Customer Shopping Behavior Analysis Report

By Mohammad Jawad Nayosh
jnayosh@gmail.com

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.

[1]:
```python
import pandas as pd
df = pd.read_csv('customer_shopping_behavior.csv')
```

[3]:
```python
df.head()
```

[3]:

	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	Previous Purchases	Payment Method	Frequency of Purchases
0	1	55	Male	Blouse	Clothing	53	Kentucky	L	Gray	Winter	3.1	Yes	Express	Yes	Yes	14	Venmo	Fortnightly
1	2	19	Male	Sweater	Clothing	64	Maine	L	Maroon	Winter	3.1	Yes	Express	Yes	Yes	2	Cash	Fortnightly
2	3	50	Male	Jeans	Clothing	73	Massachusetts	S	Maroon	Spring	3.1	Yes	Free Shipping	Yes	Yes	23	Credit Card	Weekly
3	4	21	Male	Sandals	Footwear	90	Rhode Island	M	Maroon	Spring	3.5	Yes	Next Day Air	Yes	Yes	49	PayPal	Weekly
4	5	45	Male	Blouse	Clothing	49	Oregon	M	Turquoise	Spring	2.7	Yes	Free Shipping	Yes	Yes	31	PayPal	Annually

- **Missing Data Handling:** Checked for null values and imputed missing values in the `Review Rating` column using the median rating of each product category.

```
[9]: df.isnull().sum()
```

```
[9]: Customer ID              0
     Age                      0
     Gender                   0
     Item Purchased           0
     Category                 0
     Purchase Amount (USD)    0
     Location                 0
     Size                     0
     Color                    0
     Season                   0
     Review Rating           37
     Subscription Status      0
     Shipping Type            0
     Discount Applied         0
     Promo Code Used          0
     Previous Purchases       0
     Payment Method           0
     Frequency of Purchases   0
     dtype: int64
```

```
[13]: df['Review Rating'] = df.groupby('Category')['Review Rating'].transform(lambda x: x.fillna(x.median()))
```

```
[15]: df.isnull().sum()
```

```
[15]: Customer ID              0
      Age                      0
      Gender                   0
      Item Purchased           0
      Category                 0
      Purchase Amount (USD)    0
      Location                 0
      Size                     0
      Color                    0
      Season                   0
      Review Rating            0
      Subscription Status      0
      Shipping Type            0
      Discount Applied         0
      Promo Code Used          0
      Previous Purchases       0
      Payment Method           0
      Frequency of Purchases   0
      dtype: int64
```

- **Column Standardization:** Renamed columns to **snake case** for better readability and documentation.

```
[23]: df.columns = df.columns.str.lower()
      df.columns = df.columns.str.replace(' ','_')
```

```
[25]: df.columns
```

```
[25]: Index(['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', 'previous_purchases',
             'payment_method', 'frequency_of_purchases'],
            dtype='object')
```

```
[27]: df = df.rename(columns= {'purchase_amount_(usd)':'purchase_amount'})
```

```
[29]: df.columns
```

```
[29]: Index(['customer_id', 'age', 'gender', 'item_purchased', 'category',
             'purchase_amount', 'location', 'size', 'color', 'season',
             'review_rating', 'subscription_status', 'shipping_type',
             'discount_applied', 'promo_code_used', 'previous_purchases',
             'payment_method', 'frequency_of_purchases'],
            dtype='object')
```

- **Feature Engineering:**
    - Created **age_group** column by binning customer ages.

```
dtype='object')
```

```
[35]: # Create a column age_groupabs
      labels = ['Young Adult', 'Adult','Middle-Age','Senior']
      df['age_group'] = pd.qcut(df['age'], q=4, labels=labels)
```

```
[47]: df.head(5)
```

[47]:

ɔscription_status	shipping_type	discount_applied	promo_code_used	previous_purchases	payment_method	frequency_of_purchases	age_group	purchase_frequency_days
Yes	Express	Yes	Yes	14	Venmo	Fortnightly	Middle-Age	14.0
Yes	Express	Yes	Yes	2	Cash	Fortnightly	Young Adult	14.0
Yes	Free Shipping	Yes	Yes	23	Credit Card	Weekly	Middle-Age	7.0
Yes	Next Day Air	Yes	Yes	49	PayPal	Weekly	Young Adult	7.0
Yes	Free Shipping	Yes	Yes	31	PayPal	Annually	Middle-Age	NaN

- Created **purchase_frequency_days** column from purchase data.

[51]:
```python
# Create column purchase_frequency_days
frequency_mapping = {
    'Weekly':7, 'Fortnightly':14,'Bi-Weekly':14, 'Monthly':30, 'Quarterly':90,'Every 3 Month': 90,'Annually':365
}
df['purchase_frequency_days'] = df['frequency_of_purchases'].map(frequency_mapping)
```

[53]:
```python
df.head()
```

[53]:

scription_status	shipping_type	discount_applied	promo_code_used	previous_purchases	payment_method	frequency_of_purchases	age_group	purchase_frequency_days
Yes	Express	Yes	Yes	14	Venmo	Fortnightly	Middle-Age	14.0
Yes	Express	Yes	Yes	2	Cash	Fortnightly	Young Adult	14.0
Yes	Free Shipping	Yes	Yes	23	Credit Card	Weekly	Middle-Age	7.0
Yes	Next Day Air	Yes	Yes	49	PayPal	Weekly	Young Adult	7.0
Yes	Free Shipping	Yes	Yes	31	PayPal	Annually	Middle-Age	365.0

- **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)

```sql
create database customer_shopping_behavior;
use customer_shopping_behavior;
select* from customer_shopping_behavior_clean;
```

I performed structured analysis in SQL Server to answer key business questions:

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

```sql
-- Q1. What is the total revenue generated by male customers compared to female customers?

select gender, sum(purchase_amount) as revenue
from customer_shopping_behavior_clean
group by gender;
```

100 %

Results | Messages

	gender	revenue
1	Male	157890
2	Female	75191

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

```sql
-- Q2. Which customers used a discount but still spent more than the overall average purchase amount?
select customer_id, purchase_amount
from customer_shopping_behavior_clean
where discount_applied = 'Yes' and purchase_amount >= (select AVG(purchase_amount) from customer_shopping_behavior_clean);
select* from customer_shopping_behavior_clean;
```

100 %

Results | Messages

	customer_id	purchase_amount
1	2	64
2	3	73
3	4	90
4	7	85
5	9	97
6	12	68
7	13	72
8	16	81
9	20	90
10	22	62
11	24	88
12	29	94
13	32	79

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

```sql
-- Q3. Which are the top 5 products with the highest average review rating?
select top 5
    item_purchased,
    round(avg(review_rating), 2) as average_product_rating
from customer_shopping_behavior_clean
group by item_purchased
order by avg(review_rating) desc;
```

100 %

Results | Messages

	item_purchased	average_product_rating
1	Gloves	3.86
2	Sandals	3.84
3	Boots	3.82
4	Hat	3.8
5	Skirt	3.78

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

```sql
-- Q4. Compare the average Purchase Amounts between Standard Shipping and Express Shipping.
select
    shipping_type,
    round(avg(purchase_amount),2) as average_purchase_amount
from customer_shopping_behavior_clean
group by shipping_type;

--Or only standard vs express shipping
select shipping_type, round(AVG(purchase_amount),2)
from customer_shopping_behavior_clean
where shipping_type in ('Express', 'Standard')
group by shipping_type;
```

Results | Messages

shipping_type	average_purchase_amount
Next Day Air	58
Store Pickup	59
Free Shipping	60
Express	60
Standard	58
2-Day Shipping	60

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

```sql
-- Q5. Do subscribed customers spend more? Compare both the average spend
-- and the total revenue between subscribers and non-subscribers.
select subscription_status, COUNT(customer_id)as total_customers,
round(AVG(purchase_amount),2)as avg_spent, sum(purchase_amount) as total_revenue
from customer_shopping_behavior_clean
group by subscription_status;
select* from customer_shopping_behavior_clean;
```

Results | Messages

	subscription_status	total_customers	avg_spent	total_revenue
1	Yes	1053	59	62645
2	No	2847	59	170436

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

```sql
-- Q6. Which 5 products have the highest percentage of purchases with discounts applied?
SELECT TOP 5
    item_purchased,
    ROUND(
        (SUM(CASE WHEN discount_applied = 'Yes' THEN 1 ELSE 0 END) * 100.0)
        / COUNT(*), 2
    ) AS discount_percentage
FROM customer_shopping_behavior_clean
GROUP BY item_purchased
ORDER BY discount_percentage DESC;
```

100 %

Results | Messages

	item_purchased	discount_percentage
1	Hat	50.000000000000
2	Sneakers	49.660000000000
3	Coat	49.070000000000
4	Sweater	48.170000000000
5	Pants	47.370000000000

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

```sql
-- Q7. Segment customers into New, Returning, and Loyal based on their total number of previous purchases,
-- and show the count of customers in each segment.

WITH customer_type AS (
    SELECT
        customer_id,
        previous_purchases,
        CASE
            WHEN previous_purchases = 1 THEN 'New'
            WHEN previous_purchases BETWEEN 2 AND 10 THEN 'Returning'
            ELSE 'Loyal'
        END AS customer_segment
    FROM customer_shopping_behavior_clean
)

SELECT
    customer_segment,
    COUNT(*) AS Number_of_Customers
FROM customer_type
GROUP BY customer_segment;
```

100 %

Results | Messages

	customer_segment	Number_of_Customers
1	Returning	701
2	Loyal	3116
3	New	83

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

```sql
-- Q8. What are the top 3 most purchased products within each product category?

WITH item_counts AS (
    SELECT
        category,
        item_purchased,
        COUNT(customer_id) AS total_orders,
        ROW_NUMBER() OVER (
            PARTITION BY category
            ORDER BY COUNT(customer_id) DESC
        ) AS item_rank
    FROM customer_shopping_behavior_clean
    GROUP BY category, item_purchased
)

SELECT
    item_rank,
    category,
    item_purchased,
    total_orders
FROM item_counts
WHERE item_rank <= 3
ORDER BY item_rank,category;
select* from customer_shopping_behavior_clean;
```

	item_rank	category	item_purchased	total_orders
1	1	Accessories	Jewelry	171
2	1	Clothing	Blouse	171
3	1	Footwear	Sandals	160
4	1	Outerwear	Jacket	163
5	2	Accessories	Sunglasses	161
6	2	Clothing	Pants	171
7	2	Footwear	Shoes	150
8	2	Outerwear	Coat	161
9	3	Accessories	Belt	161
10	3	Clothing	Shirt	169
11	3	Footwear	Sneakers	145

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

```sql
-- Q9. Are repeat buyers (more than 5 previous purchases) more likely to be subscribers?
select subscription_status,
count (customer_id) as repeat_buyers
from customer_shopping_behavior_clean
where previous_purchases > 5
group by subscription_status;
```

100 %

Results | Messages

	subscription_status	repeat_buyers
1	Yes	958
2	No	2518

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

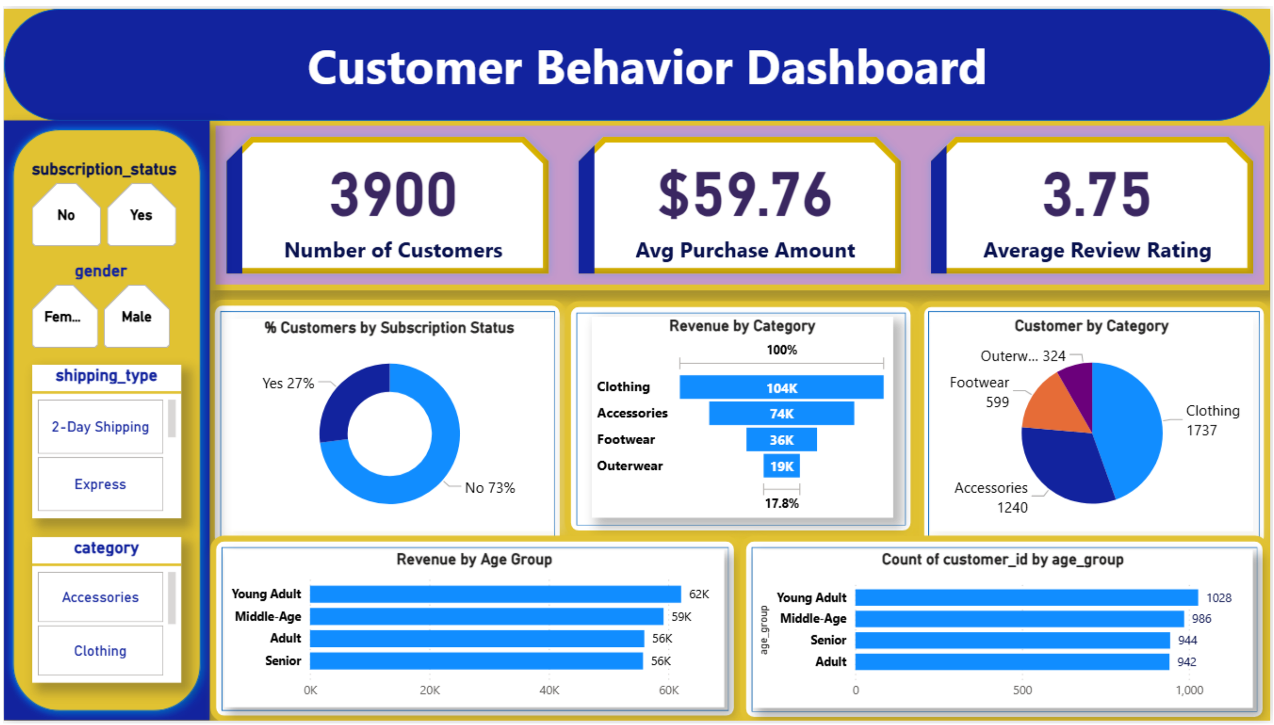
```sql
-- Q10. What is the revenue contribution (%) of each age group?

select age_group, SUM(purchase_amount) as total_revenue
from customer_shopping_behavior_clean
group by age_group
order by total_revenue desc;
```

	age_group	total_revenue
1	Young Adult	62143
2	Middle-Age	59197
3	Adult	55978
4	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.