

Customer Shopping Behavior Analysis

1. Project Overview

This project analyzes customer shopping behavior using transactional data from 3,900 purchases across various product categories. The goal is to uncover insights into spending patterns, customer segments, product preferences, and subscription behavior to guide strategic business decisions.

2. Dataset Summary

- Rows: 3,900 - Columns: 18 - Key Features:
- Customer demographics (Age, Gender, Location, Subscription Status)
- Purchase details (Item Purchased, Category, Purchase Amount, Season, Size, Color)
- Shopping behavior (Discount Applied, Promo Code Used, Previous Purchases, Frequency of Purchases, Review Rating, Shipping Type)
- Missing Data: 37 values in Review Rating column

3. Exploratory Data Analysis using Python

We began with data preparation and cleaning in Python:

- **Data Loading:** Imported the dataset using pandas.
- **Initial Exploration:** Used `df.info()` to check structure and `.describe()` for summary statistics.

```
# Summary statistics using .describe()
df.describe(include='all')
```

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Discount Applied	Promo Code Used
count	3900.000000	3900.000000	3900	3900	3900	3900.000000	3900	3900	3900	3900	3863.000000	3900	3900	3900	3900
unique	NaN	NaN	2	25	4	NaN	50	4	25	4	NaN	2	6	2	2
top	NaN	NaN	Male	Blouse	Clothing	NaN	Montana	M	Olive	Spring	NaN	No	Free Shipping	No	No
freq	NaN	NaN	2652	171	1737	NaN	96	1755	177	999	NaN	2847	675	2223	2223
mean	1950.500000	44.068462	NaN	NaN	NaN	59.764359	NaN	NaN	NaN	NaN	3.750065	NaN	NaN	NaN	NaN
std	1125.977353	15.207589	NaN	NaN	NaN	23.685392	NaN	NaN	NaN	NaN	0.716983	NaN	NaN	NaN	NaN
min	1.000000	18.000000	NaN	NaN	NaN	20.000000	NaN	NaN	NaN	NaN	2.500000	NaN	NaN	NaN	NaN
25%	975.750000	31.000000	NaN	NaN	NaN	39.000000	NaN	NaN	NaN	NaN	3.100000	NaN	NaN	NaN	NaN
50%	1950.500000	44.000000	NaN	NaN	NaN	60.000000	NaN	NaN	NaN	NaN	3.800000	NaN	NaN	NaN	NaN
75%	2925.250000	57.000000	NaN	NaN	NaN	81.000000	NaN	NaN	NaN	NaN	4.400000	NaN	NaN	NaN	NaN
max	3900.000000	70.000000	NaN	NaN	NaN	100.000000	NaN	NaN	NaN	NaN	5.000000	NaN	NaN	NaN	NaN

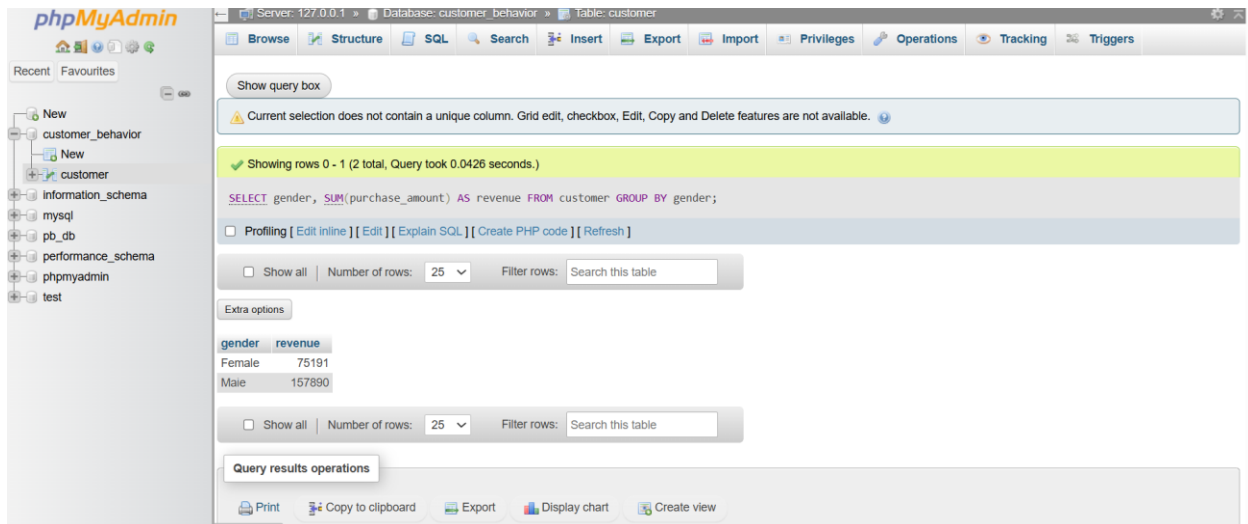
Previous Purchases	Payment Method	Frequency of Purchases
3900.000000	3900	3900
NaN	6	7
NaN	PayPal	Every 3 Months
NaN	677	584
25.351538	NaN	NaN
14.447125	NaN	NaN
1.000000	NaN	NaN
13.000000	NaN	NaN
25.000000	NaN	NaN
38.000000	NaN	NaN
50.000000	NaN	NaN

- **Missing Data Handling:** Checked for null values and imputed missing values in the Review Rating column using the median rating of each product category.
- **Column Standardization:** Renamed columns to **snake case** for better readability and documentation.
- **Feature Engineering:**
 - Created **age_group** column by binning customer ages.
 - Created **purchase_frequency_days** column from purchase data.
- **Data Consistency Check:** Verified if discount_applied and promo_code_used were redundant; dropped promo_code_used.
- **Database Integration:** Connected Python script to PostgreSQL and loaded the cleaned DataFrame into the database for SQL analysis.

4. Data Analysis using SQL (Business Transactions)

We performed structured analysis in PostgreSQL to answer key business questions:

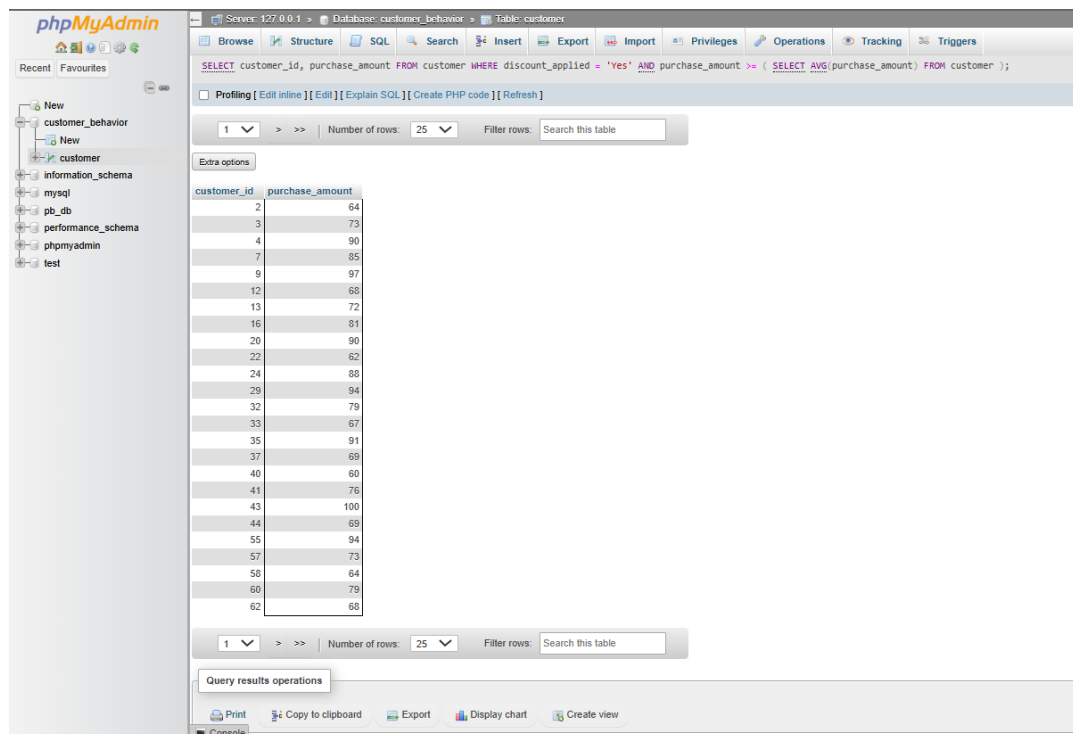
1. **Revenue by Gender** – Compared total revenue generated by male vs. female customers.



The screenshot shows the phpMyAdmin interface with the 'customer' table selected. The SQL query entered is: `SELECT gender, SUM(purchase_amount) AS revenue FROM customer GROUP BY gender;`. The results show two rows: Female with a revenue of 75191 and Male with a revenue of 157890.

gender	revenue
Female	75191
Male	157890

2. **High-Spending Discount Users** – Identified customers who used discounts but still spent above the average purchase amount.



The screenshot shows the phpMyAdmin interface with the 'customer' table selected. The SQL query entered is: `SELECT customer_id, purchase_amount FROM customer WHERE discount_applied = 'Yes' AND purchase_amount >= (SELECT AVG(purchase_amount) FROM customer);`. The results show a list of customer IDs and their purchase amounts, all of which are greater than or equal to the average purchase amount.

customer_id	purchase_amount
2	64
3	73
4	90
7	85
9	97
12	68
13	72
16	81
20	90
22	62
24	88
29	94
32	79
33	67
35	91
37	69
40	60
41	76
43	100
44	69
55	94
57	73
58	64
60	79
62	68

3. Top 5 Products by Rating – Found products with the highest average review ratings.

The screenshot shows the phpMyAdmin interface with the 'customer' table selected. The query results display the top 5 products by average rating. The query used is: `SELECT item_purchased, ROUND(AVG(CAST(review_rating AS DECIMAL(10,2))), 2) AS 'Average Product Rating' FROM customer GROUP BY item_purchased ORDER BY AVG(CAST(review_rating AS DECIMAL(10,2))) DESC LIMIT 5;`

item_purchased	Average Product Rating
Gloves	3.86
Sandals	3.84
Boots	3.82
Hat	3.80
Skirt	3.78

4. Shipping Type Comparison – Compared average purchase amounts between Standard and Express shipping.

The screenshot shows the phpMyAdmin interface with the 'customer' table selected. The query results display the average purchase amounts for Standard and Express shipping types. The query used is: `SELECT shipping_type, ROUND(AVG(purchase_amount), 2) FROM customer WHERE shipping_type IN ('Standard', 'Express') GROUP BY shipping_type;`

shipping_type	ROUND(AVG(purchase_amount), 2)
Express	60.48
Standard	58.46

5. Subscribers vs. Non-Subscribers – Compared average spend and total revenue across subscription status.

The screenshot shows the phpMyAdmin interface with a SQL query executed. The query is:

```
SELECT subscription_status, COUNT(customer_id) AS total_customers, ROUND(AVG(purchase_amount), 2) AS avg_spend, ROUND(SUM(purchase_amount), 2) AS total_revenue FROM customer GROUP BY subscription_status ORDER BY total_revenue, avg_spend DESC;
```

The results table shows the following data:

subscription_status	total_customers	avg_spend	total_revenue
Yes	1053	59.49	62645.00
No	2847	59.87	170436.00

6. Discount-Dependent Products – Identified 5 products with the highest percentage of discounted purchases.

The screenshot shows the phpMyAdmin interface with a SQL query executed. The query is:

```
-- Q6. Which 5 products have the highest percentage of purchases with discounts applied? SELECT item_purchased, ROUND(SUM(CASE WHEN discount_applied = 'Yes' THEN 1 ELSE 0 END) / CAST(COUNT(*) AS DECIMAL(10,2)), 2) AS discount_rate FROM customer GROUP BY item_purchased ORDER BY discount_rate DESC LIMIT 5;
```

The results table shows the following data:

item_purchased	discount_rate
Hat	50.00
Sneakers	49.06
Coat	49.07
Sweater	48.17
Pants	47.37

7. Customer Segmentation – Classified customers into New, Returning, and Loyal segments based on purchase history.

The screenshot shows the phpMyAdmin interface with a SQL query executed. The query is:

```
WITH customer_type AS ( SELECT customer_id, previous_purchases, CASE WHEN previous_purchases = 1 THEN 'New' WHEN previous_purchases BETWEEN 2 AND 34 THEN 'Returning' ELSE 'Loyal' END AS customer_segment FROM customer ) SELECT customer_segment, COUNT(*) AS "Number of Customers" FROM customer_type GROUP BY customer_segment;
```

The results table shows the following data:

customer_segment	Number of Customers
Loyal	3116
New	83
Returning	791

8. Top 3 Products per Category – Listed the most purchased products within each category.

The screenshot shows the phpMyAdmin interface with a query result for the 'customer' table. The query is: `SELECT category, item_rank, category, item_purchased, total_orders FROM customer ORDER BY category, item_rank DESC`. The result shows 10 rows, with the top 3 products per category highlighted in green.

item_rank	category	item_purchased	total_orders
1	Accessories	Bag	161
1	Accessories	Jewelry	171
2	Accessories	Sunglasses	161
1	Clothing	Blouse	171
2	Clothing	Pants	171
3	Clothing	Shirt	169
1	Footwear	Sandals	169
2	Footwear	Shoes	168
3	Footwear	Sneakers	145
2	Outerwear	Coat	161
1	Outerwear	Jacket	163

9. Repeat Buyers & Subscriptions – Checked whether customers with >5 purchases are more likely to subscribe.

The screenshot shows the phpMyAdmin interface with a query result for the 'customer' table. The query is: `SELECT subscription_status, COUNT(customer_id) AS repeat_buyers FROM customer WHERE previous_purchases > 5 GROUP BY subscription_status`. The result shows 2 rows: 'No' with 2518 repeat buyers and 'Yes' with 956 repeat buyers.

subscription_status	repeat_buyers
No	2518
Yes	956

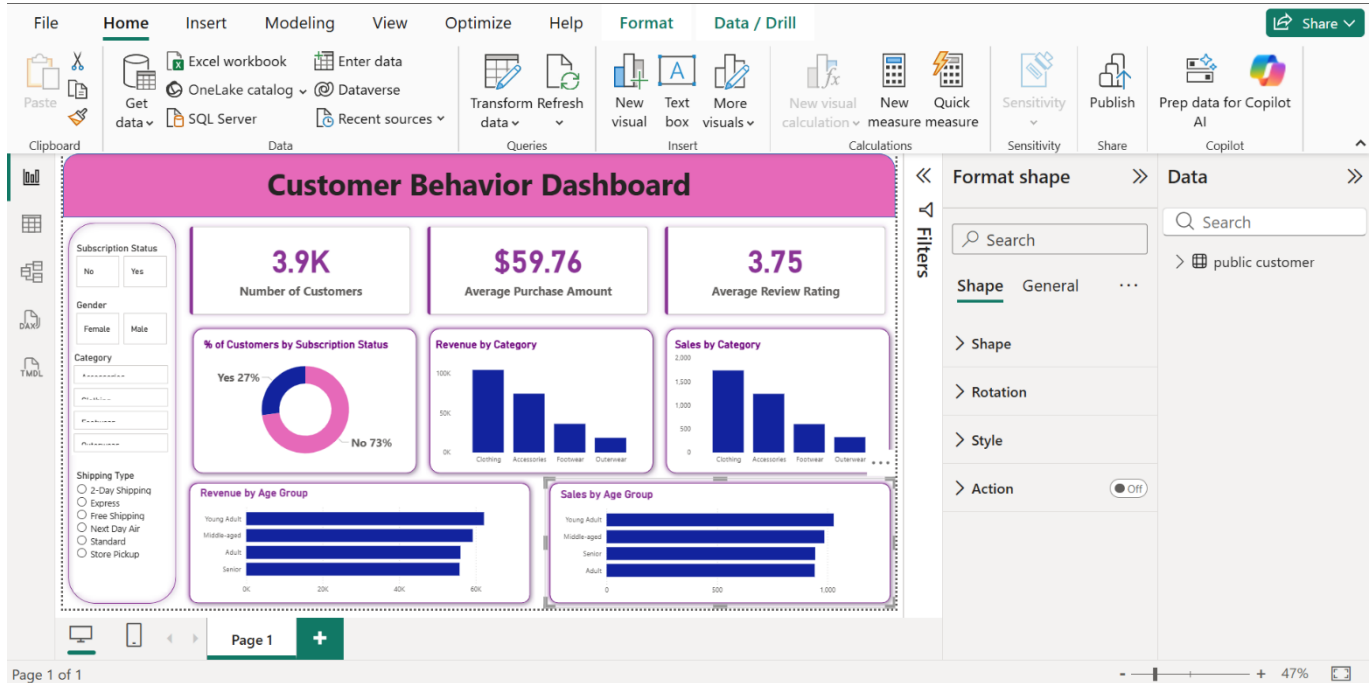
10. Revenue by Age Group – Calculated total revenue contribution of each age group.

The screenshot shows the phpMyAdmin interface with a query result for the 'customer' table. The query is: `SELECT age_group, SUM(purchase_amount) AS total_revenue FROM customer GROUP BY age_group ORDER BY total_revenue DESC`. The result shows 4 rows: 'Young Adult' with 62143 total revenue, 'Middle-aged' with 59197 total revenue, 'Adult' with 55978 total revenue, and 'Senior' with 55763 total revenue.

age_group	total_revenue
Young Adult	62143
Middle-aged	59197
Adult	55978
Senior	55763

5. Dashboard in Power BI

Finally, we built an interactive dashboard in **Power BI** to present insights visually.



6. Business Recommendations

- **Boost Subscriptions** – Promote exclusive benefits for subscribers.
- **Customer Loyalty Programs** – Reward repeat buyers to move them into the “Loyal” segment.
- **Review Discount Policy** – Balance sales boosts with margin control.
- **Product Positioning** – Highlight top-rated and best-selling products in campaigns.
- **Targeted Marketing** – Focus efforts on high-revenue age groups and express-shipping users.