Data Analyst Assignment: Customer & Operations Analysis

Prepared By: Sharmila Dwarapureddy

Submission Date: 20-02-2025

📝 1. Introduction

- **Purpose:** Analyse customer behaviour, inventory, and discount impact for an online grocery business.
- **Data Sources:** Sales data, inventory data, discount campaign data, user orders.
- **Tools Used:** Python, Excel, SQL.

📎 2. Appendix

- **References** to data sources (orders.csv, inventory.csv, discount_campaign.csv).
- **Tools Used:** Python (Pandas, Seaborn, Matplotlib, Sklearn), Excel, SQL (SQLite).

📊 3. Python Analysis

A. Customer Segmentation (K-Means Clustering)

Objective:

Segment customers based on their spending, order frequency, and recency.

Key Steps:

- Aggregated customer metrics: **Total Spend, Order Count, Recency (days since last order)**.
- Scaled data using **StandardScaler**.
- Applied **K-Means Clustering (k=3)** to group customers.
- Visualized the clusters using a **scatter plot (Total Spend vs. Order Count)**.
- Visualized the clusters using a **scatter plot (Recency vs. Total Spend)**
- Visualized Line chart of **Monthly Revenue Trend**

Key Insight:

- Cluster 1: **High Spend, Frequent Orders (Loyal Customers)**.
- Cluster 2: **Moderate Spend, Occasional Orders**.
- Cluster 3: **Low Spend, Infrequent Orders**.

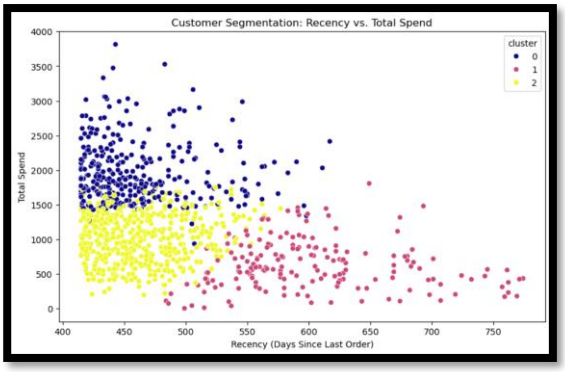
Sample code:

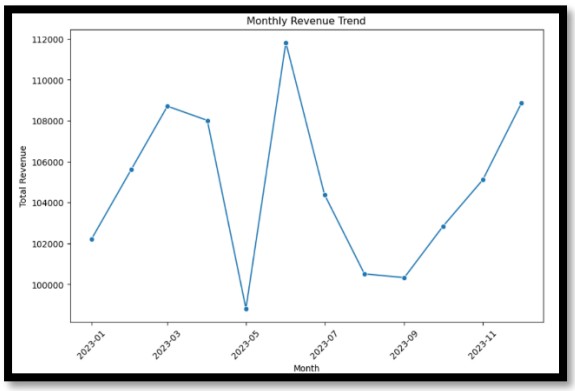
```
# Apply K-Means clustering (e.g., 3 clusters)
k = 3
kmeans = KMeans(n_clusters=k, random_state=42)
customer_metrics['cluster'] = kmeans.fit_predict(scaled_features)

# Check cluster distribution
print(customer_metrics['cluster'].value_counts())

# Visualize clusters: Total Spend vs. Order Count
plt.figure(figsize=(10, 6))
sns.scatterplot(
   x='total_spend',
   y='order_count',
   hue='cluster',
   data=customer_metrics,
   palette='viridis'
)
plt.title('Customer Segmentation: Total Spend vs. Order Count')
plt.xlabel('Total Spend')
plt.ylabel('Order Count')
plt.show()

# Convert order_date to Period and then to Timestamp
orders['order_month'] = orders['order_date'].dt.to_period('M')
orders['order_month'] = orders['order_month'].dt.to_timestamp()

# Aggregate revenue by month
monthly_revenue = orders.groupby('order_month')['total_amount'].sum().reset_index()

plt.figure(figsize=(10, 6))
```

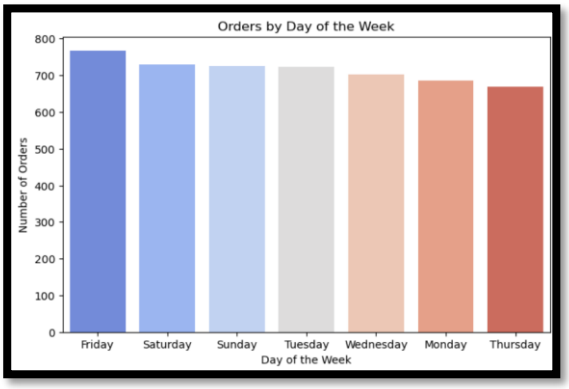
```
sns.lineplot(x='order_month', y='total_amount', data=monthly_revenue, marker='o')
plt.title('Monthly Revenue Trend')
plt.xlabel('Month')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.show()

# Extract the day of the week from order_date
orders['order_day'] = orders['order_date'].dt.day_name()

daily_orders = orders['order_day'].value_counts().reset_index()
daily_orders.columns = ['day', 'order_count']
print(daily_orders)

plt.figure(figsize=(8, 5))
sns.barplot(x='day', y='order_count', data=daily_orders, hue='day', palette='coolwarm')
plt.title('Orders by Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Orders')
plt.legend([],[], frameon=False)  # This removes the legend
plt.show()

#visualise cluster: Recency vs. Total Spend
plt.figure(figsize=(10, 6))
sns.scatterplot(
    x='recency',
    y='total_spend',
    hue='cluster',
    data=customer_metrics,
    palette='plasma'
)
```

```
plt.title('Customer Segmentation: Recency vs. Total Spend')
plt.xlabel('Recency (Days Since Last Order)')
plt.ylabel('Total Spend')
plt.show()
```

Visualization:







B. Discount Impact Analysis

Objective:

Analyze the effect of discounts on customer behavior.

Key Steps:

- Compared **order count and total spend** before and after applying a discount.
- Segmented customers into **Discount Applied (Yes/No)** groups.
- Visualized the difference in **spending behavior**.

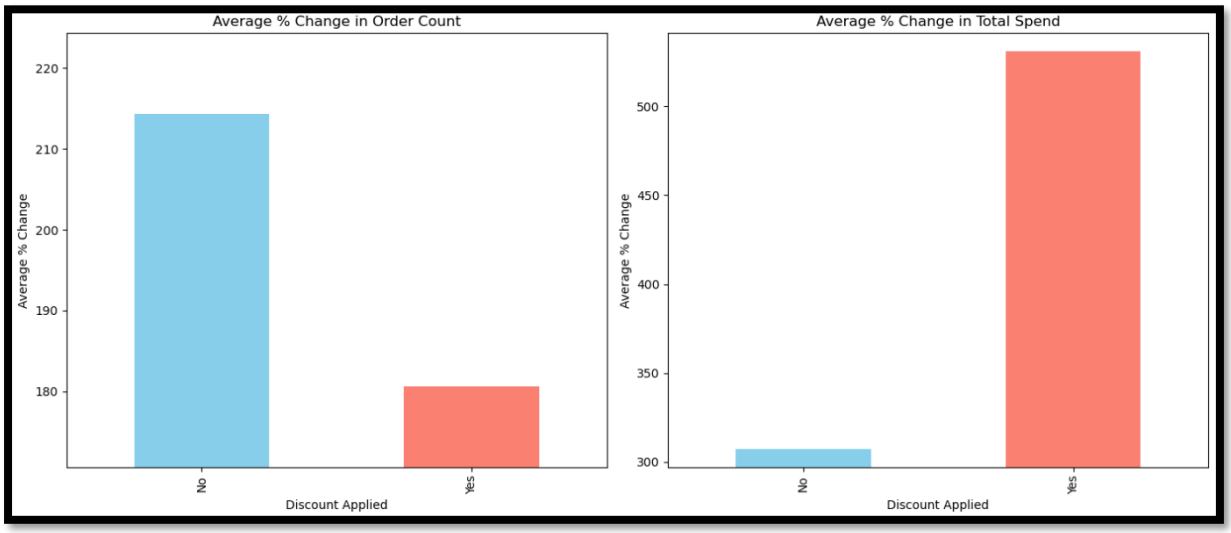
Key Insight:

- Customers **who received discounts** showed **an increase in spending and order frequency**.
- Customers **without discounts** showed **a stable or slight decrease** in spending.

Sample Code:

```
# Set up a figure with two subplots side-by-side

fig, axes = plt.subplots(1, 2, figsize=(14, 6))


# Plot average percentage change in order count

summary['order_count_change_pct'].plot(kind='bar', ax=axes[0], color=['skyblue',
'salmon'])

axes[0].set_title("Average % Change in Order Count")

axes[0].set_xlabel("Discount Applied")

axes[0].set_ylabel("Average % Change")

axes[0].set_ylim([min(summary['order_count_change_pct'])-10,
max(summary['order_count_change_pct'])+10])

# Plot average percentage change in total spend

summary['spend_change_pct'].plot(kind='bar', ax=axes[1], color=['skyblue',
'salmon'])

axes[1].set_title("Average % Change in Total Spend")

axes[1].set_xlabel("Discount Applied")

axes[1].set_ylabel("Average % Change")

axes[1].set_ylim([min(summary['spend_change_pct'])-10,
max(summary['spend_change_pct'])+10])

plt.tight_layout()

plt.show()
```

Visualization:



📈 4. Excel Analysis

A. Inventory Performance Analysis

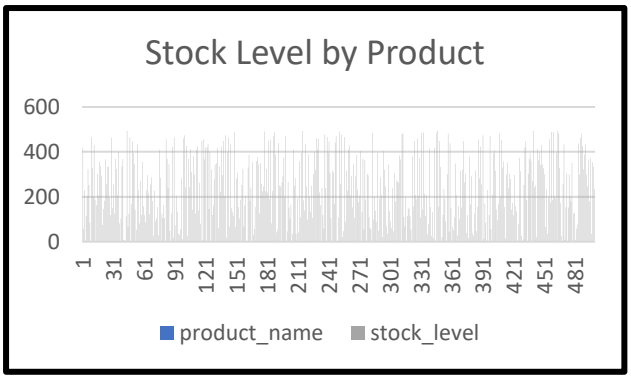
Key Analysis:

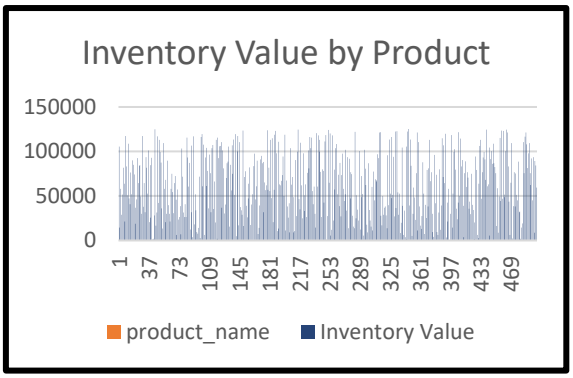
- **Stock Level Overview:** Identified products with low stock.
- **Out-of-Stock Status:** Created a **Pie Chart** showing **Yes/No** status.
- **Inventory Value:** Suggested adding **Unit Price column** to calculate **Stock Level * Unit Price**.

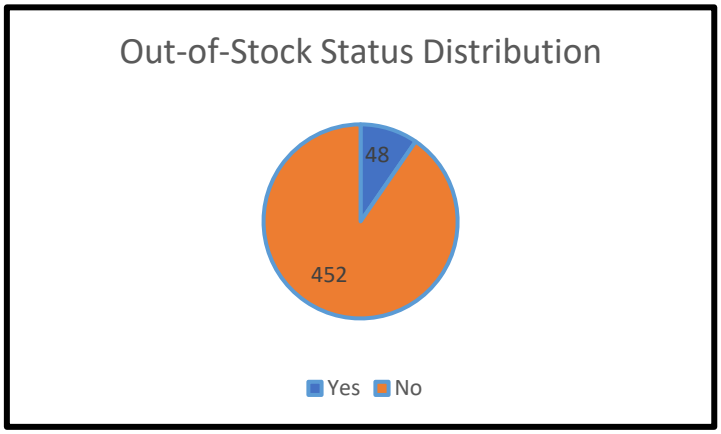
Key Insight:

- **Frequent Out-of-Stock Products:** Need better inventory planning.
- **Low Stock Products:** Require restocking.

Visualizations:







B. Sales Trend Analysis

Key Analysis:

A. **Monthly Revenue Trend:** Line chart showing revenue pattern over time.

B. **Daily Orders:** Bar chart showing order distribution by day.

C. **Regional Sales:** Pivot table analysis of revenue across regions.

Key Insight:

- **Seasonal Trends:** Peak sales in certain months.
- **Best-Performing Region:** Identified based on revenue.

Visualizations:

Table:

Row Labels	Sum of total_revenue
Jan	₹ 1,02,152.91
Feb	₹ 1,09,304.17
Mar	₹ 1,03,387.82
Apr	₹ 1,02,659.68
May	₹ 1,03,633.13
Jun	₹ 1,01,293.64
Jul	₹ 1,14,113.43
Aug	₹ 97,395.46
Sep	₹ 1,07,841.21
Oct	₹ 1,15,082.64
Nov	₹ 1,02,224.75
Dec	₹ 1,03,771.55
Grand Total	**₹ 12,62,860.39**

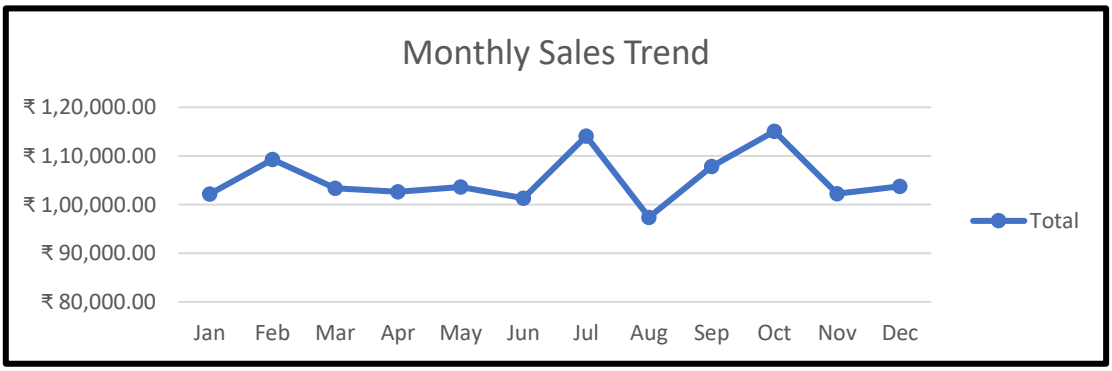


Table:

Row Labels	Sum of total_revenue
Fruits	₹ 1,92,956.76
Chicken	₹ 1,87,973.58
Bread	₹ 1,85,681.73
Rice	₹ 1,82,133.30
Eggs	₹ 1,73,621.45
Vegetables	₹ 1,70,379.17
Milk	₹ 1,70,114.40
Grand Total	**₹ 12,62,860.39**

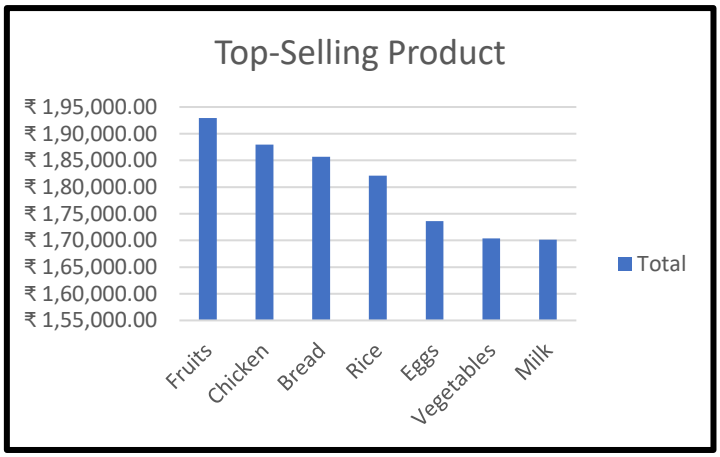
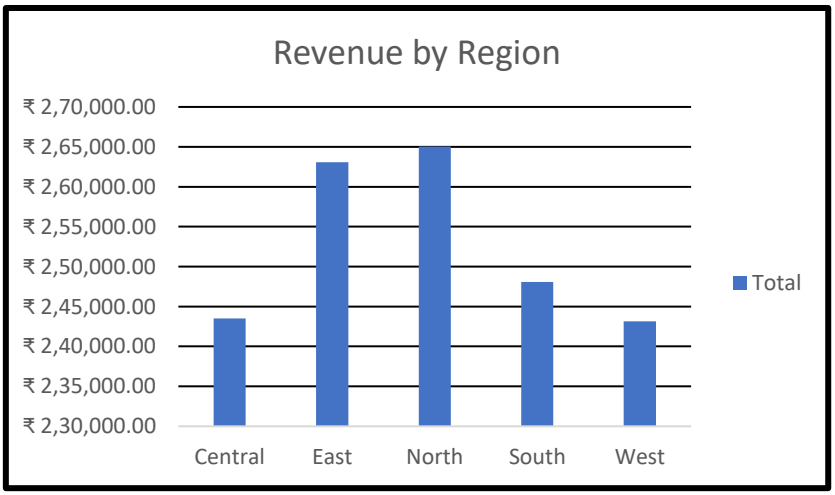


Table:

Row Labels	Sum of total_revenue
Central	₹ 2,43,508.68
East	₹ 2,63,080.64
North	₹ 2,65,056.72
South	₹ 2,48,066.92
West	₹ 2,43,147.43
Grand Total	**₹ 12,62,860.39**



👷 5. SQL Analysis

Objective

The goal of this analysis is to evaluate customer retention, purchasing trends, and delivery performance. Insights from this data can help the business improve customer engagement and delivery operations.

A. Identify Consecutive orders Customers 60 Days ago

Approach:

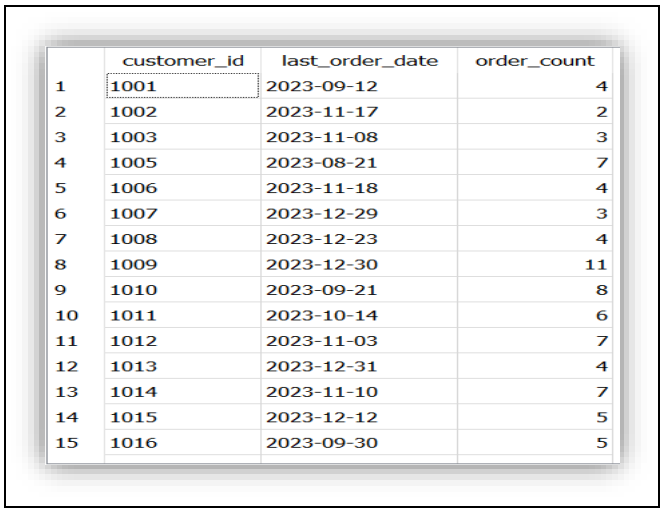
1. Identify customers with at least 2 past orders.

2. Find the last order date for each customer.
3. Filter customers whose last order was more than 60 days ago.

SQL Query: max_orders.sql

```
SELECT field2 AS customer_id,
    MAX(field3) AS last_order_date,
    COUNT(field1) AS order_count
FROM orders1
GROUP BY field2
HAVING MAX(field3) < date('now', '-60 days')
  AND COUNT(field1) >= 2;
```

Output Sample:



	customer_id	last_order_date	order_count
1	1001	2023-09-12	4
2	1002	2023-11-17	2
3	1003	2023-11-08	3
4	1005	2023-08-21	7
5	1006	2023-11-18	4
6	1007	2023-12-29	3
7	1008	2023-12-23	4
8	1009	2023-12-30	11
9	1010	2023-09-21	8
10	1011	2023-10-14	6
11	1012	2023-11-03	7
12	1013	2023-12-31	4
13	1014	2023-11-10	7
14	1015	2023-12-12	5
15	1016	2023-09-30	5

Insights:

- These customers were once active but have not ordered recently.
- Re-engagement campaigns (emails, discounts) can win back their business.

B. Average Time Between Consecutive Orders for Customers

Approach:

1. Focus on customers with more than one order.

2. Calculate the date difference between consecutive orders for each customer.
3. Compute the average time between orders.

SQL Query:

```
WITH order_diffs AS (
  SELECT
    field2 AS customer_id,
    field3 AS order_date,
    julianday(field3) - julianday(LAG(field3) OVER (PARTITION BY field2 ORDER BY field3)) AS diff_days
  FROM orders1
)
SELECT
  customer_id,
  AVG(diff_days) AS avg_days_between_orders
FROM order_diffs
WHERE diff_days IS NOT NULL
GROUP BY customer_id;
```

Output Sample:

	customer_id	avg_days_between_orders
1	1001	45.6666666666667
2	1002	225.0
3	1003	141.5
4	1005	24.5
5	1006	95.6666666666667
6	1007	133.5
7	1008	109.0
8	1009	30.3
9	1010	28.2857142857143
10	1011	49.6
11	1012	35.6666666666667
12	1013	78.3333333333333

Insights:

- Helps forecast future orders and manage stock levels.
- Identifying drops in frequency may signal customer disengagement.

C. Top 10% Customers by Total Spend & Their Average Order Value

Approach:

1. Sum the total spending per customer.
2. Identify the top 10% of customers by spending.
3. Calculate their average order value.

SQL Query:

```
WITH customer_spend AS (
  SELECT
    field2 AS customer_id,
    SUM(field5) AS total_spend,
    COUNT(field1) AS order_count
  FROM orders1
  GROUP BY field2
),
percentile_threshold AS (
  SELECT
    total_spend
  FROM customer_spend
  ORDER BY total_spend DESC
  LIMIT 1 OFFSET (SELECT CAST(0.9 * COUNT(*) AS INT) FROM customer_spend) - 1
)
SELECT
  customer_id,
  total_spend,
  order_count,
  ROUND(total_spend * 1.0 / order_count, 2) AS avg_order_value
FROM customer_spend
WHERE total_spend >= (SELECT total_spend FROM percentile_threshold)
```

Output Sample:

	customer_id	total_spend	order_count	avg_order_value
1	1001	841.381178072244	4	210.35
2	1002	706.672404584228	2	353.34
3	1003	1113.83639712201	3	371.28
4	1005	2100.90423449626	7	300.13
5	1006	887.618689904539	4	221.9
6	1007	648.128972093439	3	216.04
7	1008	1337.30562671332	4	334.33
8	1009	2601.82189018852	11	236.53
9	1010	1474.58943324307	8	184.32
10	1011	1573.47701650857	6	262.25
11	1012	1686.75035859126	7	240.96

Insights:

- High-value customers drive a large portion of revenue.
- Special loyalty programs or personalized discounts can ensure retention.

D. Analyse Delivery Time Efficiency

Approach:

1. Join orders1 and delivery_performance on order_id.
2. Calculate the percentage of on-time deliveries for each city (region).

SQL Query:

```
SELECT
  o.field4 AS region,
  ROUND(
    100.0 * SUM(CASE WHEN TRIM(UPPER(d.field3)) = 'ON TIME' THEN 1 ELSE 0
END)
    / COUNT(*), 2
  ) AS on_time_delivery_percentage
FROM orders1 o
JOIN delivery_performance d ON o.field1 = d.field1
GROUP BY o.field4;
```

Output Sample:

	region	on_time_delivery_percentage
1	Chicago	83.66
2	Houston	84.91
3	Los Angeles	84.3
4	New York	84.58
5	San Francisco	84.54

Insights:

- Some regions consistently meet delivery times; others need improvement.
- Delays can damage customer satisfaction and lead to churn.
- Focus on improving logistics in underperforming regions.

Key Findings Summary:

Task	Key Insight	Business Implication
Inactive Customers	Certain customers have not ordered in 60+ days.	Target them with re-engagement campaigns.
Order Frequency	Repeat customers show ordering patterns.	Use patterns for demand forecasting and personalized offers.
High-Value Customers	Top 10% contribute significantly to revenue.	Prioritize loyalty and retention strategies for them.
Delivery Performance	Delivery efficiency varies by region.	Improve operations in low-performing regions.

Business Implications (Overall):

- Proactively re-engage inactive customers to reduce churn.
- Focus on retaining high-value customers with loyalty programs.
- Leverage order frequency patterns to optimize stock and marketing.
- Address delivery inefficiencies in specific regions to boost customer satisfaction.

✅ **6. Key Recommendations**

- **Customer Segmentation:** Develop **targeted marketing** for **high-value customers**.
- **Discount Campaigns:** Offer **personalized discounts** to **boost spending**.
- **Inventory Management:** Improve **stock forecasting** to **reduce out-of-stock issues**.
- **Sales Strategy:** Focus on **high-performing regions** for **promotional campaigns**.
