon usage. It starts by calculating Spend_With_Discount for each transaction, adjusting the spend based on whether a coupon was used. Then, it groups transactions by both CustomerID and Coupon_Status, summing up total spend, actual savings (for used coupons), potential savings (for coupons clicked but not used or not used at all), and the count of transactions. The query then further groups the data by CustomerID alone, creating a detailed view for each customer that includes the breakdown of transactions by coupon status. This breakdown includes total spend, actual savings achieved through used coupons, potential savings that could have been achieved, and the number of transactions per coupon status. This aggregation is particularly valuable for understanding the impact of coupon usage on customer spending behavior and the effectiveness of coupon campaigns.

Code

```
db.shoppingCartCollection.aggregate([
  { $addFields: {
      Spend_With_Discount: {
        $cond: {
          if: { $eq: ["$Coupon_Status", "Used"] },
          then: { $multiply: ["$Avg_Price", "$Quantity", {
             $subtract: [1, { $divide: ["$Discount_pct", 100] }]
             }] },
          else: { $multiply: ["$Avg_Price", "$Quantity"] }
```

```
        }
      }
    }
  },
  { $group: {
      _id: { CustomerID: "$CustomerID", Coupon_Status:
         "$Coupon_Status" },
      Total_Spend: { $sum: "$Spend_With_Discount" },
      Actual_Savings: {
        $sum: {
          $cond: {
            if: { $eq: ["$Coupon_Status", "Used"] },
            then: { $multiply: ["$Avg_Price", "$Quantity", {
               $divide: ["$Discount_pct", 100] }] },
            else: 0
          }
        }
      },
      Potential_Savings: {
        $sum: {
          $cond: {
            if: { $in: ["$Coupon_Status", ["Not Used",
               "Clicked"]] },
            then: { $multiply: ["$Avg_Price", "$Quantity", {
               $divide: ["$Discount_pct", 100] }] },
            else: 0
          }
        }
      },
      Transaction_Count: { $sum: 1 }
    }
  },
  { $group: {
      _id: "$_id.CustomerID",
      Coupon_Details: {
        $push: {
          Status: "$_id.Coupon_Status",
          Total_Spend: "$Total_Spend",
          Actual_Savings: "$Actual_Savings",
```

```
          Potential_Savings: "$Potential_Savings",
          Transaction_Count: "$Transaction_Count"
        }
      }
    }
  }
])
```

Output

```
{
  _id: 15750,
  Coupon_Details: [
    {
      Status: 'Not Used',
      Total_Spend: 487.58,
      Actual_Savings: 0,
      Potential_Savings: 146.274,
      Transaction_Count: 3
    },
    {
      Status: 'Used',
      Total_Spend: 837.788,
      Actual_Savings: 359.052,
      Potential_Savings: 0,
      Transaction_Count: 9
    },
    {
      Status: 'Clicked',
      Total_Spend: 440.32,
      Actual_Savings: 0,
      Potential_Savings: 119.29799999999999,
      Transaction_Count: 9
    }
  ]
}
{
  _id: 15838,
  Coupon_Details: [
```

```
    {
      Status: 'Used',
      Total_Spend: 27.965,
      Actual_Savings: 11.985000000000001,
      Potential_Savings: 0,
      Transaction_Count: 1
    },
    {
      Status: 'Clicked',
      Total_Spend: 317.49,
      Actual_Savings: 0,
      Potential_Savings: 95.247,
      Transaction_Count: 4
    },
    {
      Status: 'Not Used',
      Total_Spend: 122.77,
      Actual_Savings: 0,
      Potential_Savings: 12.277000000000001,
      Transaction_Count: 1
    }
  ]
}
continues...
```

4.3.5. Annual Sales and Transaction Analysis by Month and Day

This aggregation query performs a detailed analysis of sales data by date in the shoppingCartCollection. It starts by extracting the day, month, and year from the Transaction_Date field. The data is then grouped by these extracted date components (Year, Month, Day), calculating total sales (multiplying average price and quantity) and the count of transactions for each day. Subsequently, the data is re-grouped at the month level, collecting daily sales data within each month. This step is followed by a similar regrouping at the year level, organizing monthly data under each year. The final structure of the data represents a hierarchical breakdown: each year contains monthly data, and each month contains daily data, with total sales and transaction counts for each period. The query includes sorting steps at each level of grouping (day, month, year) to ensure

the data is presented in chronological order.

Code

```
db.shoppingCartCollection.aggregate([
  { $addFields: {
      Day: { $dayOfMonth: "$Transaction_Date" },
      Month: "$Month",
      Year: { $year: "$Transaction_Date" }
    }
  },
  { $group: {
      _id: { Year: "$Year", Month: "$Month", Day: "$Day" },
      Total_Sales: { $sum: { $multiply: ["$Avg_Price",
         "$Quantity"] } },
      Transaction_Count: { $sum: 1 }
    }
  },
  { $sort: { "_id.Year": 1, "_id.Month": 1, "_id.Day": 1 } },
  { $group: {
      _id: { Year: "$_id.Year", Month: "$_id.Month" },
      Daily_Data: { $push: { Day: "$_id.Day", Total_Sales:
         "$Total_Sales", Transaction_Count: "$Transaction_Count"
         } }
    }
  },
  { $sort: { "_id.Year": 1, "_id.Month": 1 } },
  { $group: {
      _id: "$_id.Year",
      Monthly_Data: { $push: { Month: "$_id.Month", Daily_Data:
         "$Daily_Data" } }
    }
  },
  { $project: { Year: "$_id", _id: 0, Monthly_Data: 1 } },
  { $sort: { "Year": 1 } }
])
```

Output

```
{
  Monthly_Data: [
    {
      Month: 1,
      Daily_Data: [
        {
          Day: 1,
          Total_Sales: 7380.49,
          Transaction_Count: 89
        },
        {
          Day: 2,
          Total_Sales: 13438.470000000001,
          Transaction_Count: 115
        },
        {
          Day: 3,
          Total_Sales: 23834.21,
          Transaction_Count: 207
        },
        {
          Day: 4,
          Total_Sales: 16603.41,
          Transaction_Count: 169
        },
        {
          Day: 5,
          Total_Sales: 17373.19,
          Transaction_Count: 189
        },
        {
          Day: 6,
          Total_Sales: 14350.73,
          Transaction_Count: 159
        },
        {
          Day: 7,
          Total_Sales: 5905.04,
          Transaction_Count: 53
```

```
  },
  {
    Day: 8,
    Total_Sales: 7585.6,
    Transaction_Count: 71
  },
  {
    Day: 9,
    Total_Sales: 18482.71,
    Transaction_Count: 188
  },
  {
    Day: 10,
    Total_Sales: 14724.460000000001,
    Transaction_Count: 118
  },
  {
    Day: 11,
    Total_Sales: 11747.94,
    Transaction_Count: 129
  },
  {
    Day: 12,
    Total_Sales: 10492.28,
    Transaction_Count: 120
  },
  {
    Day: 13,
    Total_Sales: 14939.64,
    Transaction_Count: 150
  },
  {
    Day: 14,
    Total_Sales: 4484.37,
    Transaction_Count: 57
  },
  {
    Day: 15,
    Total_Sales: 8485.78,
```

```
    Transaction_Count: 55
  },
  {
    Day: 16,
    Total_Sales: 11555.81,
    Transaction_Count: 117
  },
  {
    Day: 17,
    Total_Sales: 17860.91,
    Transaction_Count: 185
  },
  {
    Day: 18,
    Total_Sales: 20372.93,
    Transaction_Count: 187
  },
  {
    Day: 19,
    Total_Sales: 12778.72,
    Transaction_Count: 119
  },
  {
    Day: 20,
    Total_Sales: 13621.84,
    Transaction_Count: 122
  },
  {
    Day: 21,
    Total_Sales: 5537.52,
    Transaction_Count: 56
  },
  {
    Day: 22,
    Total_Sales: 11832.49,
    Transaction_Count: 118
  },
  {
    Day: 23,
```

```
    Total_Sales: 15113.9,
    Transaction_Count: 147
  },
  {
    Day: 24,
    Total_Sales: 14025.51,
    Transaction_Count: 140
  },
  {
    Day: 25,
    Total_Sales: 13320.68,
    Transaction_Count: 157
  },
  {
    Day: 26,
    Total_Sales: 13648.91,
    Transaction_Count: 157
  },
  {
    Day: 27,
    Total_Sales: 13526.25,
    Transaction_Count: 168
  },
  {
    Day: 28,
    Total_Sales: 6275.81,
    Transaction_Count: 39
  },
  {
    Day: 29,
    Total_Sales: 7568.16,
    Transaction_Count: 99
  },
  {
    Day: 30,
    Total_Sales: 18683.32,
    Transaction_Count: 189
  },
  {
```

```
          Day: 31,
          Total_Sales: 18073.5,
          Transaction_Count: 194
        }
      ]
    },

    continues....

  ],
  Year: 2019
}
```

4.3.6. Gender-Based Online and Offline Spending Analysis by Location

This query performs a comprehensive analysis of spending habits based on gender and location. It aggregates total online and offline expenditures for customers, distinguishing between male and female shoppers in various locations. Initially, the query groups data by gender and location, calculating the total online spend, total offline spend, and the combined overall spend for each group. It then sorts these groups by location and overall spend, ensuring that the data is organized starting with the highest spending group per location. Finally, the query re-groups this sorted data by gender, creating a detailed summary of spending patterns. This summary includes a breakdown of online and offline spending in each location for both male and female customers, providing valuable insights into consumer behavior across different channels and regions. The sorted results offer a clear understanding of which gender spends more in which location and how their spending is distributed between online and offline channels.

Code

```
db.shoppingCartCollection.aggregate([
  { $group: {
      _id: { Gender: "$Gender", Location: "$Location" },
      Total_Online_Spend: { $sum: "$Online_Spend" },
      Total_Offline_Spend: { $sum: "$Offline_Spend" },
      Total_Spend: { $sum: { $add: ["$Online_Spend",
        "$Offline_Spend"] } }
```

```
    }
  },
  { $sort: { "_id.Location": 1, "Total_Spend": -1 } },
  { $group: {
      _id: "$_id.Gender",
      Location_Spending: { $push: { Location: "$_id.Location",
         Online: "$Total_Online_Spend", Offline:
         "$Total_Offline_Spend", Total: "$Total_Spend" } }
    }
  },
  { $sort: { "_id": 1 } }
])
```

Output

```
{
  _id: 'F',
  Location_Spending: [
    {
      Location: 'California',
      Online: 19294537.42,
      Offline: 29199900,
      Total: 48494437.42
    },
    {
      Location: 'Chicago',
      Online: 21429256.2,
      Offline: 33303300,
      Total: 54732556.2
    },
    {
      Location: 'New Jersey',
      Online: 6122225.97,
      Offline: 8822300,
      Total: 14944525.97
    },
    {
      Location: 'New York',
      Online: 13580982.16,
```

```
      Offline: 19548700,
      Total: 33129682.16
    },
    {
      Location: 'Washington DC',
      Online: 2279914.72,
      Offline: 3600400,
      Total: 5880314.72
    }
  ]
}
continues...
```

4.3.7. Top 100 High-Spending Customers and Their Spending by Product Category

This query identifies the top 100 customers with the highest spending in the shopping cart database. It calculates each customer's total expenditure and fetches their detailed transaction data. The data is then analyzed to determine spending across different product categories for these high-spenders. The final output provides a breakdown of each top spender's total expenditure and their preferred product categories, offering insights into customer purchasing patterns and preferences.

Code

```
db.shoppingCartCollection.aggregate([
  { $group: {
      _id: "$CustomerID",
      Total_Spend: { $sum: { $multiply: ["$Avg_Price",
         "$Quantity"] } }
    }
  },
  { $sort: { "Total_Spend": -1 } },
  { $limit: 100 },
  { $lookup: {
      from: "shoppingCartCollection",
      localField: "_id",
      foreignField: "CustomerID",
```

```
      as: "Customer_Transactions"
    }
  },
  { $unwind: "$Customer_Transactions" },
  { $group: {
      _id: { CustomerID: "$_id", Product_Category:
         "$Customer_Transactions.Product_Category" },
      Total_Spend: { $first: "$Total_Spend" },
      Category_Spend: { $sum: { $multiply:
         ["$Customer_Transactions.Avg_Price",
         "$Customer_Transactions.Quantity"] } }
    }
  },
  { $sort: { "Total_Spend": -1, "Category_Spend": -1 } },
  { $group: {
      _id: "$_id.CustomerID",
      Total_Spend: { $first: "$Total_Spend" },
      Preferred_Categories: { $push: { Category:
         "$_id.Product_Category", Spend: "$Category_Spend" } }
    }
  }
])
```

Output

```
{
  _id: 13093,
  Total_Spend: 11879.72,
  Preferred_Categories: [
    {
      Category: 'Nest-USA',
      Spend: 5591.41
    },
    {
      Category: 'Nest',
      Spend: 2889.75
    },
    {
      Category: 'Drinkware',
```

```
    Spend: 758.4
  },
  {
    Category: 'Apparel',
    Spend: 698.4
  },
  {
    Category: 'Office',
    Spend: 559.39
  },
  {
    Category: 'Nest-Canada',
    Spend: 511
  },
  {
    Category: 'Lifestyle',
    Spend: 402.09
  },
  {
    Category: 'Bags',
    Spend: 197.92000000000002
  },
  {
    Category: 'Gift Cards',
    Spend: 100
  },
  {
    Category: 'Housewares',
    Spend: 74.98
  },
  {
    Category: 'Headgear',
    Spend: 48.1
  },
  {
    Category: 'Notebooks & Journals',
    Spend: 19.5
  },
  {
```

```
      Category: 'Waze',
      Spend: 15.19
    },
    {
      Category: 'Accessories',
      Spend: 13.59
    }
  ]
}
continues...
```

4.3.8. Monthly Best-Selling Products Analysis with Discount Consideration

query is crafted to analyze the best-selling products across different categories on a monthly basis, taking into account any discounts applied. It begins by adding fields to calculate the effective sales value, adjusting for discounts if a coupon was used. The query then groups the data by month, year, product category, and SKU, aggregating the total sales to reflect the discounted prices. For each category within every month, the top-selling product is identified based on these adjusted sales figures. The results are further grouped and sorted to present a clear, chronological view of the best-selling products in each category for every month.

Code

```
db.shoppingCartCollection.aggregate([
  { $addFields: {
      Year: { $year: "$Transaction_Date" },
      Effective_Sales: {
        $cond: {
          if: { $eq: ["$Coupon_Status", "Used"] },
          then: { $multiply: ["$Avg_Price", "$Quantity", {
             $subtract: [1, { $divide: ["$Discount_pct", 100] }]
             }] },
          else: { $multiply: ["$Avg_Price", "$Quantity"] }
        }
      }
    }
```

```
  },
  { $group: {
      _id: { Month: "$Month", Year: "$Year", Product_Category:
         "$Product_Category", Product_SKU: "$Product_SKU" },
      Total_Sales: { $sum: "$Effective_Sales" }
    }
  },
  { $sort: { "Total_Sales": -1 } },
  { $group: {
      _id: { Month: "$_id.Month", Year: "$_id.Year",
         Product_Category: "$_id.Product_Category" },
      Top_Product: { $first: "$_id.Product_SKU" },
      Top_Product_Sales: { $first: "$Total_Sales" }
    }
  },
  { $sort: { "_id.Year": 1, "_id.Month": 1, "Top_Product_Sales":
     -1 } },
  { $group: {
      _id: { Month: "$_id.Month", Year: "$_id.Year" },
      Category_Best_Sellers: { $push: { Category:
         "$_id.Product_Category", Top_Product: "$Top_Product",
         Top_Product_Sales: "$Top_Product_Sales" } }
    }
  },
  { $project: {
      Month: "$_id.Month",
      Year: "$_id.Year",
      _id: 0,
      Category_Best_Sellers: 1
    }
  },
  { $sort: { "Year": 1, "Month": 1 } }
])
```

Output

```
{
  Category_Best_Sellers: [
    {
```

```
    Category: 'Nest-USA',
    Top_Product: 'GGOENEBJ079499',
    Top_Product_Sales: 111992.65000000001
  },
  {
    Category: 'Nest-Canada',
    Top_Product: 'GGOENEBJ081899',
    Top_Product_Sales: 3633.8100000000004
  },
  {
    Category: 'Office',
    Top_Product: 'GGOEGOCB017499',
    Top_Product_Sales: 3102.6059999999998
  },
  {
    Category: 'Bags',
    Top_Product: 'GGOEGBRJ037399',
    Top_Product_Sales: 2729.1600000000003
  },
  {
    Category: 'Drinkware',
    Top_Product: 'GGOEGDHQ015399',
    Top_Product_Sales: 1877.594
  },
  {
    Category: 'Notebooks & Journals',
    Top_Product: 'GGOEYOCR078099',
    Top_Product_Sales: 1264.026
  },
  {
    Category: 'Lifestyle',
    Top_Product: 'GGOEAKDH019899',
    Top_Product_Sales: 1242.581
  },
  {
    Category: 'Apparel',
    Top_Product: 'GGOEGAFB035814',
    Top_Product_Sales: 1068.597
  },
```

```
  {
    Category: 'Housewares',
    Top_Product: 'GGOEGCBQ016499',
    Top_Product_Sales: 928.19
  },
  {
    Category: 'More Bags',
    Top_Product: 'GGOEGBMB073599',
    Top_Product_Sales: 744.03
  },
  {
    Category: 'Headgear',
    Top_Product: 'GGOEYHPA003610',
    Top_Product_Sales: 469.15600000000006
  },
  {
    Category: 'Google',
    Top_Product: 'GGOEGDHG082499',
    Top_Product_Sales: 372.19
  },
  {
    Category: 'Bottles',
    Top_Product: 'GGOEADHH055999',
    Top_Product_Sales: 301.16400000000004
  },
  {
    Category: 'Backpacks',
    Top_Product: 'GGOEGBRD079699',
    Top_Product_Sales: 268.19
  },
  {
    Category: 'Waze',
    Top_Product: 'GGOEWEBB082699',
    Top_Product_Sales: 174.064
  },
  {
    Category: 'Gift Cards',
    Top_Product: 'GGOEGGCX056499',
    Top_Product_Sales: 144.42399999999998
```

```
    },
    {
      Category: 'Fun',
      Top_Product: 'GGOEGOBC078699',
      Top_Product_Sales: 87.14999999999999
    },
    {
      Category: 'Accessories',
      Top_Product: 'GGOEGBPB081999',
      Top_Product_Sales: 51.57
    },
    {
      Category: 'Android',
      Top_Product: 'GGOEAAAH083314',
      Top_Product_Sales: 37.116
    }
  ],
  Month: 1,
  Year: 2019
}
continues...
```

5 | Extra

Through the subsequent sections, the chapter explores various dimensions of the dataset: from the interplay of gender in purchasing decisions to the influence of customer tenure on spending habits. It scrutinizes how these factors manifest over time, providing a clearer picture of the customer journey and the lifecycle of market trends.

This comprehensive analysis serves as an indispensable tool for businesses looking to understand and leverage consumer behavior. It opens doors to targeted marketing, predictive analytics, and strategic planning, all aimed at capturing the essence of what drives the modern consumer. The ensuing subsections will meticulously dissect each facet of this dataset to expose the underlying economic patterns and consumer trends.

Culminating the chapter is the final section, which introduces a cutting-edge interface developed using the Flask web framework. This dynamic interface acts as a portal through which the analyses articulated in the preceding sections can be dynamically viewed and interacted with. It represents the synthesis of data analysis with web technology, enabling a real-time, interactive exploration of the insights garnered from the dataset. This interface is designed not just as a showcase of data visualization, but as an interactive analytical tool that can dynamically display various plots pertinent to the discussions held within the chapter.

5.1. Analysis of Monthly Sales by Gender

The bar chart presented in Figure 5.1 illustrates the monthly sales distribution split by gender, where sales are calculated as the product of `Avg_Price` and `Quantity` for each transaction.

From the chart, it is evident that there are distinct trends in purchasing behaviors between male and female customers across different months. Key observations include:

- **Consistent Gender Distribution:** Unlike showing a dominant trend for female gender, the sales for both male and female customers are relatively consistent across different months. This consistency suggests a balanced customer base in terms of

gender, without significant gender-specific purchasing patterns.

- **Monthly Sales Trends:** The total sales for both genders show similar trends across the months. This pattern indicates that the purchasing behavior of both male and female customers is influenced similarly by monthly factors, which could include seasonal changes, holidays, or marketing campaigns.
- **Stability in Consumer Behavior:** The lack of dramatic peaks or troughs in the sales data points to a stable consumer behavior throughout the year. This stability could be indicative of a regular, loyal customer base with predictable purchasing patterns.

Hence, the data suggests a balanced and stable market in terms of gender distribution.

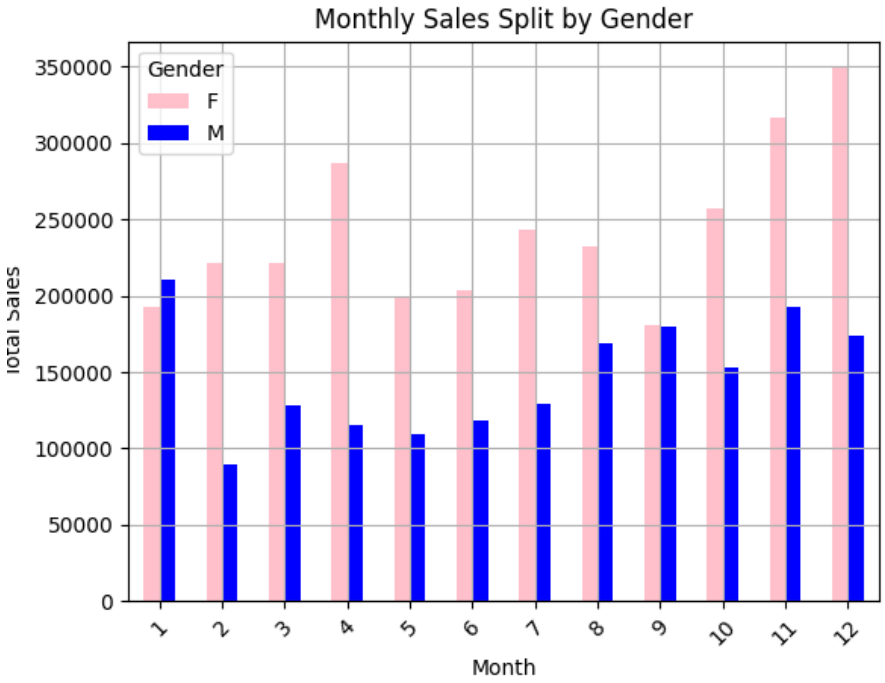

Figure 5.1: Monthly Sales Distribution by Gender

5.2. Monthly Sales Trends

The histogram in Figure 5.2 represents the total sales per month. The sales are computed in the same way of the previous cases. This histogram provides a visual representation of sales activity throughout the year and indicates key periods where sales volumes increase.

Observations from the histogram are summarized as follows:

- **Monthly Sales Progression:** There is a clear progression in total sales from the start to the end of the year. Initial months show a steady amount of sales, with a gradual increase as the year progresses.
- **End-of-Year Surge:** Notably, there is a significant surge in sales towards the end of the year, particularly in the last two months. This increase might be attributable to holiday shopping seasons, such as Black Friday, Cyber Monday, and Christmas.
- **Mid-Year Sales Activity:** Mid-year months exhibit a moderate level of sales activity, which could correspond to summer sales or other mid-year promotions.

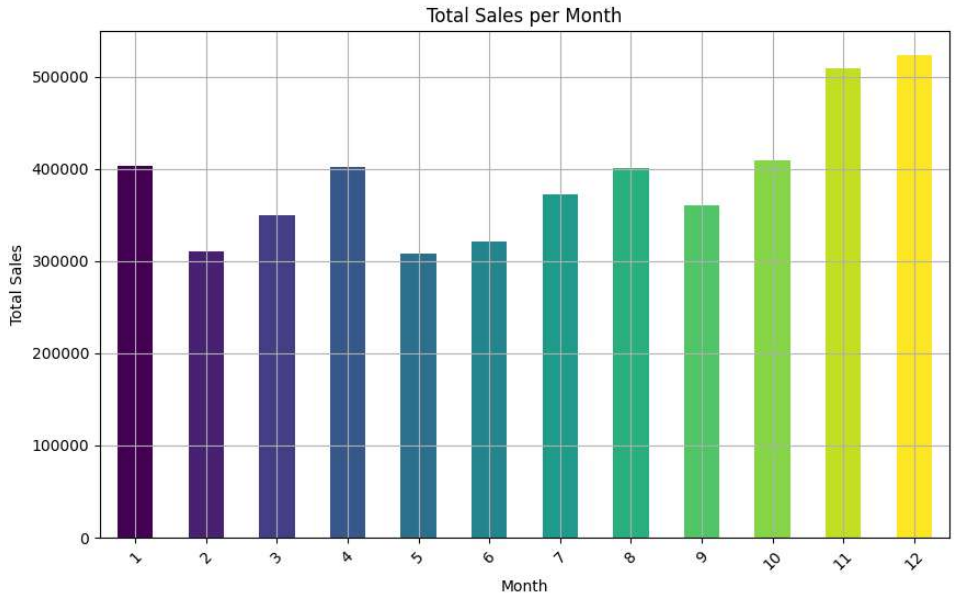

Figure 5.2: Total Sales per Month

5.3. Product Category Sales Trends Over the Year

The line chart depicted in Figure 5.3 showcases the monthly sales for various product categories. The sales figures are computed by multiplying the average price (`Avg_Price`) by the quantity sold (`Quantity`) for each product category on a monthly basis.

The following insights can be drawn from the line chart:

- **Category-Specific Dynamics:** Each product category demonstrates distinct sales trajectories over the months. Some categories show consistent sales throughout the year, while others exhibit significant variability, reflecting consumer demand and possibly promotional activities.
- **Seasonal Peaks:** Certain categories experience sharp increases in sales during

specific months. For instance, the `Bags` category shows a notable spike towards the end of the year, which could be associated with holiday season promotions and gift purchasing.

- **Sales Distribution:** A few categories dominate the sales figures, suggesting they are either more popular or hold higher average prices and quantities sold. Conversely, several categories maintain low sales across all months, indicating niche markets or less demand.

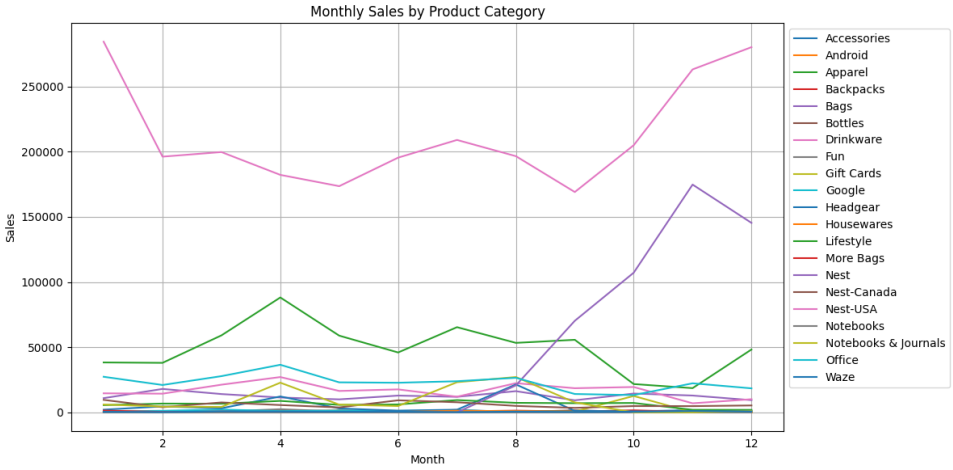

Figure 5.3: Monthly Sales by Product Category

5.4. Customer Tenure and Spending Behavior

5.4.1. Customer Segmentation Over Time

Figure 5.4 presents the distribution of customers across different tenure categories. The data provides insights into the demographics of the customer base in relation to their shopping tenure.

Key takeaways include:

- **Largest Segment:** The '2+ years' category forms the largest segment, indicating a strong base of long-term customers.
- **Declining Numbers with Increased Tenure:** There is a noticeable decrease in the number of customers as tenure increases from '1-2 years' to '0-6 months', which may signal a drop-off in customer retention over time.
- **Potential for Growth:** The smaller segments of '0-6 months' and '7-12 months' offer potential areas for growth through improved customer engagement and retention

initiatives.

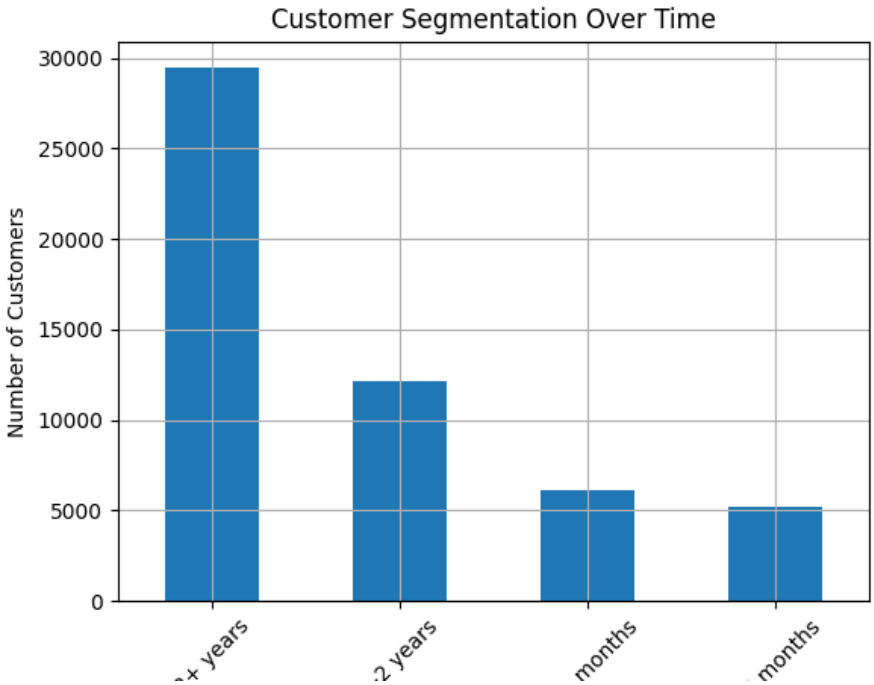

Figure 5.4: Customer Segmentation Over Time.

5.4.2. Average Spending Trends by Customer Tenure

The bar chart in Figure 5.5 illustrates the average spending trends segregated by customer tenure categories. This analysis is pivotal in understanding how the duration of the customer's relationship with the platform influences their spending habits.

Observations from the chart reveal that:

- **Online vs. Offline Spending:** Customers tend to spend more in offline channels compared to online across all tenure categories. This could be indicative of customer preferences or the nature of products purchased.
- **Spending Increases with Tenure:** There is a general trend of increasing expenditure with the length of customer tenure, suggesting that long-term customers are more valuable in terms of cumulative spending.
- **Short-Term vs. Long-Term Behavior:** Customers within the '0-6 months' category exhibit the lowest average spending, which significantly increases as they cross into the '7-12 months' bracket and beyond.

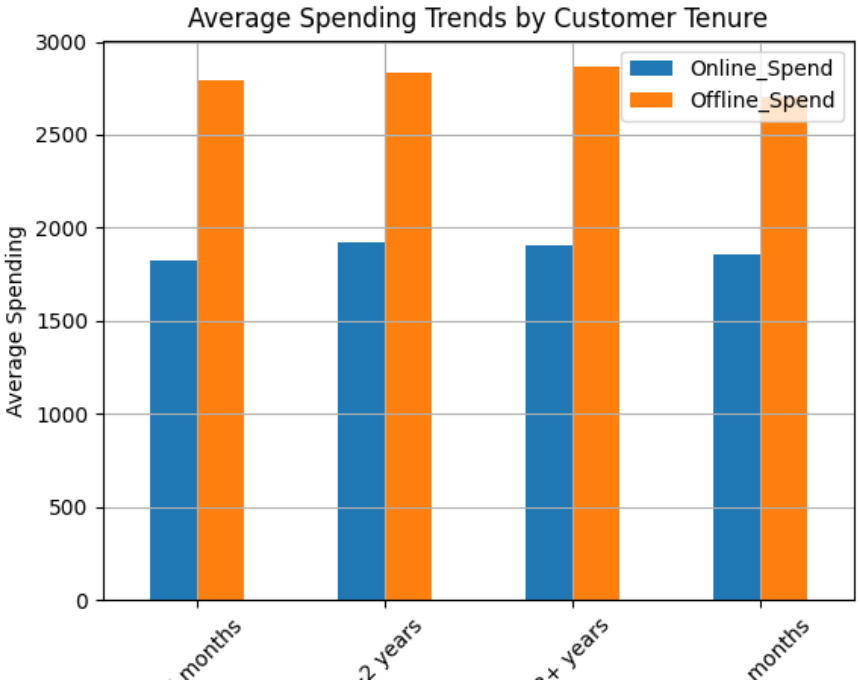

Figure 5.5: Average Spending Trends by Customer Tenure

5.5. Spending Analysis by Gender

The boxplot in Figure 5.6 depicts the distribution of total spending by gender. The `Effective Price` represents the actual price paid by customers, accounting for any discounts received.

From the boxplot, several conclusions can be drawn:

- **Central Tendency:** The median spending appears to be slightly higher for male gender, indicating a central tendency for this gender to spend more per transaction.
- **Spread and Variability:** Both genders display a wide interquartile range, showing substantial variability in spending habits among customers.
- **Outliers:** There are numerous outliers for both genders, suggesting that there are customers who spend significantly more than the average.
- **Maximum Spending:** The range of spending, excluding outliers, also varies between genders, with one showing a higher maximum spending.

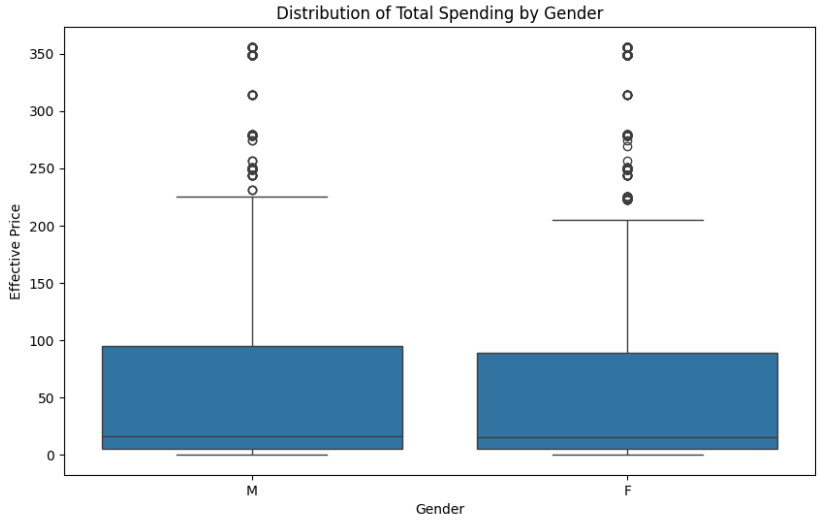

Figure 5.6: Distribution of Total Spending by Gender

5.6. Exploration of Monthly Spending Patterns

The line graph in Figure 5.7 demonstrates the total monthly spending trend. The trend is obtained by summing the `Effective Price` for each month, which takes into account the average price adjusted by any applied discounts.

The plot reveals several notable features of spending behavior:

- **Overall Trend:** There is a clear upward trend in total spending as the months progress, which may be indicative of growing customer acquisition, retention, or increased average spending per customer.
- **Seasonal Fluctuations:** The plot exhibits fluctuations within the year, with spending dips and spikes that may correspond to seasonal sales cycles, promotional events, or consumer buying patterns.
- **End-of-Year Increase:** A significant increase in spending is observed towards the end of the year, likely due to holiday shopping or end-of-year discounts.

5.7. Interface

In the final section of our analysis, we present an innovative interface crafted using the Flask web framework, which serves as a dynamic platform to showcase the various plots and insights derived from the aforementioned sections, enabling users to interact with the

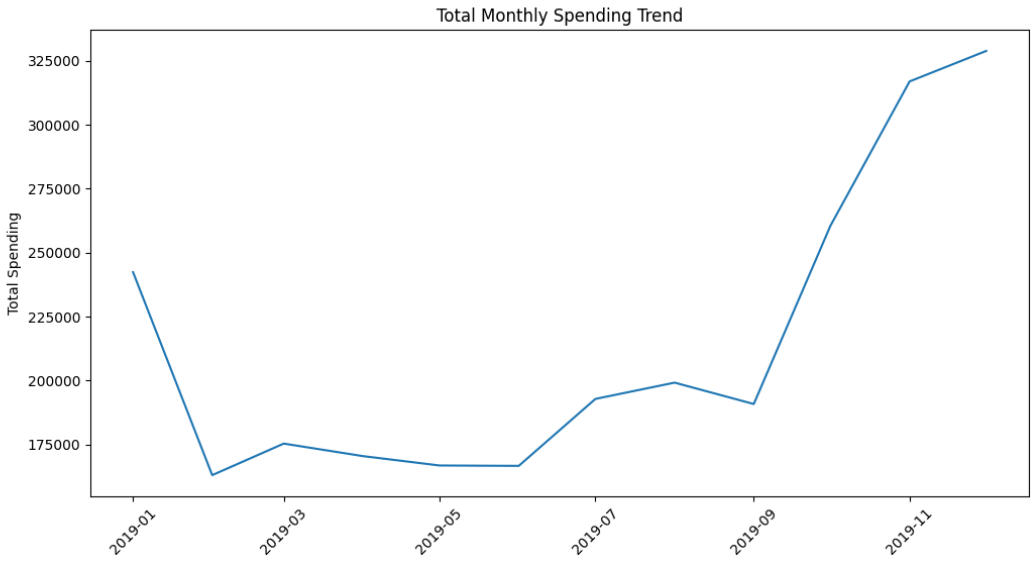

Figure 5.7: Total Monthly Spending Trend

dataset's findings in real time.

The Flask application, which is executed within a Python environment, is designed to be user-centric, providing an intuitive and interactive experience. Upon accessing the interface, users are greeted with a selection of plots, each corresponding to a significant analysis discussed in previous sections. These plots encompass a range of insights, from *Monthly Sales Split by Gender* to *Total Monthly Spending Trend*, offering a multifaceted view of consumer behavior.

The logic underlying the interface is straightforward yet effective. Upon the user's selection of a particular plot, the Flask app invokes a subprocess that executes the associated Python script. This script generates the selected plot and stores it as an image file. Flask then serves this image file, rendering it within the user's browser. This method ensures that the most current data is used each time, reflecting the latest trends and information available from the dataset.

To facilitate a comprehensive understanding of the interface's capabilities, a representative video will be included in the delivery folder. This video will guide viewers through the interface, demonstrating its functionalities and the ease with which the different analyses can be accessed and visualized.

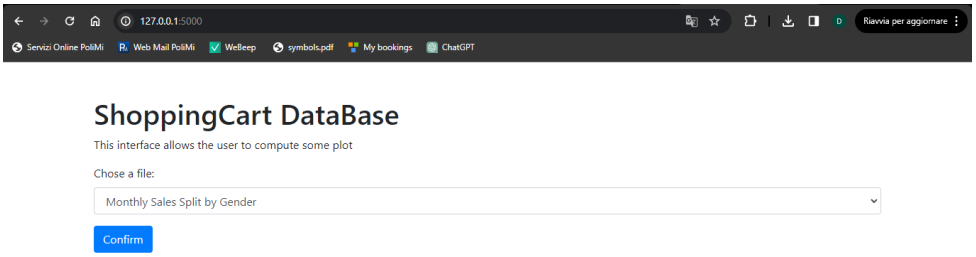

Figure 5.8: Interface Screen

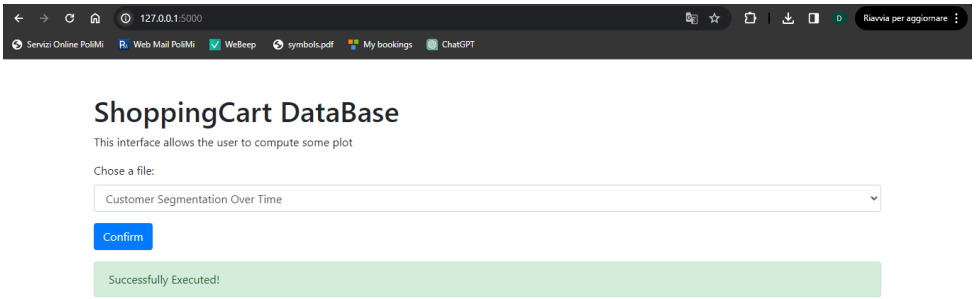

Figure 5.9: Plot loaded successfully

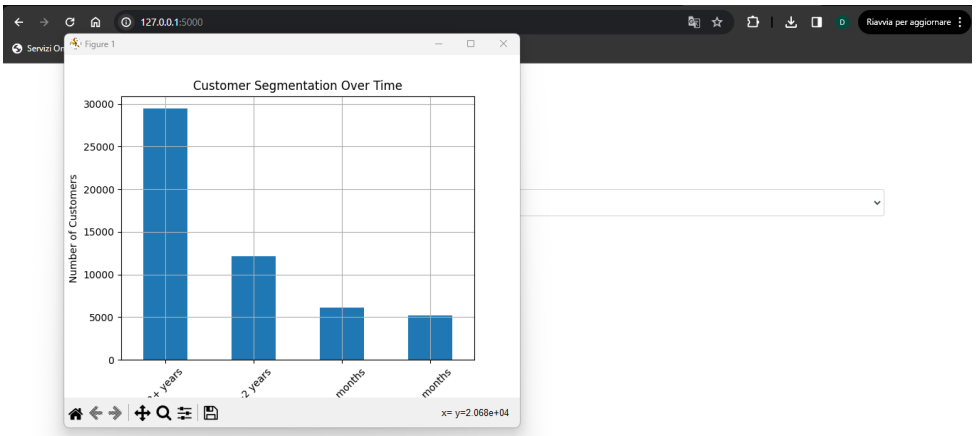

Figure 5.10: Plot shown

List of Figures