2/ Customer Analysis

Customer Analysis:

Who are the top customers by revenue and profit?
What is the average order value and frequency for different customer segments?
How does customer lifetime value vary across different segments?
What factors drive higher customer lifetime value?

How do discounts and promotions impact customer purchasing behavior?

```
# Merge the order dataset with merged data to include revenue and
profit information
customer_order_data =
pd.merge(order_dateset[['order_id','customer_id','order_status','order
_purchase_timestamp','order_estimated_delivery_date']], merged_data,
on='order_id')

# Merge with customer dataset to include customer details
customer_data = pd.merge(customer_order_data, customers_dataset,
on='customer_id')

# Aggregate revenue and profit by customer_id
customer_performance = customer_data.groupby('customer_id').agg({
    'total_revenue': 'sum',
    'profit': 'sum'
}).reset_index()

# Sort the customers by revenue and profit
top_customers_by_revenue =
customer_performance.sort_values(by='total_revenue', ascending=False)
top_customers_by_profit =
customer_performance.sort_values(by='profit', ascending=False)

# Define the number of top customers to display
top_n = 10

# Top customers by revenue
top_n_customers_revenue = top_customers_by_revenue.head(top_n)

# Top customers by profit
top_n_customers_profit = top_customers_by_profit.head(top_n)


print("Top Customers by Revenue:")
top_n_customers_revenue
```

```
Top Customers by Revenue:
```

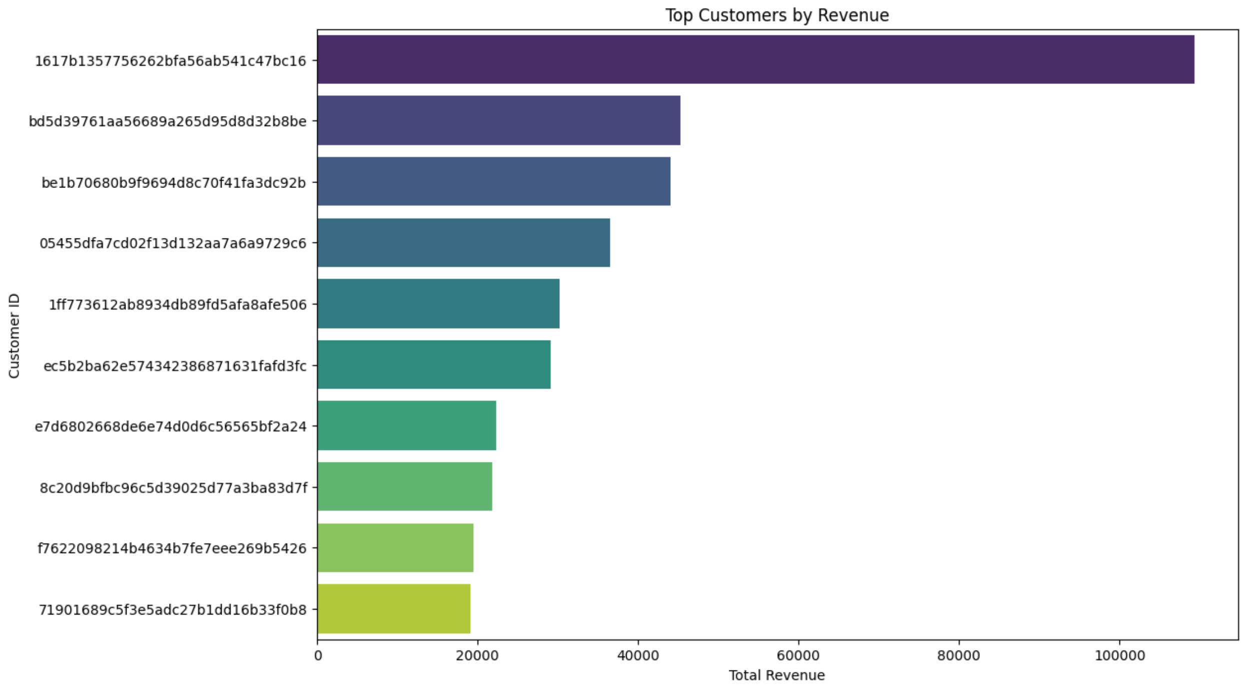
```
{"summary":"{\n  \"name\": \"top_n_customers_revenue\",\n  \"rows\":
10,\n  \"fields\": [\n    {\n      \"column\": \"customer_id\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n
\"f7622098214b4634b7fe7eee269b5426\",\n
\"bd5d39761aa56689a265d95d8d32b8be\",\n
\"ec5b2ba62e574342386871631fafd3fc\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"total_revenue\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
26915.176806903757,\n        \"min\": 19174.38,\n        \"max\":
109312.64,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          19457.04,\n          45256.0,\n          29099.52\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"profit\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
23965.988615715883,\n        \"min\": 16074.630000000001,\n
\"max\": 95872.64,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          16537.64,\n          43282.00000000001,\n
21939.52\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"top_n_customers_revenue"}
```

```python
print("\nTop Customers by Profit:")
top_n_customers_profit
```

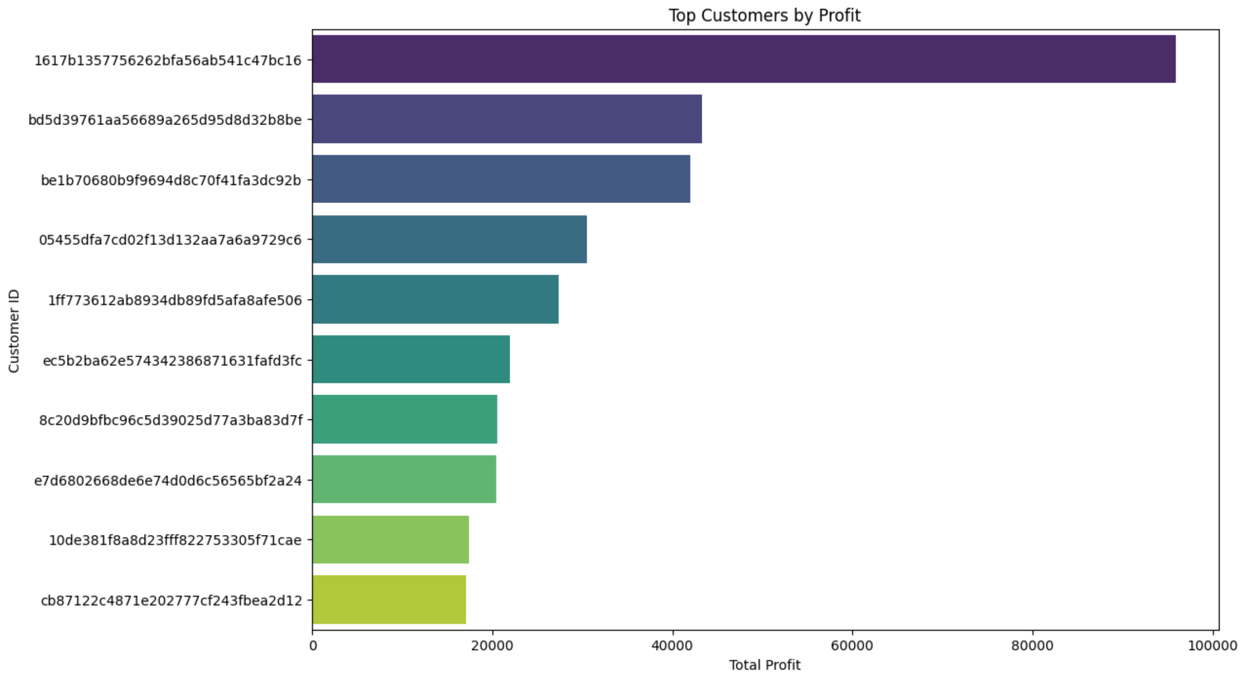
```
Top Customers by Profit:
```

```
{"summary":"{\n  \"name\": \"top_n_customers_profit\",\n  \"rows\":
10,\n  \"fields\": [\n    {\n      \"column\": \"customer_id\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 10,\n        \"samples\": [\n
\"10de381f8a8d23fff822753305f71cae\",\n
\"bd5d39761aa56689a265d95d8d32b8be\",\n
\"ec5b2ba62e574342386871631fafd3fc\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"total_revenue\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
27037.007258074584,\n        \"min\": 18384.75,\n        \"max\":
109312.64,\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          18384.75,\n          45256.0,\n          29099.52\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"profit\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
23814.697224421452,\n        \"min\": 17147.54,\n        \"max\":
95872.64,\n        \"num_unique_values\": 10,\n        \"samples\": [\
n          17402.4,\n          43282.00000000001,\n          21939.52\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"top_n_customers_profit"}
```

```python
plt.figure(figsize=(12, 8))
sns.barplot(x='total_revenue', y='customer_id',
data=top_n_customers_revenue, hue ='customer_id', palette='viridis')
plt.title('Top Customers by Revenue')
plt.xlabel('Total Revenue')
plt.ylabel('Customer ID')
plt.show()
```



```
plt.figure(figsize=(12, 8))
sns.barplot(x='profit', y='customer_id',
data=top_n_customers_profit,hue = 'customer_id', palette='viridis')
plt.title('Top Customers by Profit')
plt.xlabel('Total Profit')
plt.ylabel('Customer ID')
plt.show()
```



Who are the top customers by revenue and profit?

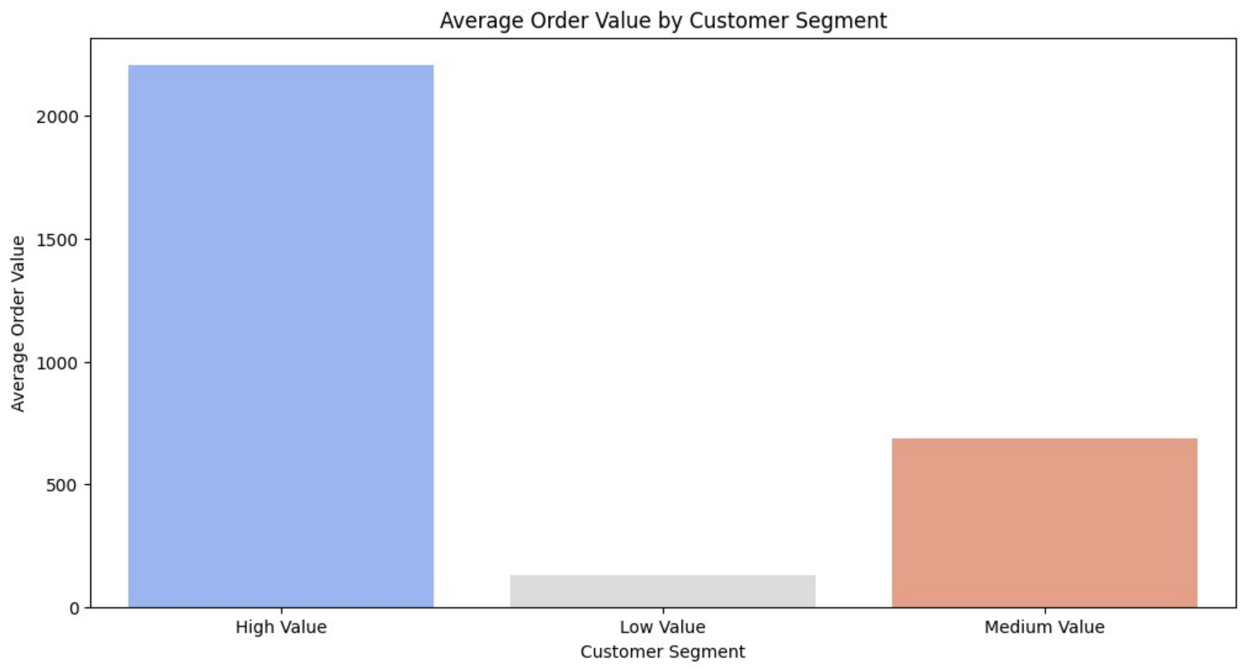
```
# top customer by revenue and profit is

top_n_customers_profit.iloc[0]
```

```
customer_id      1617b1357756262bfa56ab541c47bc16
total_revenue                           109312.64
profit                                   95872.64
Name: 8475, dtype: object
```

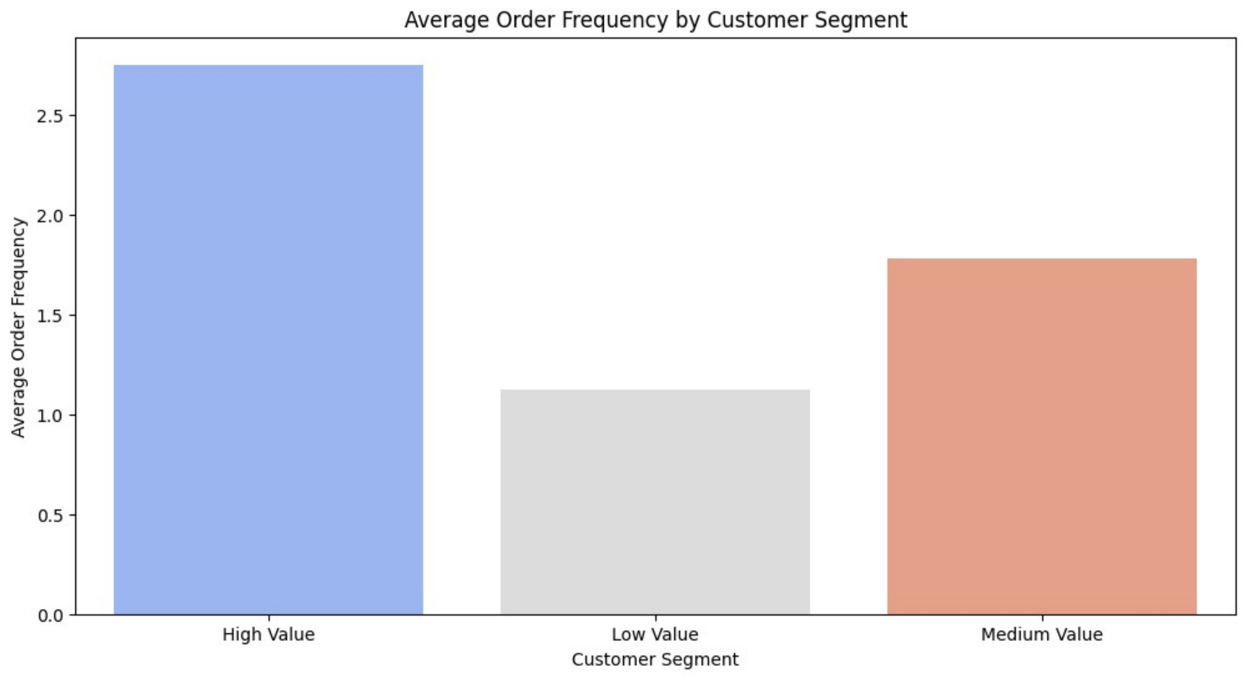
What is the average order value and frequency for different customer segments?

```
# Assuming merged_data contains the relevant data
# Calculate total revenue and frequency for each customer
customer_summary = customer_data.groupby('customer_id').agg({
    'total_revenue': 'sum',
    'order_id': 'count'
}).reset_index()

# Rename columns for clarity
customer_summary.rename(columns={'total_revenue': 'total_spending',
'order_id': 'order_frequency'}, inplace=True)

# Define segments based on total spending and order frequency
def segment_customer(row):
    if row['total_spending'] > 1000:
        return 'High Value'
    elif row['total_spending'] > 500:
        return 'Medium Value'
    else:
        return 'Low Value'

customer_summary['segment'] = customer_summary.apply(segment_customer,
axis=1)

# Calculate average order value (total spending divided by order
frequency) for each segment
segment_summary = customer_summary.groupby('segment').agg({
    'total_spending': 'mean',
    'order_frequency': 'mean'
}).reset_index()

# Rename columns for clarity
segment_summary.rename(columns={'total_spending':
'average_order_value', 'order_frequency': 'average_order_frequency'},
inplace=True)
```

```python
# Plot average order value by segment
plt.figure(figsize=(12, 6))
sns.barplot(x='segment', y='average_order_value',
data=segment_summary,hue = 'segment', palette='coolwarm')
plt.title('Average Order Value by Customer Segment')
plt.xlabel('Customer Segment')
plt.ylabel('Average Order Value')
plt.show()
```



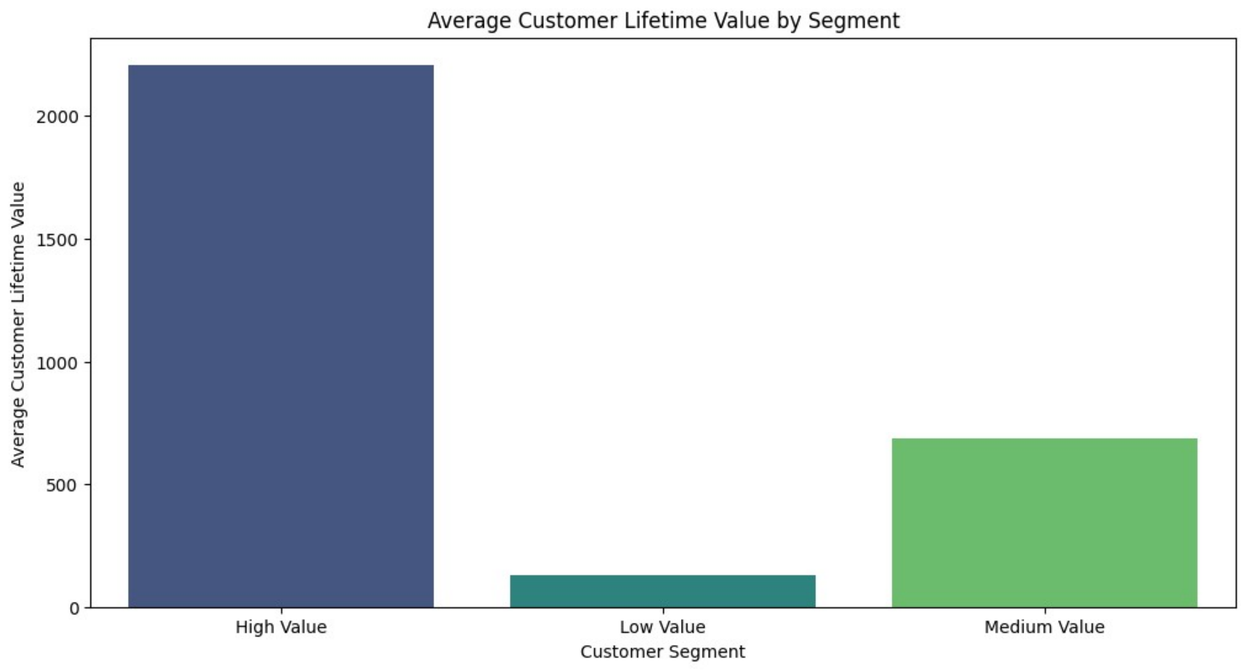
```python
# Plot average order frequency by segment
plt.figure(figsize=(12, 6))
sns.barplot(x='segment', y='average_order_frequency',
data=segment_summary,hue = 'segment', palette='coolwarm')
plt.title('Average Order Frequency by Customer Segment')
plt.xlabel('Customer Segment')
plt.ylabel('Average Order Frequency')
plt.show()
```



How does customer lifetime value vary across different segments?

```
# Calculate total revenue per customer
customer_clv = customer_data.groupby('customer_id').agg({
    'total_revenue': 'sum'
}).reset_index()

# Rename columns for clarity
customer_clv.rename(columns={'total_revenue': 'CLV'}, inplace=True)

# Define segments based on CLV
def segment_customer_by_clv(row):
    if row['CLV'] > 1000:
        return 'High Value'
    elif row['CLV'] > 500:
        return 'Medium Value'
    else:
        return 'Low Value'

customer_clv['segment'] = customer_clv.apply(segment_customer_by_clv,
axis=1)

# Calculate average CLV for each segment
segment_clv_summary = customer_clv.groupby('segment').agg({
    'CLV': 'mean'
}).reset_index()

# Rename columns for clarity
segment_clv_summary.rename(columns={'CLV': 'average_CLV'},
inplace=True)
```

```python
# Plot average CLV by segment
plt.figure(figsize=(12, 6))
sns.barplot(x='segment', y='average_CLV', data=segment_clv_summary,hue
= 'segment', palette='viridis')
plt.title('Average Customer Lifetime Value by Segment')
plt.xlabel('Customer Segment')
plt.ylabel('Average Customer Lifetime Value')
plt.show()
```



What factors drive higher customer lifetime value?

What factors drive higher customer lifetime value?

Higher Customer Lifetime Value (CLV) is influenced by several key factors. Understanding these factors can help businesses improve their strategies for acquiring and retaining high-value customers. Here are some of the main factors that drive higher CLV:

1. Frequency of Purchase Description: Customers who make purchases more frequently have higher CLV. Increasing the frequency of customer transactions can significantly boost CLV. Strategies: Implement loyalty programs, offer subscription services, or send personalized promotions to encourage repeat purchases.
2. Average Order Value (AOV) Description: Higher average order values contribute directly to higher CLV. If customers spend more per transaction, their lifetime value increases. Strategies: Use upselling and cross-selling techniques, offer bundle deals, or provide incentives for higher-value purchases.
3. Customer Retention Rate Description: The longer a customer stays with a brand, the higher their CLV. Retaining customers over time ensures more opportunities for them to spend. Strategies: Focus on customer satisfaction, provide excellent customer service, and regularly engage with customers through email and social media.

4. Customer Acquisition Cost (CAC) Description: Lower CAC means that the cost of acquiring new customers is less relative to their lifetime value, improving overall profitability. Strategies: Optimize marketing campaigns, target high-value customer segments, and improve conversion rates to reduce CAC.
5. Product/Service Quality Description: High-quality products or services lead to better customer experiences and higher CLV. Satisfied customers are more likely to return and make repeat purchases. Strategies: Continuously improve product quality, gather and act on customer feedback, and maintain high standards.
6. Customer Experience Description: A positive overall customer experience encourages repeat business and enhances CLV. This includes every interaction a customer has with the brand. Strategies: Ensure a seamless shopping experience, provide excellent support, and personalize interactions to meet individual customer needs.
7. Engagement and Personalization Description: Engaged customers who feel valued are more likely to make repeat purchases. Personalizing offers and communications can drive higher CLV. Strategies: Use data analytics to tailor marketing efforts, offer personalized recommendations, and create targeted promotions.
8. Loyalty Programs Description: Loyalty programs can increase CLV by incentivizing repeat purchases and rewarding long-term customers. Strategies: Implement a rewards program that offers points, discounts, or exclusive perks for repeat business.
9. Seasonal and Behavioral Trends Description: Understanding seasonal buying patterns and customer behavior can help tailor strategies to maximize CLV. Strategies: Analyze historical data to identify trends and plan marketing efforts around key buying periods or behaviors.
10. Customer Segmentation Description: Segmenting customers based on their purchasing behavior and preferences can help tailor strategies to increase CLV. Strategies: Use segmentation to target high-value customers with specific offers and tailor your marketing approach to different customer groups. Example Analysis

How do discounts and promotions impact customer purchasing behavior?

```
customer_data.rename(columns={'payment_installments': 'discounts'},
inplace=True)

# Group by discount presence
discount_comparison = customer_data.groupby('discounts').agg({
    'total_revenue': 'sum',
    'order_id': 'count'
}).rename(columns={'order_id': 'order_count'}).reset_index()

discount_comparison
```

```
{"summary":"{\n  \"name\": \"discount_comparison\",\n  \"rows\": 24,\n
\"fields\": [\n    {\n      \"column\": \"discounts\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7,\n        \"min\": 0,\n        \"max\": 24,\n
\"num_unique_values\": 24,\n        \"samples\": [\n          8,\n
16,\n          0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"total_revenue\",\n      \"properties\": {\n        \"dtype\":
```

```
\"number\",\n        \"std\": 1689336.0247286377,\n        \"min\":
228.71,\n        \"max\": 7756342.37,\n        \"num_unique_values\":
24,\n        \"samples\": [\n          1572881.7,\n
2765.5699999999997,\n          318.57\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"order_count\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
12129,\n        \"min\": 1,\n        \"max\": 58617,\n
\"num_unique_values\": 22,\n        \"samples\": [\n          3,\n
18,\n          5063\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"discount_comparison"}
```

```
#  Calculate average order value
avg_order_value = customer_data.groupby('discounts')
['total_revenue'].mean().reset_index()
avg_order_value.columns = ['Discounts', 'Average Order Value']
avg_order_value
```

```
{"summary":"{\n  \"name\": \"avg_order_value\",\n  \"rows\": 24,\n
\"fields\": [\n    {\n      \"column\": \"Discounts\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7,\n        \"min\": 0,\n        \"max\": 24,\n
\"num_unique_values\": 24,\n        \"samples\": [\n          8,\n
16,\n          0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Average Order Value\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 168.278856982075,\n        \"min\":
106.19,\n        \"max\": 801.7379411764706,\n
\"num_unique_values\": 24,\n        \"samples\": [\n
310.66199881493185,\n          395.08142857142855,\n          106.19\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\
n}","type":"dataframe","variable_name":"avg_order_value"}
```

```
# Calculate order frequency with and without discounts
order_frequency = customer_data.groupby('discounts').agg({
    'order_id': 'count',
    'customer_id': 'nunique'
}).rename(columns={'order_id': 'total_orders', 'customer_id':
'total_customers'}).reset_index()

order_frequency['orders_per_customer'] =
order_frequency['total_orders'] / order_frequency['total_customers']
order_frequency
```

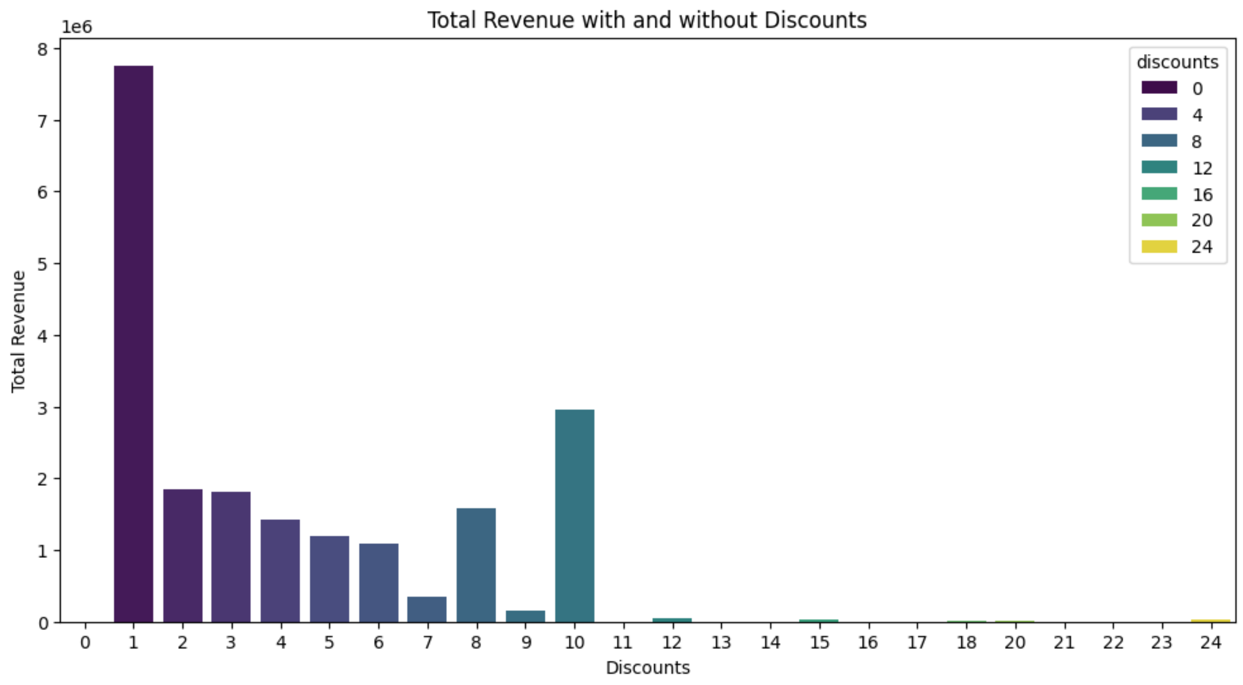
```
{"summary":"{\n  \"name\": \"order_frequency\",\n  \"rows\": 24,\n
\"fields\": [\n    {\n      \"column\": \"discounts\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7,\n        \"min\": 0,\n        \"max\": 24,\n
\"num_unique_values\": 24,\n        \"samples\": [\n          8,\n
16,\n          0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"total_orders\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 12129,\n        \"min\": 1,\n
\"max\": 58617,\n        \"num_unique_values\": 22,\n
\"samples\": [\n          3,\n          18,\n          5063\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"total_customers\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 10108,\n        \"min\": 1,\n
\"max\": 48609,\n        \"num_unique_values\": 23,\n
\"samples\": [\n          74,\n          635,\n          2\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"orders_per_customer\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.21125272688668825,\n        \"min\":
1.0,\n        \"max\": 1.8888888888888888,\n
\"num_unique_values\": 22,\n        \"samples\": [\n          1.5,\n
1.125,\n          1.1943854682708186\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"order_frequency"}
```

```
# Calculate average profit margin with and without discounts
avg_profit_margin = customer_data.groupby('discounts')
['profit_margin'].mean().reset_index()
avg_profit_margin.columns = ['Discounts', 'Average Profit Margin']
avg_profit_margin
```

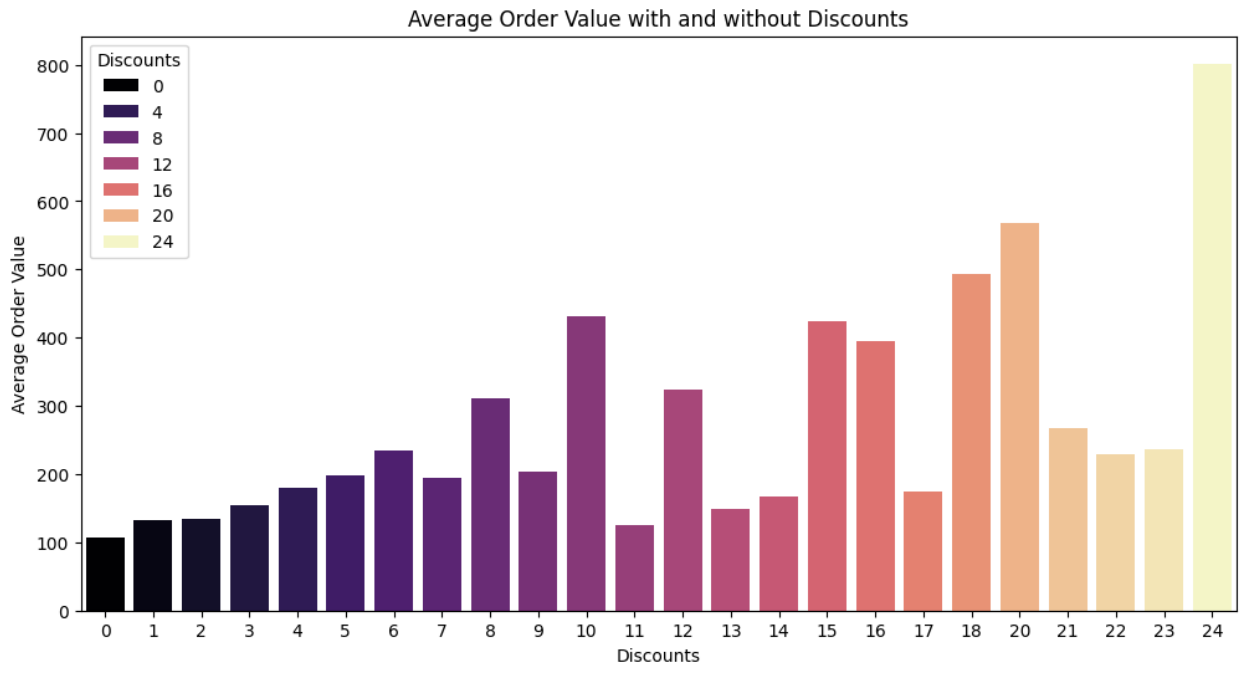
```
/usr/local/lib/python3.10/dist-packages/pandas/core/nanops.py:1010:
RuntimeWarning: invalid value encountered in subtract
  sqr = _ensure_numeric((avg - values) ** 2)
```

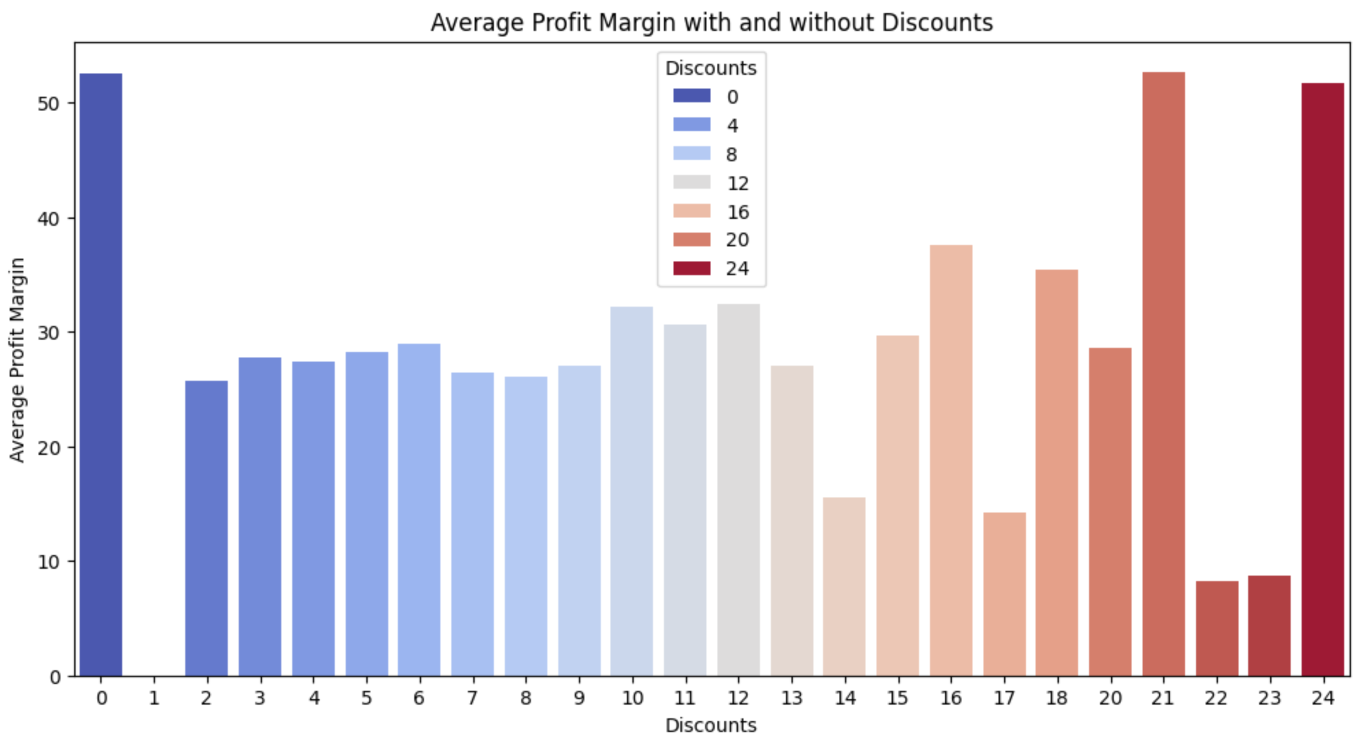
```
{"repr_error":"Out of range float values are not JSON compliant: -
inf","type":"dataframe","variable_name":"avg_profit_margin"}
```

```
plt.figure(figsize=(12, 6))
sns.barplot(x='discounts', y='total_revenue',
data=discount_comparison,hue = 'discounts', palette='viridis')
plt.title('Total Revenue with and without Discounts')
plt.xlabel('Discounts')
plt.ylabel('Total Revenue')
plt.show()
```



```python
plt.figure(figsize=(12, 6))
sns.barplot(x='Discounts', y='Average Order Value',
data=avg_order_value,hue = 'Discounts', palette='magma')
plt.title('Average Order Value with and without Discounts')
plt.xlabel('Discounts')
plt.ylabel('Average Order Value')
plt.show()
```





Average Profit Margin with and without Discounts
Discounts
0
4
8
12
16
20
24
Average Profit Margin
0
10
20
30
40
50
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
20
21
22
23
24
Discounts