Elmain Boston

Use Case Study Report

Group 5
Ragul Narayanan Magesh
Barath Keshav Sriram Kumaran

617-792-7068
857-421-1410

magesh.ra@northeastern.edu
sriramkumaran.b@northeastern.edu

Percentage of Effort Contributed by Student 1: ____50%____

Percentage of Effort Contributed by Student 2: ____50%____

Signature of Student 1: _____

Signature of Student 2: _____

Submission Date: 12/08/2024

USE CASE STUDY REPORT

Group no.: Group 5

Student names: Barath Keshav Sriram Kumaran and Ragul Narayanan Magesh

Executive Summary:

This study focuses on the design and implementation of a relational database tailored for the online retail industry, addressing critical operational challenges faced by e-commerce platforms. Many online retailers need help with the repetitive and time-consuming task of managing customer data, order histories, and inventory, leading to inefficiencies and data duplication. This project aims to streamline these processes by implementing a robust relational database system, reducing data entry time by 50%, and delivering significant cost savings across the retail sector.

In addition to enhancing data management, the database incorporates an advanced analytics platform capable of providing valuable insights into consumer behavior, popular product categories, and sales trends. This platform enables retailers to optimize inventory, identify emerging market opportunities, and improve overall business performance. Employee management, an often-overlooked aspect of online retail operations, is also integrated into the database design, ensuring seamless tracking of employee activities and performance.

The database was developed by analyzing the essential data requirements for online retail operations, incorporating inputs from industry leaders to ensure practical applicability. Entity-relationship (ER) and Unified Modeling Language (UML) diagrams were created to define the database structure, followed by mapping the conceptual model to a relational model with primary and foreign keys. The database was fully implemented in MySQL, and a prototype was tested in a NoSQL environment using MongoDB to explore its feasibility for graph-based queries. The implemented database has proven to be highly effective. By integrating it with tools like Python, the analytics capabilities provide actionable insights, such as tracking customer purchasing patterns and identifying supply chain bottlenecks.

Future improvements will include implementing comprehensive Data Governance measures, enhancing data security, and enabling scalable operations. With these advancements, this database project is poised to be a game-changer for the online retail industry, offering a foundation for improved efficiency, customer satisfaction, and sustainable growth.

1. Introduction

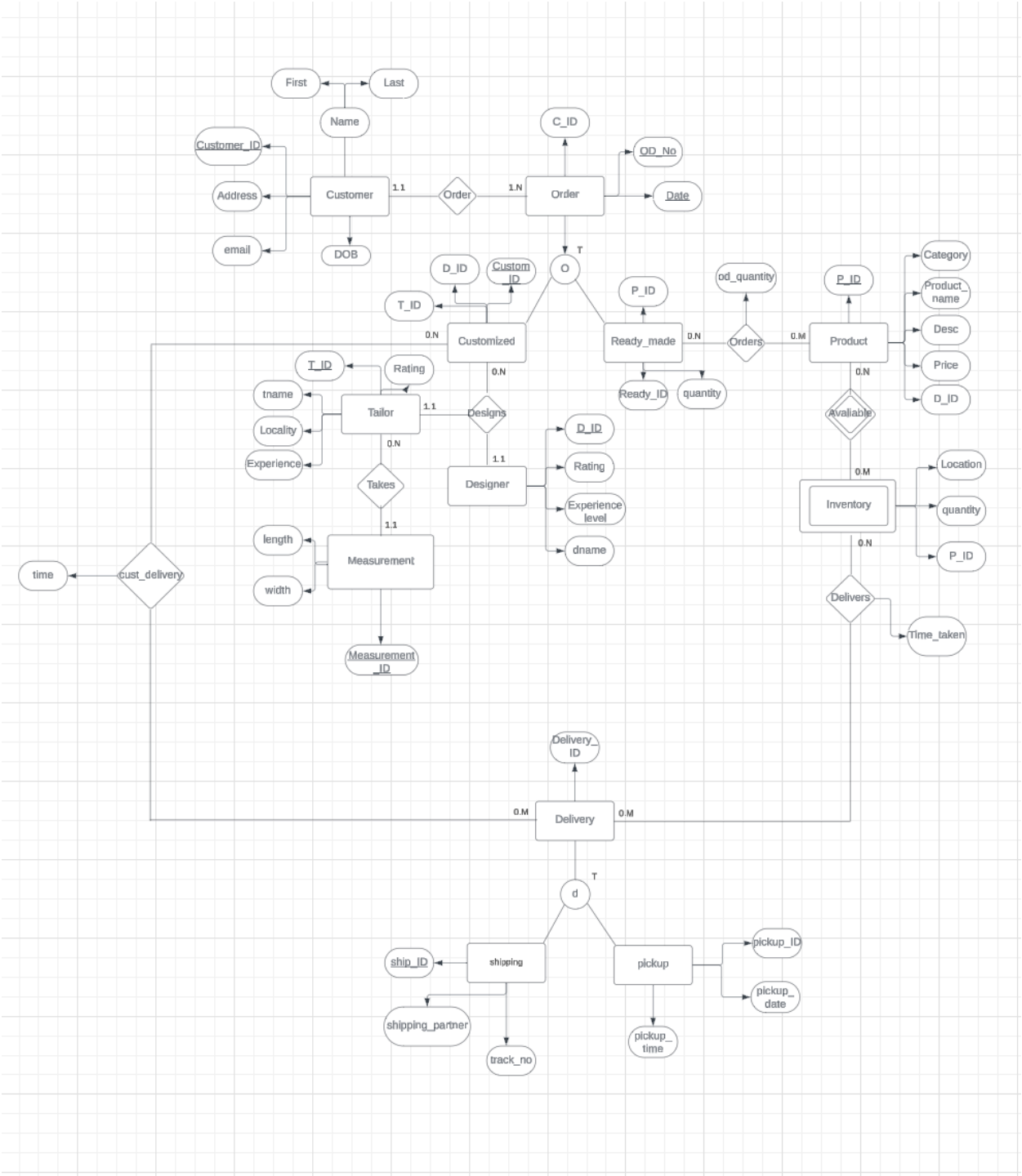
Elmain Boston, a 75-year-old luxury designer clothing brand, is undergoing a strategic transformation to reclaim its market position and strengthen customer engagement in a rapidly evolving retail landscape. The company, traditionally reliant on its six brick-and-mortar stores in major U.S. cities, has faced declining sales and diminished patronage, driven by the rise of e-commerce and a shift in consumer preferences toward personalized shopping experiences. In response, Elmain Boston has decided to launch an e-commerce platform and mobile application to expand its reach and modernize its operations.

The new online platform will enable customers nationwide to browse Elmain's extensive catalog, order products for home delivery, or opt for same-day in-store pickup if the product is available. This move eliminates geographical constraints and allows the company to serve customers even in remote locations. Additionally, Elmain aims to stand out by embracing personalization—a growing trend in the fashion industry. Customers can collaborate with affiliated designers in one-on-one virtual sessions to create custom apparel that aligns with their tastes. To ensure a perfect fit, Elmain offers the option of uploading self-measured dimensions or scheduling an in-home visit by a certified tailor. These services emphasize a customer-first approach, blending convenience with the luxury of bespoke fashion.

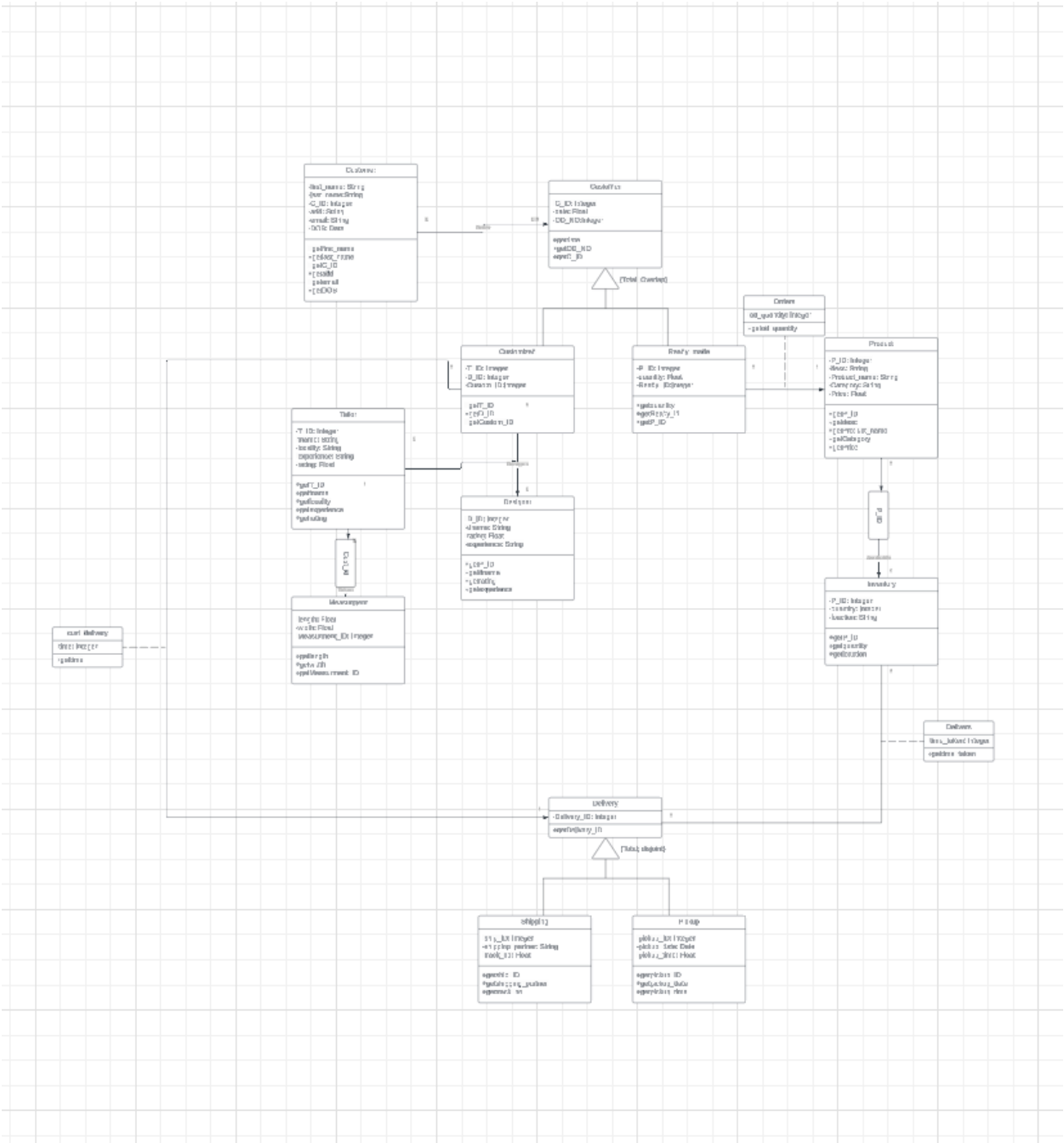
To support this initiative, Elmain will establish a robust operational framework. A comprehensive database will capture customer information, including demographic details, preferences, and purchasing history, to provide targeted offers and services. The company will also maintain detailed records of its designers, including their expertise, ratings, availability, and past projects, to streamline scheduling and uphold quality standards. Inventory management will be enhanced through a product database that tracks stock levels, pricing, and customer feedback, ensuring an efficient supply chain. Furthermore, Elmain will train and certify a network of tailors across the country, documenting their experience, credentials, and performance to provide high-quality, localized services.

2. Conceptual Data Modeling

1. EER Diagram



2. UML Diagram



3.Mapping Conceptual Model to Relational Model

Primary key - <u>Underlined</u>

Customer (<u>Customer_ID</u>, First_Name NOT NULL, Last_Name ,
Address, Email, State, Zip_Code, Date_of_Birth, Mobile_Number)

Designer (<u>Designer_ID</u>, Designer_Name NOT NULL, Rating, No_Customers_Designed, Experience_Level)

Tailor (<u>Tailor_ID</u>, Tailor_Name NOT NULL, Locality,
State, Zip_Code, Rating, Certification_Status, No_Customers_Worked, Experience)

Product (<u>Product_ID</u>, Product_Name NOT NULL, Category, Price,
Customer_Rating, Designer_ID (FK))

Store (<u>Store_ID</u>, Store_Location, Contact_Info, State, Zip_Code)

Inventory (<u>Store_ID</u> (FK), <u>Product_ID</u> (FK), Quantity)

Ready_Made (<u>Ready_Made_ID</u>, Product_ID (FK), Quantity)

Measurement (<u>Measurement_ID</u>, Length, Width)

Customized (<u>Custom_ID</u>, Tailor_ID (FK), Designer_ID (FK), Measurement_ID (FK))

Shipping (<u>Shipping_ID</u>, Shipping_Partner, Tracking_Number)

Pickup (<u>Pickup_ID</u>, Store_ID (FK), Pickup_Date,Pickup_Time_Slot)

Delivery (<u>Delivery_ID</u>, Delivery_Type NOT NULL, Date_Of_Delivery, Shipping_ID (FK), Pickup_ID (FK))

Orders (<u>Order_ID</u>, Customer_ID (FK), Order_Date, Sale_Type NOT NULL, Ready_Made_ID (FK), Custom_ID (FK), Delivery_ID (FK))

3. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:
The database was created in MySQL and the following queries were performed:

Query 1: Display Customer ID as ID, Frist Name as Name, State and date of birth from the customer table

```
SELECT Customer_ID AS ID, First_Name AS Name,
State, Date_of_Birth
FROM customer;
```

ID	Name	State	Date_of_Birth
1	Jose	New Jersey	1994-11-25
2	Joseph	Wisconsin	1953-09-28
3	Pamela	Montana	1958-05-06
4	David	Maine	1984-11-03
5	Kevin	Vermont	1962-11-11
6	Alice	North Dakota	1955-12-02

Query 2: Display unique shipping partner from the shipping table

```
SELECT DISTINCT Shipping_Partner
FROM shipping;
```

Shipping_Partner
FedEx
UPS
USPS
DHL
Amazon Logistics

Query 3: Display all tailors name whose name starts with letter 'A'

```
SELECT Tailor_Name
FROM tailor
WHERE TAILOR_NAME LIKE 'A%';
```

Tailor_Name
Abbie
Anna
Alexandro

Query 4: Display average product price in each product category

```
SELECT Category, AVG(PRICE) As Price_AVG
FROM product
GROUP BY Category;
```

Category	Price_AVG
Men's Apparel	122.813333
Women's Apparel	113.418571
Men's Apparel	135.050000
Outerwear	156.120000
Activewear	65.400000
Casual Wear	115.700000
Accessories	128.740000
Swimwear	115.190000

Query 5: Display combined price of all product in the category 'Outerwear'

```
SELECT SUM(Price) AS Total_Price
FROM product
WHERE Category='Outerwear';
```

Category	Price_AVG
Men's Apparel	122.813333
Women's Apparel	113.418571
Men's Apparel	135.050000
Outerwear	156.120000
Activewear	65.400000
Casual Wear	115.700000
Accessories	128.740000
Swimwear	115.190000

Query 6: Display name of the designer who designed the product with product id 'P0006' using inner join

```
SELECT D.Designer_Name
FROM Designer D, Product P
WHERE D.Designer_ID = P.Designer_ID AND P.Product_ID ='P0006';
```

Designer_Name
Avery Clark

Query 7: List all designers and the product they designed also include name of designer who have not designed any product

```
SELECT D.Designer_ID, D.Designer_Name, P.Product_Name
FROM Designer D
LEFT OUTER JOIN Product P
ON D.Designer_ID = P.Designer_ID;
```

Designer_ID	Designer_Name	Product_Name
D001	Aiden Schmidt	Cozy Knit Scarf
D002	Eliza Krajcik	Leather Biker Jacket
D003	Mason Goodman	NULL
D004	Olivia Wells	NULL
D005	Ethan Chang	Graphic Hoodie
D006	Sophia Choi	Cargo Work Pants
D007	Jackson Willis	Ruffled Party Dress

Query 8: Display designer name and rating who have rating higher than the average rating of all designers

```
SELECT Designer_Name, Rating
FROM Designer
WHERE Rating > (SELECT AVG(Rating)
FROM Designer);
```

Designer_Name	Rating
Aiden Schmidt	4.3
Mason Goodman	4.7
Sophia Choi	4.8
Lucas Pierce	4.5
James Davis	4.3
Charlotte Quinn	4.6

Query 9: Display customer name and their date of birth, who have placed at least one order

```
SELECT Customer_ID, First_Name, Last_Name,
Date_of_Birth
FROM Customer
WHERE Customer_ID IN (SELECT DISTINCT
Customer_ID
FROM Orders);
```

Customer_ID	First_Name	Last_Name	Date_of_Birth
12	Brian	Walters	1998-09-07
23	Jared	Gardner	1998-10-17
29	Christopher	Allen	1947-11-29
33	John	Johnson	1944-10-19
42	Thomas	Porter	1993-07-15

Query 10: Display customer name and state whose state match with any state which has a physical store

```
SELECT Customer_ID, First_Name, State
FROM Customer C
WHERE EXISTS (SELECT *
FROM Store S WHERE C.State = S.State);
```

Customer_ID	First_Name	State
12	Brian	Illinois
13	Sherri	California
17	Rachel	Nevada
57	Juan	California
59	Haley	Massachusetts

Query 11: Display customers names who have more than 2 tailors present in their state

```
SELECT First_Name, Last_Name, State
FROM Customer c
WHERE 2 < (SELECT COUNT(*)
FROM Tailor WHERE t.State = c.State);
```

First_Name	Last_Name	State
Kevin	Hall	North Carolina
Sandra	Terry	North Carolina

Query 12: Display name of designers who have worked on products which cost more than the average price of all product

```
SELECT Designer_Name
FROM Designer d
WHERE EXISTS (SELECT * FROM Product p
WHERE p.Designer_ID = d.Designer_ID
AND p.Price > (SELECT AVG(Price) FROM Product));
```

Designer_Name
Ava Moreno
Scarlett King
Liam Harris
James Davis
Eliza Krajcik

Query 13: Display names of tailors who have worked with designers who have rating above 4.5

```
SELECT t.Tailor_Name
FROM Tailor t
WHERE t.Tailor_ID IN (SELECT c.Tailor_ID
FROM Customized c
WHERE c.Designer_ID = ANY (SELECT d.Designer_ID
FROM Designer d
WHERE d.Rating > 4.5));
```

Tailor_Name
Abbie
Jamel
Sister
Penelope
Macey
Cordelia

Query 14: Display all unique customers who have place either only ‘customized’ or only ‘ready – made’

```
SELECT DISTINCT Customer_ID FROM Orders
WHERE Sale_Type = 'Customized' AND
Customer_ID NOT IN (SELECT Customer_ID
FROM Orders
WHERE Sale_Type = 'Ready-made')
UNION
SELECT DISTINCT Customer_ID
FROM Orders
WHERE Sale_Type = 'Ready-made' AND
Customer_ID NOT IN (SELECT Customer_ID FROM Orders
WHERE Sale_Type = 'Customized');
```

Customer_ID
12
29
33
45
56
57
58

Query 15: Display customer_ID, First_Name and the number of orders placed by each customer of top 5 customers who made the highest number of orders

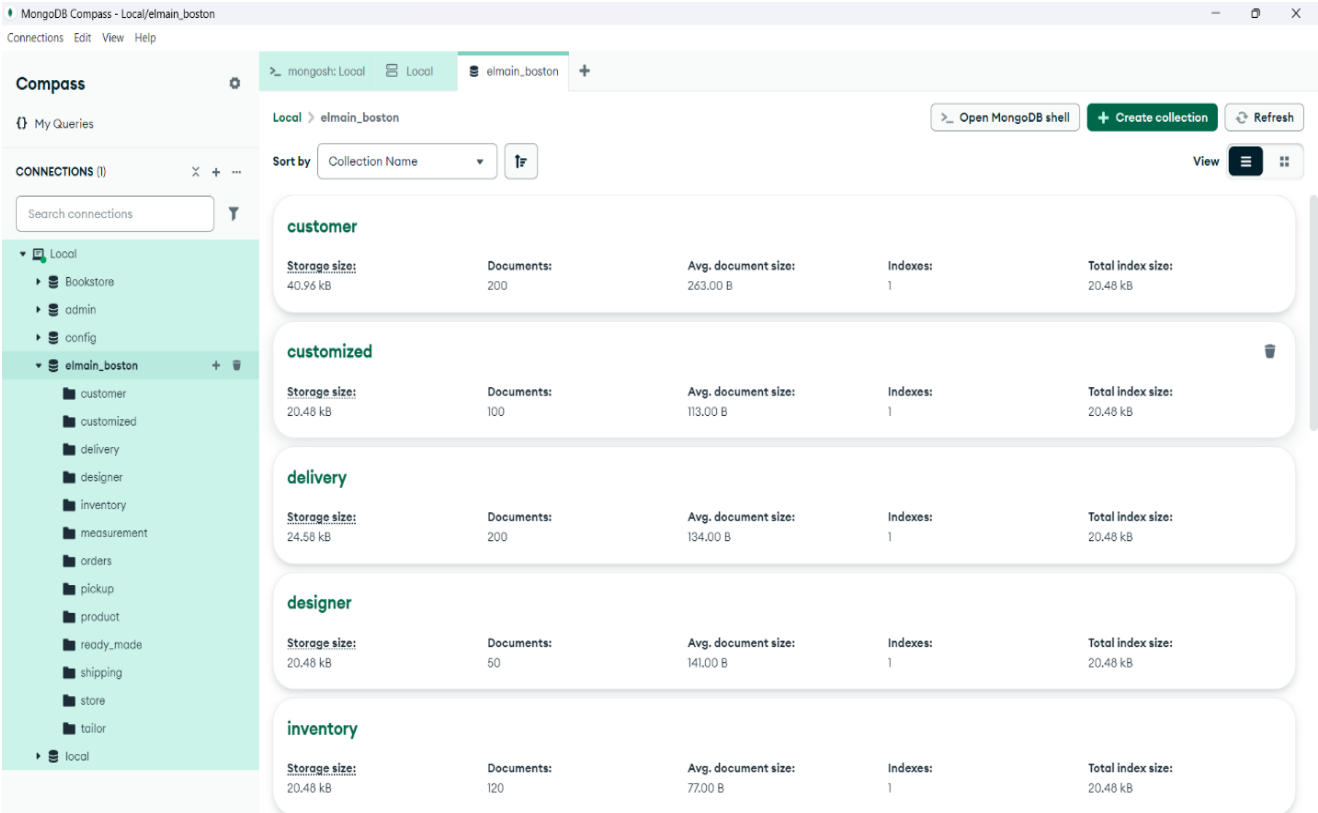
```
SELECT C.Customer_ID, C.First_Name, C.Last_Name,
(SELECT COUNT(*) FROM Orders O
WHERE O.Customer_ID = C.Customer_ID) AS Order_Count
FROM Customer C
ORDER BY Order_Count DESC
LIMIT 5;
```

Customer_ID	First_Name	Last_Name	Order_Count
115	Dalton	Anderson	6
110	Jeff	Montoya	6
120	Marilyn	Torres	5
103	Stephanie	Rivera	4
112	Ryan	Sanchez	4

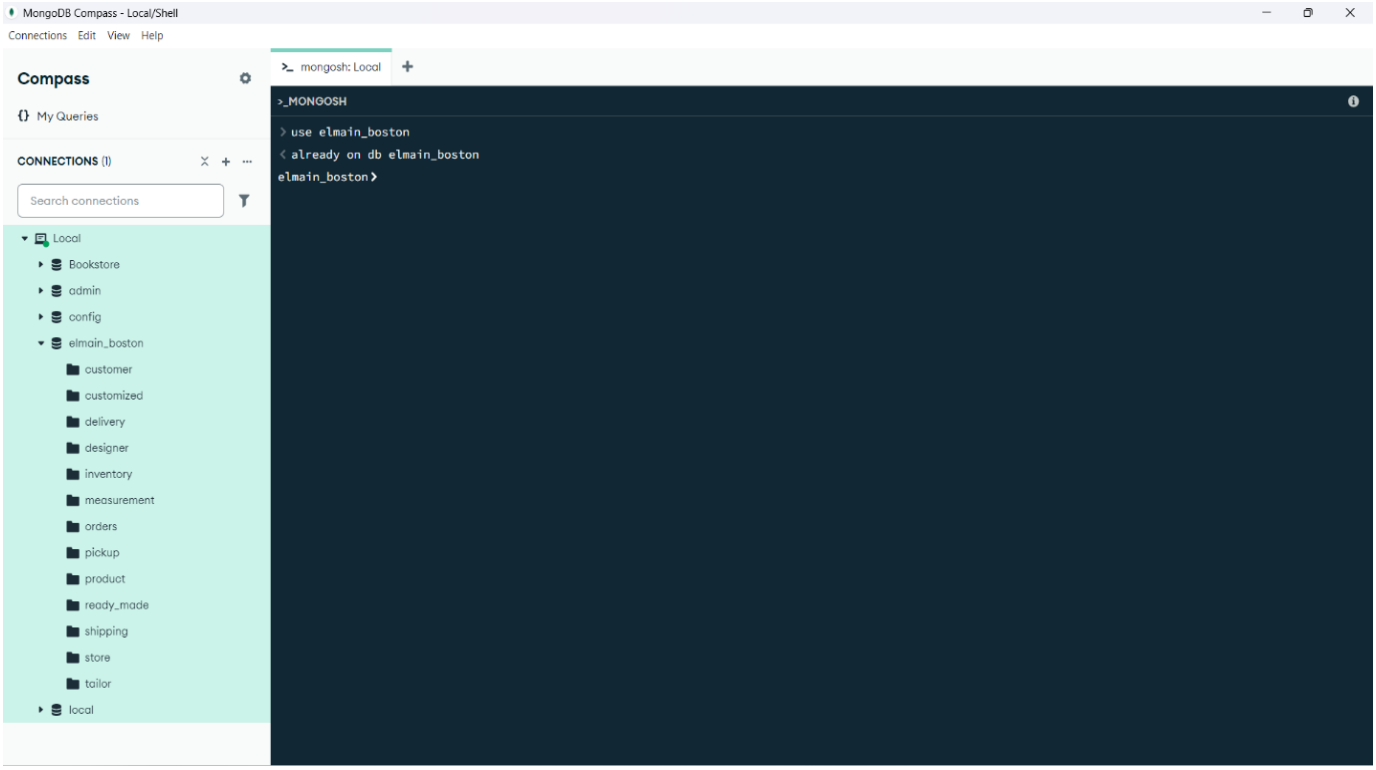
NoSql Implementation:

All table and records from MySQL database of Elmain Boston is transferred and hosted in MongoDB. The following Mongo queries were done:

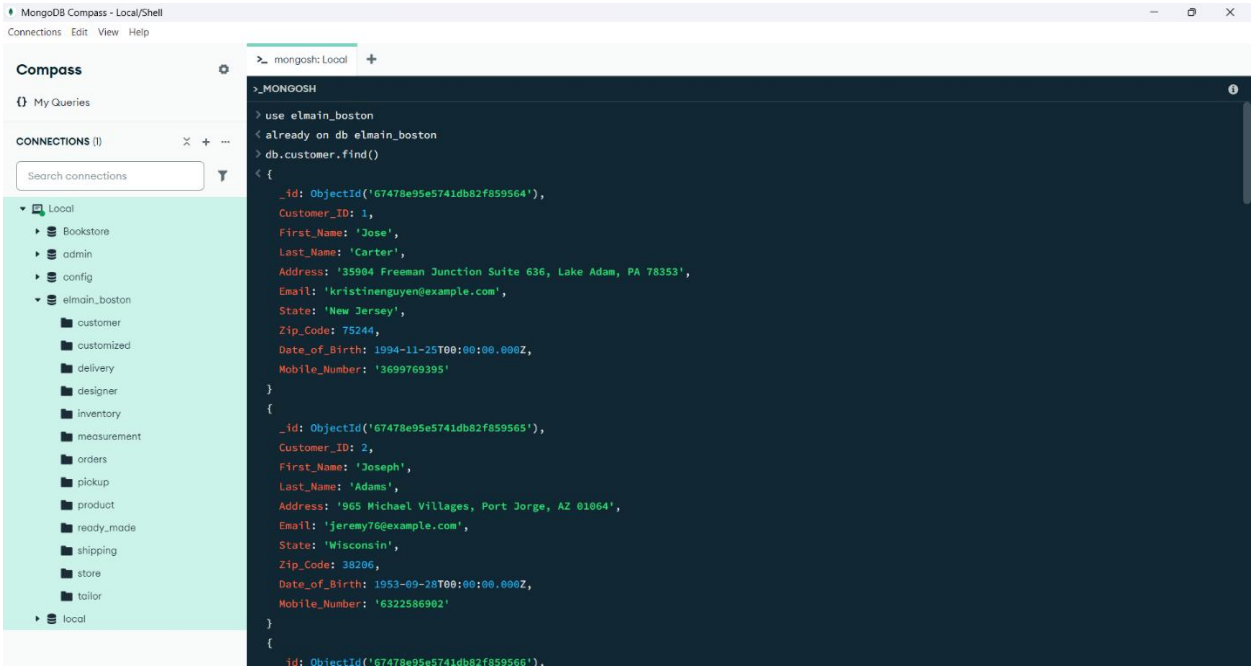
The **Elmain_Boston** database is hosted and operating in MongoDB platform



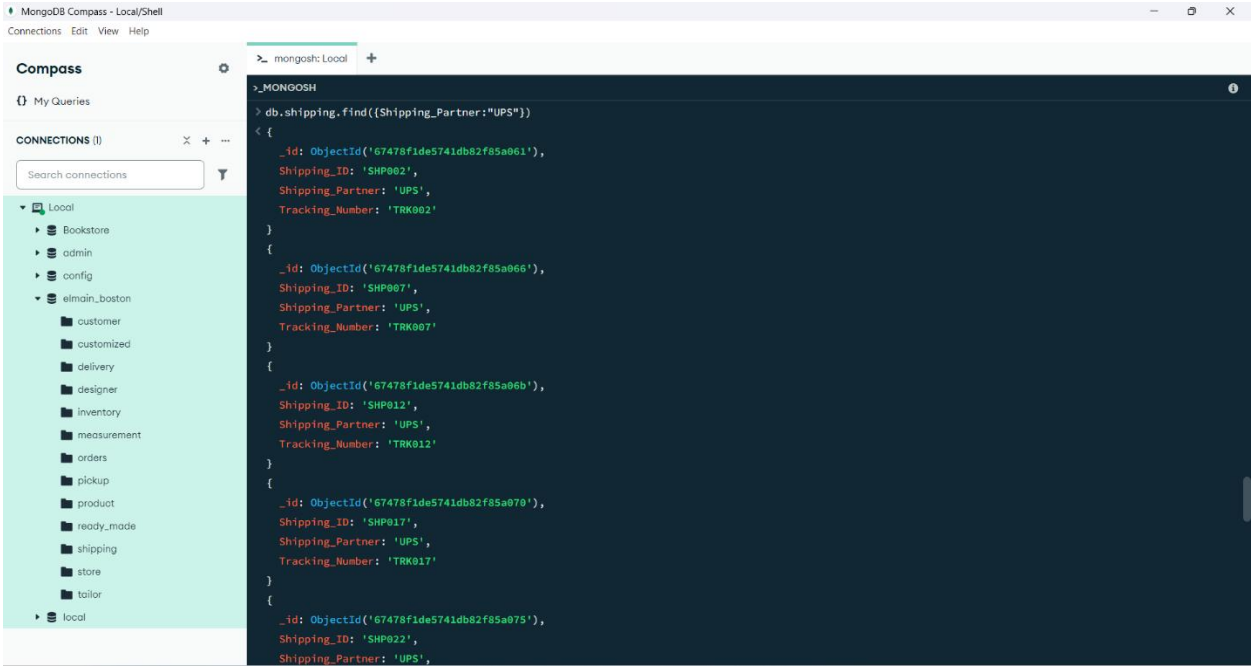
Command to use the elmain_boston database



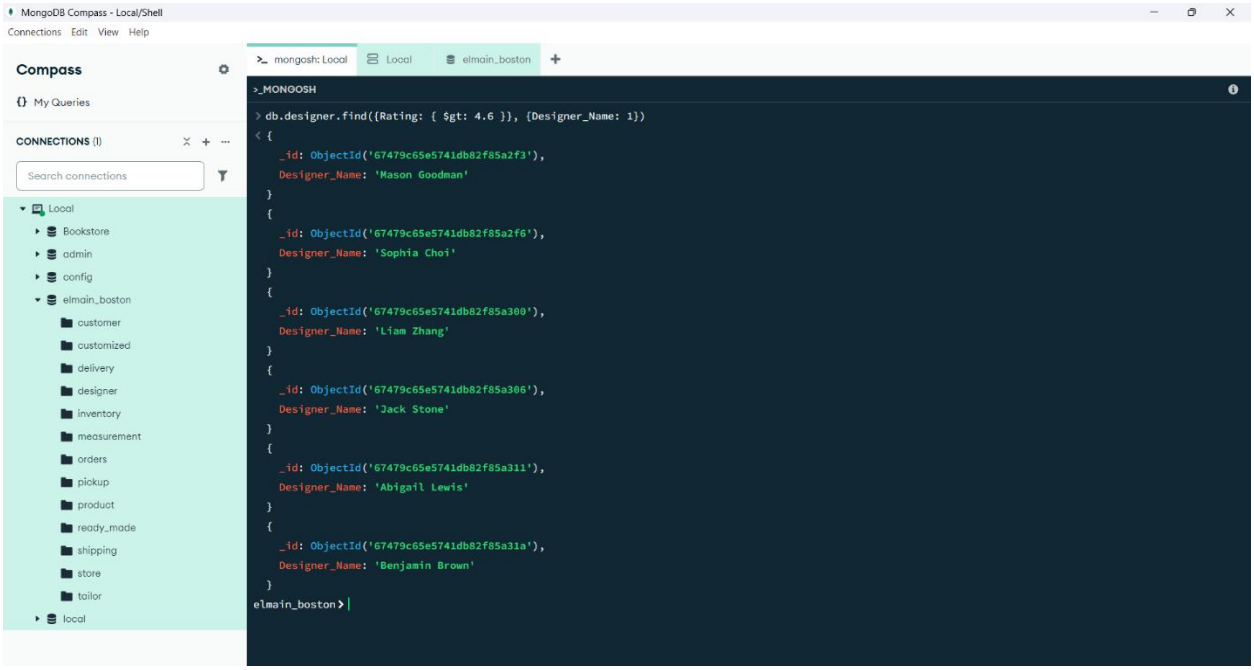
Query to display all documents in the Customer Collection



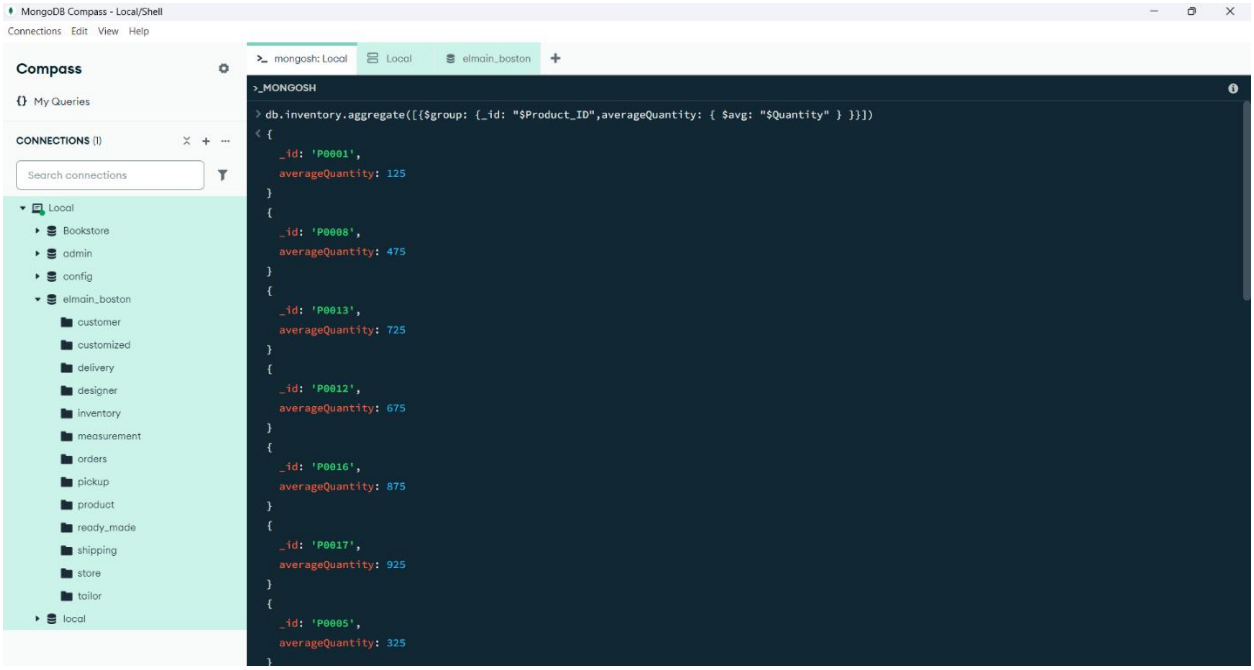
Query to display all documents from Shipping collection which has Shipping Partner as ‘UPS’



Query to display the Name of the Designers who have rating more than 4.6 in Designer Collection



Query to find the average quantity of each product in all stores in the Inventory collection



Database Access via Python

Database Application using Python

Group 5

Ragul Narayanan Magesh & Barath Keshav Sriram Kumaran

```python
# 1. Displaying distinct Names of Shipping Partners

import mysql.connector

try:
    connection = mysql.connector.connect(
        host='127.0.0.1',
        database='elmain_boston',
        user='root',
        password='IE6700',
        auth_plugin='caching_sha2_password'
    )

    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record, "\n")

        # Perform SQL query to select distinct shipping partners
        sql_select_Query = "SELECT DISTINCT Shipping_Partner FROM
shipping"
        cursor.execute(sql_select_Query)
        records = cursor.fetchall()

        print("Distinct Shipping Partner:\n")
        for row in records:
            print('Shipping_Partner =', row[0], "\n")

except mysql.connector.Error as err:
    print("Error: ", err)

finally:
    if connection.is_connected():
        connection.close()
        print("MySQL connection is closed.")
```

```
Connected to MySQL Server version  8.0.40
You're connected to database:  ('elmain_boston',)

Distinct Shipping Partner:
```

```
Shipping_Partner = FedEx

Shipping_Partner = UPS

Shipping_Partner = USPS

Shipping_Partner = DHL

Shipping_Partner = Amazon Logistics

MySQL connection is closed.
```

```python
# 2. Displaying Name of the designer who designed the product P0006

import mysql.connector

try:
    connection = mysql.connector.connect(
        host='127.0.0.1',
        database='elmain_boston',
        user='root',
        password='IE6700',
        auth_plugin='caching_sha2_password'
    )

    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record, "\n")

        # Perform SQL query to select Designer name who desined the
product with product id P0006
        sql_select_Query = "SELECT D.Designer_Name FROM Designer D,
Product P WHERE D.Designer_ID = P.Designer_ID AND P.Product_ID
='P0006'"
        cursor.execute(sql_select_Query)
        records = cursor.fetchall()

        print("Designer of Product P0006:\n")
        for row in records:
            print('Designer Name =', row[0], "\n")

except mysql.connector.Error as err:
    print("Error: ", err)

finally:
    if connection.is_connected():
```

```
        connection.close()
        print("MySQL connection is closed.")
```

```
Connected to MySQL Server version  8.0.40
You're connected to database:  ('elmain_boston',)

Designer of Product P0006:

Designer Name = Avery Clark

MySQL connection is closed.
```

```
# 3. Displaying Tailors Name who have experience greater than the
average experience

import mysql.connector

try:
    connection = mysql.connector.connect(
        host='127.0.0.1',
        database='elmain_boston',
        user='root',
        password='IE6700',
        auth_plugin='caching_sha2_password'
    )

    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record, "\n")

        # Perform SQL query to Display Tailors who have experience
greater than the average experience
        sql_select_Query = "SELECT Tailor_Name, Experience, State FROM
Tailor t1 WHERE Experience > ( SELECT AVG(Experience) FROM Tailor t2
WHERE t2.State = t1.State)"
        cursor.execute(sql_select_Query)
        records = cursor.fetchall()

        print("Tailors who have experience greater than the average
experience:\n")
        for row in records:
            print('Tailor Name =', row[0], "\n")

except mysql.connector.Error as err:
    print("Error: ", err)
```

```
finally:
    if connection.is_connected():
        connection.close()
        print("MySQL connection is closed.")
```

```
Connected to MySQL Server version  8.0.40
You're connected to database:  ('elmain_boston',)

Tailors who have experience greater than the average experience:

Tailor Name = Jamel

Tailor Name = Salvatore

Tailor Name = Burnice

Tailor Name = Brisa

Tailor Name = Kurt

Tailor Name = Hilario

Tailor Name = Serena

Tailor Name = Bobbie

Tailor Name = Ila

Tailor Name = Alexandro

Tailor Name = Sim

Tailor Name = Jaron

MySQL connection is closed.
```

```
# 4. Displaying State name and number of customers in that State
import pandas as pd
import mysql.connector


try:
    connection = mysql.connector.connect(
        host='127.0.0.1',
        database='elmain_boston',
        user='root',
        password='IE6700',
        auth_plugin='caching_sha2_password'
    )

    if connection.is_connected():
```

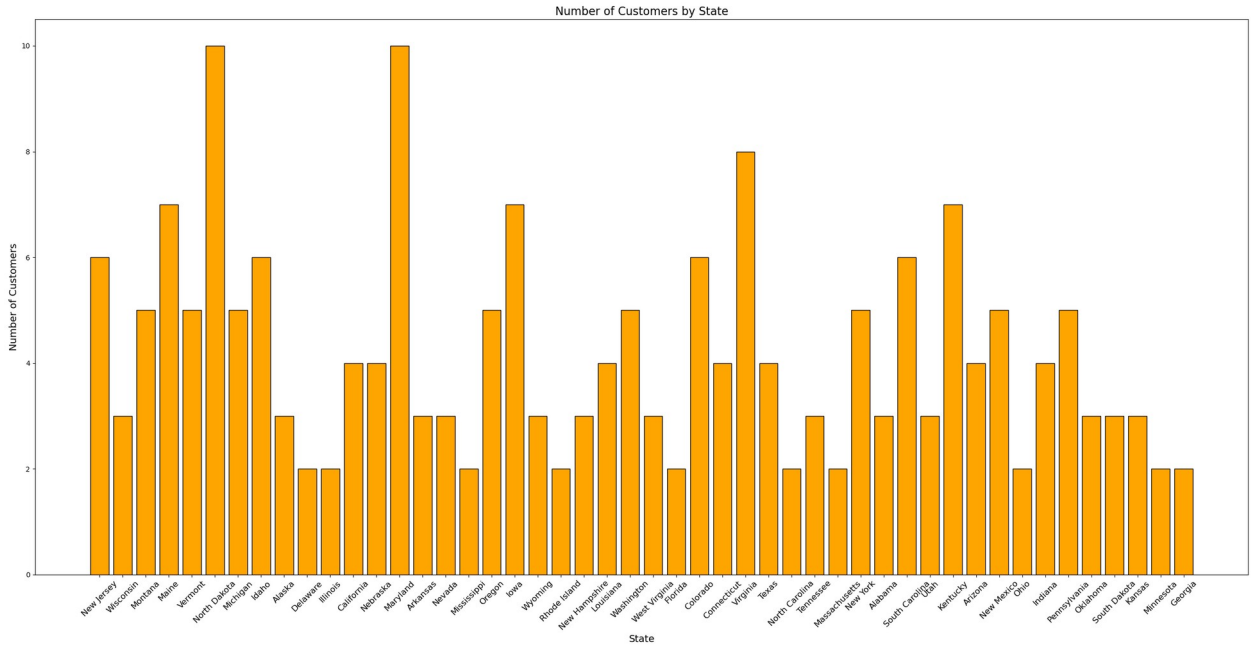
```
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record1 = cursor.fetchone()
        print("You're connected to database: ", record1, "\n")

        # Perform SQL query to Display State and number of customers
in that particular State
        sql_select_Query = "SELECT State, COUNT(Customer_ID) AS
Number_of_Customers FROM customer GROUP BY State;"
        cursor.execute(sql_select_Query)
        record1 = cursor.fetchall()

        print("State and number of customers in that particular
State:\n")
        for row in record1:
            print( "State:", row[0], ", No.of customers:", row[1], "\
n")

except mysql.connector.Error as err:
    print("Error: ", err)

finally:
    if connection.is_connected():
        connection.close()
```

```
Connected to MySQL Server version  8.0.40
You're connected to database:  ('elmain_boston',)

State and number of customers in that particular State:

State: New Jersey , No.of customers: 6

State: Wisconsin , No.of customers: 3

State: Montana , No.of customers: 5

State: Maine , No.of customers: 7

State: Vermont , No.of customers: 5

State: North Dakota , No.of customers: 10

State: Michigan , No.of customers: 5

State: Idaho , No.of customers: 6

State: Alaska , No.of customers: 3

State: Delaware , No.of customers: 2
```

```
State: Illinois , No.of customers: 2
State: California , No.of customers: 4
State: Nebraska , No.of customers: 4
State: Maryland , No.of customers: 10
State: Arkansas , No.of customers: 3
State: Nevada , No.of customers: 3
State: Mississippi , No.of customers: 2
State: Oregon , No.of customers: 5
State: Iowa , No.of customers: 7
State: Wyoming , No.of customers: 3
State: Rhode Island , No.of customers: 2
State: New Hampshire , No.of customers: 3
State: Louisiana , No.of customers: 4
State: Washington , No.of customers: 5
State: West Virginia , No.of customers: 3
State: Florida , No.of customers: 2
State: Colorado , No.of customers: 6
State: Connecticut , No.of customers: 4
State: Virginia , No.of customers: 8
State: Texas , No.of customers: 4
State: North Carolina , No.of customers: 2
State: Tennessee , No.of customers: 3
State: Massachusetts , No.of customers: 2
State: New York , No.of customers: 5
State: Alabama , No.of customers: 3
State: South Carolina , No.of customers: 6
```

```
State: Utah , No.of customers: 3

State: Kentucky , No.of customers: 7

State: Arizona , No.of customers: 4

State: New Mexico , No.of customers: 5

State: Ohio , No.of customers: 2

State: Indiana , No.of customers: 4

State: Pennsylvania , No.of customers: 5

State: Oklahoma , No.of customers: 3

State: South Dakota , No.of customers: 3

State: Kansas , No.of customers: 3

State: Minnesota , No.of customers: 2

State: Georgia , No.of customers: 2
```

```
# 4.1 Plotting bar chart to display Number of customers in a State

import matplotlib.pyplot as plt

states = [row[0] for row in record1]
customer_counts = [row[1] for row in record1]

# Create the bar chart
plt.figure(figsize=(25, 13))
plt.bar(states, customer_counts, color='orange', edgecolor='black')
plt.title("Number of Customers by State", fontsize=16)
plt.xlabel("State", fontsize=14)
plt.ylabel("Number of Customers", fontsize=14)
plt.xticks(rotation=45, fontsize=12)
plt.tight_layout()
plt.show()
```




```python
# 5. Displaying CustomerId and Age of the customer

import pandas as pd
import mysql.connector


try:
    connection = mysql.connector.connect(
        host='127.0.0.1',
        database='elmain_boston',
        user='root',
        password='IE6700',
        auth_plugin='caching_sha2_password'
    )

    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record, "\n")

        # Perform SQL query to Display CustomerID and Age
        sql_select_Query = "SELECT
Customer_ID,FLOOR(DATEDIFF(CURDATE(), Date_of_Birth) / 365) AS Age
FROM Customer"
        cursor.execute(sql_select_Query)
        record2 = cursor.fetchall()
```

```
        print("CustomerID and the age of the customer:\n")
        for row in record2:
            print( "CustomerID:", row[0], ", Age:", row[1], "\n")

except mysql.connector.Error as err:
    print("Error: ", err)

finally:
    if connection.is_connected():
        connection.close()
```

```
Connected to MySQL Server version  8.0.40
You're connected to database:  ('elmain_boston',)

CustomerID and the age of the customer:

CustomerID: 1 , Age: 30

CustomerID: 2 , Age: 71

CustomerID: 3 , Age: 66

CustomerID: 4 , Age: 40

CustomerID: 5 , Age: 62

CustomerID: 6 , Age: 69

CustomerID: 7 , Age: 20

CustomerID: 8 , Age: 57

CustomerID: 9 , Age: 30

CustomerID: 10 , Age: 62

CustomerID: 11 , Age: 28

CustomerID: 12 , Age: 26

CustomerID: 13 , Age: 36

CustomerID: 14 , Age: 36

CustomerID: 15 , Age: 48

CustomerID: 16 , Age: 37

CustomerID: 17 , Age: 79

CustomerID: 18 , Age: 34
```

```
CustomerID: 19 , Age: 20
CustomerID: 20 , Age: 78
CustomerID: 21 , Age: 31
CustomerID: 22 , Age: 68
CustomerID: 23 , Age: 26
CustomerID: 24 , Age: 46
CustomerID: 25 , Age: 81
CustomerID: 26 , Age: 80
CustomerID: 27 , Age: 78
CustomerID: 28 , Age: 20
CustomerID: 29 , Age: 77
CustomerID: 30 , Age: 66
CustomerID: 31 , Age: 71
CustomerID: 32 , Age: 65
CustomerID: 33 , Age: 80
CustomerID: 34 , Age: 42
CustomerID: 35 , Age: 61
CustomerID: 36 , Age: 31
CustomerID: 37 , Age: 33
CustomerID: 38 , Age: 71
CustomerID: 39 , Age: 40
CustomerID: 40 , Age: 28
CustomerID: 41 , Age: 54
CustomerID: 42 , Age: 31
CustomerID: 43 , Age: 78
CustomerID: 44 , Age: 63
```

```
CustomerID: 45 , Age: 78
CustomerID: 46 , Age: 67
CustomerID: 47 , Age: 30
CustomerID: 48 , Age: 48
CustomerID: 49 , Age: 21
CustomerID: 50 , Age: 71
CustomerID: 51 , Age: 23
CustomerID: 52 , Age: 49
CustomerID: 53 , Age: 35
CustomerID: 54 , Age: 71
CustomerID: 55 , Age: 50
CustomerID: 56 , Age: 43
CustomerID: 57 , Age: 70
CustomerID: 58 , Age: 27
CustomerID: 59 , Age: 76
CustomerID: 60 , Age: 64
CustomerID: 61 , Age: 25
CustomerID: 62 , Age: 18
CustomerID: 63 , Age: 61
CustomerID: 64 , Age: 46
CustomerID: 65 , Age: 80
CustomerID: 66 , Age: 20
CustomerID: 67 , Age: 29
CustomerID: 68 , Age: 28
CustomerID: 69 , Age: 37
CustomerID: 70 , Age: 69
```

```
CustomerID: 71 , Age: 49
CustomerID: 72 , Age: 47
CustomerID: 73 , Age: 24
CustomerID: 74 , Age: 24
CustomerID: 75 , Age: 77
CustomerID: 76 , Age: 41
CustomerID: 77 , Age: 37
CustomerID: 78 , Age: 31
CustomerID: 79 , Age: 79
CustomerID: 80 , Age: 33
CustomerID: 81 , Age: 59
CustomerID: 82 , Age: 23
CustomerID: 83 , Age: 42
CustomerID: 84 , Age: 24
CustomerID: 85 , Age: 20
CustomerID: 86 , Age: 57
CustomerID: 87 , Age: 46
CustomerID: 88 , Age: 38
CustomerID: 89 , Age: 54
CustomerID: 90 , Age: 25
CustomerID: 91 , Age: 79
CustomerID: 92 , Age: 43
CustomerID: 93 , Age: 46
CustomerID: 94 , Age: 40
CustomerID: 95 , Age: 71
CustomerID: 96 , Age: 33
```

```
CustomerID: 97 , Age: 37
CustomerID: 98 , Age: 67
CustomerID: 99 , Age: 26
CustomerID: 100 , Age: 44
CustomerID: 101 , Age: 33
CustomerID: 102 , Age: 49
CustomerID: 103 , Age: 58
CustomerID: 104 , Age: 65
CustomerID: 105 , Age: 31
CustomerID: 106 , Age: 63
CustomerID: 107 , Age: 81
CustomerID: 108 , Age: 20
CustomerID: 109 , Age: 75
CustomerID: 110 , Age: 78
CustomerID: 111 , Age: 77
CustomerID: 112 , Age: 78
CustomerID: 113 , Age: 36
CustomerID: 114 , Age: 79
CustomerID: 115 , Age: 80
CustomerID: 116 , Age: 63
CustomerID: 117 , Age: 43
CustomerID: 118 , Age: 37
CustomerID: 119 , Age: 52
CustomerID: 120 , Age: 41
CustomerID: 121 , Age: 64
CustomerID: 122 , Age: 75
```

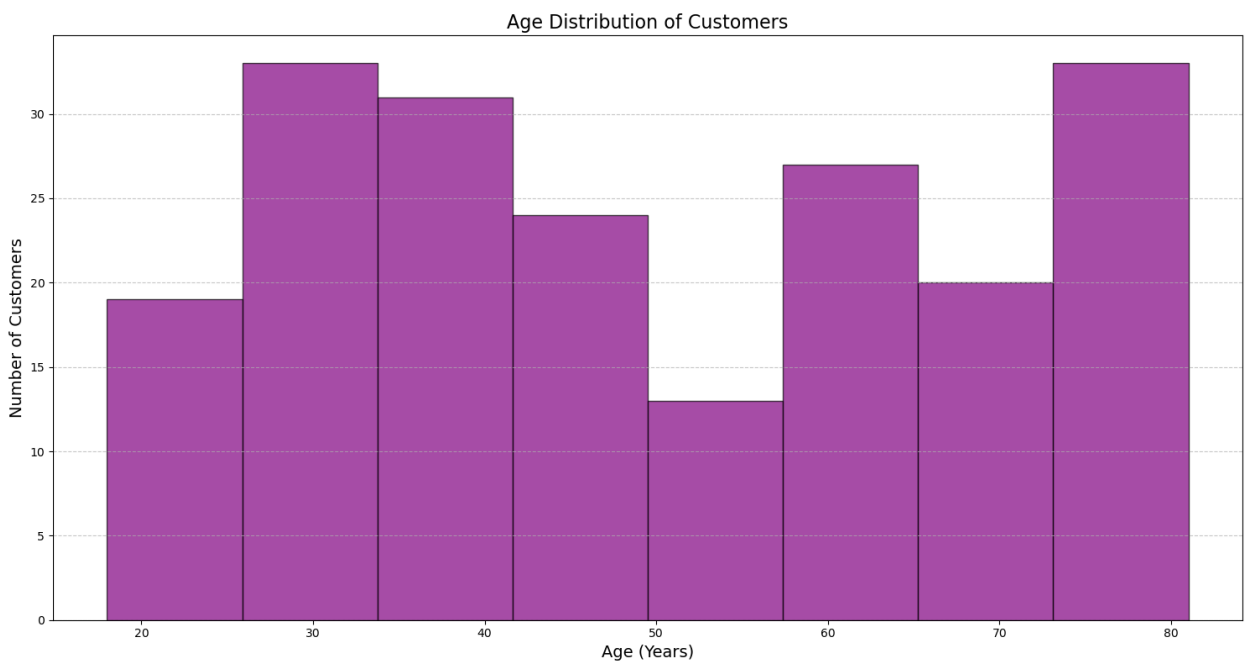
```
CustomerID: 123 , Age: 42
CustomerID: 124 , Age: 78
CustomerID: 125 , Age: 32
CustomerID: 126 , Age: 61
CustomerID: 127 , Age: 35
CustomerID: 128 , Age: 54
CustomerID: 129 , Age: 50
CustomerID: 130 , Age: 35
CustomerID: 131 , Age: 36
CustomerID: 132 , Age: 22
CustomerID: 133 , Age: 40
CustomerID: 134 , Age: 51
CustomerID: 135 , Age: 31
CustomerID: 136 , Age: 25
CustomerID: 137 , Age: 68
CustomerID: 138 , Age: 62
CustomerID: 139 , Age: 76
CustomerID: 140 , Age: 76
CustomerID: 141 , Age: 53
CustomerID: 142 , Age: 63
CustomerID: 143 , Age: 33
CustomerID: 144 , Age: 39
CustomerID: 145 , Age: 68
CustomerID: 146 , Age: 30
CustomerID: 147 , Age: 46
CustomerID: 148 , Age: 59
```

```
CustomerID: 149 , Age: 29
CustomerID: 150 , Age: 31
CustomerID: 151 , Age: 27
CustomerID: 152 , Age: 78
CustomerID: 153 , Age: 69
CustomerID: 154 , Age: 40
CustomerID: 155 , Age: 70
CustomerID: 156 , Age: 43
CustomerID: 157 , Age: 77
CustomerID: 158 , Age: 64
CustomerID: 159 , Age: 39
CustomerID: 160 , Age: 64
CustomerID: 161 , Age: 30
CustomerID: 162 , Age: 60
CustomerID: 163 , Age: 37
CustomerID: 164 , Age: 74
CustomerID: 165 , Age: 81
CustomerID: 166 , Age: 58
CustomerID: 167 , Age: 66
CustomerID: 168 , Age: 34
CustomerID: 169 , Age: 20
CustomerID: 170 , Age: 58
CustomerID: 171 , Age: 26
CustomerID: 172 , Age: 26
CustomerID: 173 , Age: 40
CustomerID: 174 , Age: 56
```

```
CustomerID: 175 , Age: 38
CustomerID: 176 , Age: 76
CustomerID: 177 , Age: 80
CustomerID: 178 , Age: 51
CustomerID: 179 , Age: 61
CustomerID: 180 , Age: 57
CustomerID: 181 , Age: 42
CustomerID: 182 , Age: 26
CustomerID: 183 , Age: 42
CustomerID: 184 , Age: 41
CustomerID: 185 , Age: 36
CustomerID: 186 , Age: 49
CustomerID: 187 , Age: 44
CustomerID: 188 , Age: 59
CustomerID: 189 , Age: 49
CustomerID: 190 , Age: 70
CustomerID: 191 , Age: 81
CustomerID: 192 , Age: 60
CustomerID: 193 , Age: 27
CustomerID: 194 , Age: 32
CustomerID: 195 , Age: 37
CustomerID: 196 , Age: 76
CustomerID: 197 , Age: 62
CustomerID: 198 , Age: 18
CustomerID: 199 , Age: 62
```

```
CustomerID: 200 , Age: 34
```

```python
# 5.1 Plotting Histogram to display Customer Age Distribution

import matplotlib.pyplot as plt

ages = [row[1] for row in record2]
plt.figure(figsize=(15, 8))
plt.hist(ages, bins=8, color='purple', edgecolor='black', alpha=0.7)
plt.title("Age Distribution of Customers", fontsize=16)
plt.xlabel("Age (Years)", fontsize=14)
plt.ylabel("Number of Customers", fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



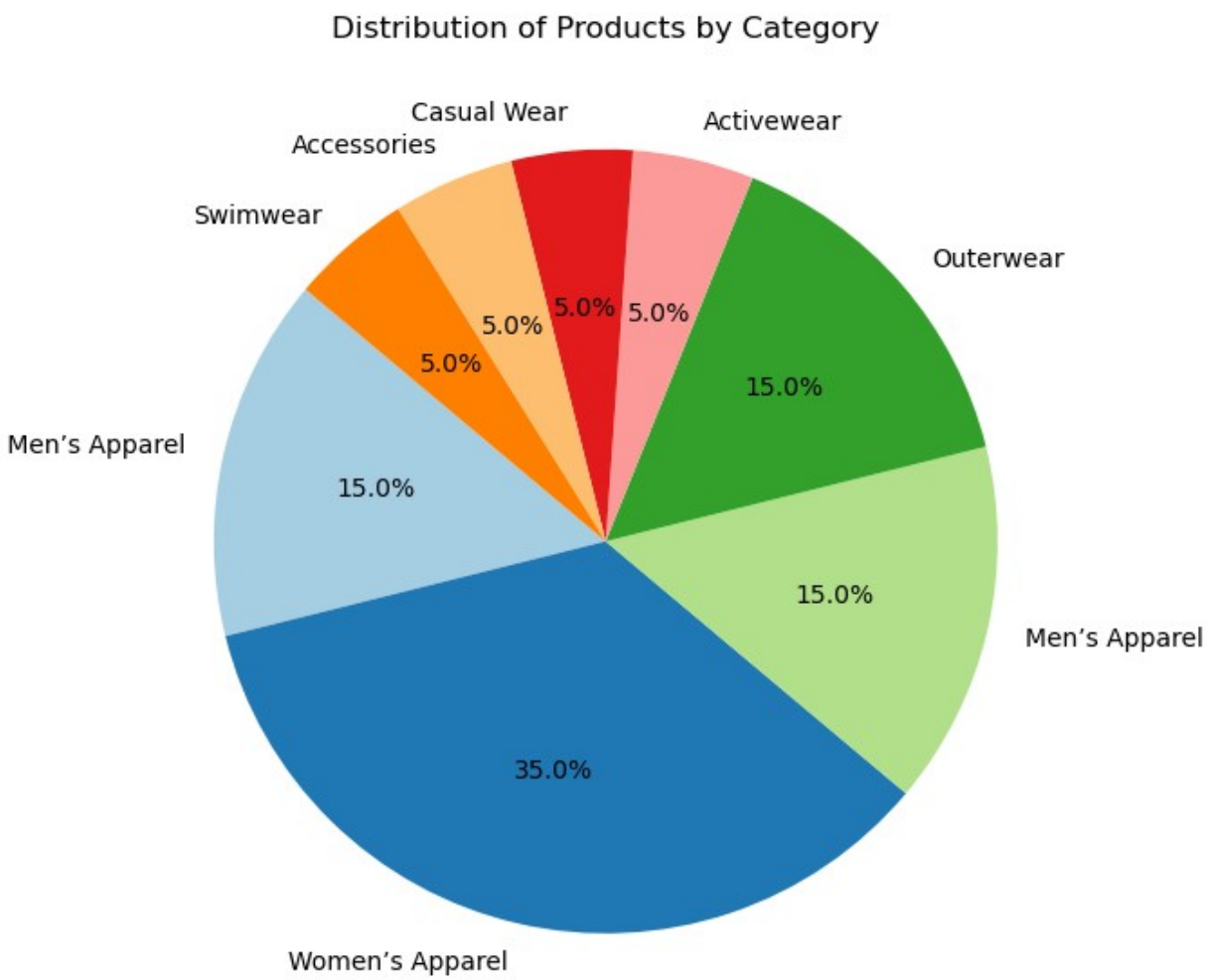

```python
# 6. Displaying Product Category and number of Products in that
particular Category

import pandas as pd
import mysql.connector


try:
    connection = mysql.connector.connect(
        host='127.0.0.1',
        database='elmain_boston',
        user='root',
```

```
        password='IE6700',
        auth_plugin='caching_sha2_password'
    )

    if connection.is_connected():
        db_Info = connection.get_server_info()
        print("Connected to MySQL Server version ", db_Info)
        cursor = connection.cursor()
        cursor.execute("select database();")
        record = cursor.fetchone()
        print("You're connected to database: ", record, "\n")

        # Perform SQL query to Display Category and count of products
        sql_select_Query = "SELECT Category, COUNT(*) AS Product_Count
FROM Product GROUP BY Category;"
        cursor.execute(sql_select_Query)
        record3 = cursor.fetchall()

        print("Product category and number of products in that
category:\n")
        for row in record3:
            print( "Category:", row[0], ", No. of Products:", row[1],
"\n")

except mysql.connector.Error as err:
    print("Error: ", err)

finally:
    if connection.is_connected():
        connection.close()
```

```
Connected to MySQL Server version  8.0.40
You're connected to database:  ('elmain_boston',)

Product category and number of products in that category:

Category:  Men’s Apparel , No. of Products: 3

Category: Women’s Apparel , No. of Products: 7

Category: Men’s Apparel , No. of Products: 3

Category: Outerwear , No. of Products: 3

Category: Activewear , No. of Products: 1

Category: Casual Wear , No. of Products: 1

Category: Accessories , No. of Products: 1
```

```
Category: Swimwear , No. of Products: 1
```

```python
# 6.1 Pie chart to show distribution of products by Category

import matplotlib.pyplot as plt
df = pd.DataFrame(record3, columns=['Category', 'Product_Count'])


# Plotting the bar chart
plt.figure(figsize=(7, 10))
plt.pie(df['Product_Count'], labels=df['Category'], autopct='%1.1f%%',
startangle=140, colors=plt.cm.Paired.colors)
plt.title('Distribution of Products by Category')
plt.show()
```



7. Summary And Recommendation

The Elmain Boston project is a transformative initiative aimed at modernizing the operations of a 75-year-old luxury clothing brand to address challenges posed by shifting consumer behaviors and declining in-store traffic. Historically reliant on its six flagship stores in major U.S. cities, Elmain has faced difficulties adapting to the rise of e-commerce and growing demand for personalized shopping experiences. In response, the brand is launching an e-commerce platform and mobile application to expand its reach nationwide. This platform allows customers to explore products, place orders for home delivery or in-store pickup, and collaborate with fashion designers in virtual sessions to create customized apparel. Measurement services, either self-reported or provided through in-home tailor visits, enhance the personalized experience further.

Supporting this transformation is a robust operational framework, including databases to manage customer information, designer schedules, inventory, order tracking, and tailor assignments. Additionally, the platform integrates advanced analytics capabilities to offer insights into customer behavior, preferences, and operational efficiency. This initiative positions Elmain Boston to regain its market relevance by combining its heritage of luxury with innovative digital solutions, ultimately enhancing customer satisfaction and fostering long-term growth.

To ensure the success and sustainability of this transformation, Elmain Boston should focus on implementing robust data governance measures. This will enhance data security, maintain accuracy, and ensure compliance with privacy regulations, building trust among customers. The brand should also invest in scalable cloud-based infrastructure to accommodate growing demand and seamlessly integrate future services. Expanding the network of certified tailors across the country will help maintain high service standards, ensuring timely and reliable measurements for customers seeking personalized clothing.

Marketing efforts should focus on promoting the e-commerce platform and its unique features, leveraging targeted campaigns and collaborations with designers and influencers to attract a younger, tech-savvy audience. Continuous feedback from customers post-launch will be crucial for identifying areas of improvement, allowing the platform to evolve with customer needs. Finally, incorporating sustainability and ethical practices into operations and marketing will align with modern consumer values, further solidifying Elmain Boston's position as a forward-thinking, customer-centric luxury brand.