1 Import Dataset to Analysis

```
!pip install -q -U google-genai
!pip install numpy pandas
!pip install -q -U google-genai
```

```
Collecting google-genai
```

Once we log in with the appropriate account (for one any free Bytecoin access to), we will connect our notes in the **BigQuery Project** to this Gemini API: `https://aistudio.google.com/app/apikey`

```
!gcloud auth login --no-browser
!gcloud --version
!gcloud config set project [project]
!bq --version && bq mk --dataset --location=EU technical_challenge_analytics
```

We can then use SQL to query your BigQuery data and save the output of the query as a dataframe with the following code:

```
# Magic command to enable cell magic.
%load_ext google.cloud.bigquery.dbtools.extensions.bigquery_magics

# Run a query.
%%bq query --project=[project] --to_dataframe df_name
SELECT
-- Select all columns from the table.
*
FROM `[project].[dataset].[table_name]`
LIMIT 1000
```

1.1 Load Libraries

Load the main libraries in **Python** to carry out exploratory data analysis on this notebook.

```
!pip install numpy pandas
!pip install -q -U google-genai
```

```
Collecting google-genai
```

Part. 1 - Technical Challenge Analysis «Commerce Events History in Cosmetics Shop» Dataset

This dataset includes behavioral data from **mid-sized cosmetics online store** from **Nov 2019 to Jan 2020**. Each entry in the dataset represents an event related to products and users. The events consist of different types such as views, items added to cart, items removed from cart, and purchases.

1.1 Data Overview

To starts, we present the data and **provide meaningful insights**. At start by loading the dataset and performing an **initial analysis**. This will include examining the structure of the data, the types of columns available, and some basic statistics. Let's begin by loading and exploring the data.

```
# Show a general view of data
print("Dataset shape:", df.shape)
df.head(10)
```

	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
0	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.96	415836715	d7b7fb73-c3d9-4783-8237-b4d3f4d4d8d0
1	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	d1e7c1f8-f6c4-4ee7-81a6-f7d1f4
2	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	0f1e2d3c-4b5a-6789-7e8f-9a0b1c2d3e4f
3	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	4a1b2c3d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
4	2020-01-01 00:00:00	remove_from_cart	5802432	1487580005754445313	NaN	aqua	7.70	549220573	9f8e7d6c-5b4a-3210-fedc-ba9876543210
5	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	2b3c4d5e-6f7a-8b9c-0d1e-2f3a4b5c6d7e
6	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	7c8d9e0f-1a2b-3c4d-5e6f-7a8b9c0d1e2f
7	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	5d6e7f8a-9b0c-1d2e-3f4a-5b6c7d8e9f0a
8	2020-01-01 00:00:00	remove_from_cart	5802432	1487580005754445313	NaN	aqua	7.70	549220573	1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
9	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	6a7b8c9d-0e1f-2a3b-4c5d-6e7f8a9b0c1d

The dataset provides the same following columns:

- event_time : Timestamp of the event.
- event_type : Type of event (e.g., 'view', 'remove_from_cart', 'purchase').
- product_id : Identifier for the product.
- category_id : Identifier for the product category.
- category_code : Category of the product (where one meaning).
- brand : Brand of the product (some entries are missing).
- price : Price of the product.
- user_id : Identifier for the user.
- user_session : Identifier for the user session.

```
# A preliminary check for duplicates, nulls, number of non-null values, etc.
df.info()
```

```
<class> pandas.core.frame.DataFrame'>
RangeIndex: 885129 entries, 0 to 885128
Data columns (total 9 columns):
 #   column       Dtype
---  ------       -----
 0   event_time   object
 1   event_type   object
 2   product_id   int64
 3   category_id  int64
 4   category_code  object
 5   brand        object
 6   price        float64
 7   user_id      int64
 8   user_session  object
dtypes: float64(1), int64(3), object(5)
memory usage: 61.3 MB
```

```
# A summary for the dataset
df.describe()
```

	product_id	category_id	price	user_id
count	2.000000e+3	2.000000e+3	2.000000e+3	2.000000e+3
mean	5.802432e+6	1.487580e+3	7.960000e+0	5.492207e+8
std	0.000000e+0	1.487580e+3	0.000000e+0	0.000000e+0
min	5802432.000000	1487580005754445313.000000	7.960000	549220573.000000
25%	5802432.000000	1487580005754445313.000000	7.960000	549220573.000000
50%	5802432.000000	1487580005754445313.000000	7.960000	549220573.000000
75%	5802432.000000	1487580005754445313.000000	7.960000	549220573.000000
max	5802432.000000	1487580005754445313.000000	7.960000	549220573.000000

To get a better understanding of the data, I will perform the following analysis:

- **Check for missing values** in important columns like `category_code`, `brand`, and `price`.
- **Distribution of `event_type`** to see the frequency of each event.
- Check for **missing values** in important columns like `category_code`, `brand`, and `user_type`.
- **Summary of unique values** in columns like `product_id`, `user_id`, and `user_session`.

```
# Check for missing values
print("Missing values:\n", df.isnull().sum())

# Distribution of event_type
print(df['event_type'].value_counts())

# Check for missing values in important columns
missing_cols = ['category_code', 'brand', 'price']
print("Missing values in important columns:\n", df[missing_cols].isnull().sum())

# Summary of unique values in key columns
print("Unique values in key columns:")
for col in ['product_id', 'user_id', 'user_session']:
    print(f"{col}: {df[col].nunique()}")
```

```
Missing values:
 event_time    0
event_type    0
product_id    0
category_id   0
category_code  885129
brand         864162
price         0
user_id       0
user_session  0
dtype: int64
event_type
view                 884451
remove_from_cart     678
Name: count, dtype: int64
Missing values in important columns:
 category_code  885129
brand          864162
price          0
Unique values in key columns:
product_id: 1
user_id: 1
user_session: 2
```

```
# Distribution of numerical features
df[['price']].describe()
```

	price
count	2.000000e+3
mean	7.960000e+0
std	0.000000e+0
min	7.960000
25%	7.960000
50%	7.960000
75%	7.960000
max	7.960000

```
# Check for duplicate rows
duplicate_count = df.duplicated().sum()
print("Number of duplicate rows:", duplicate_count)

# Optionally, if you want to see the duplicate rows:
if duplicate_count > 0:
    duplicates = df[df.duplicated(keep=False)]
    print("Duplicate rows:")
    display(duplicates.head())
```

```
Number of duplicate rows: 883129
Duplicate rows:
```

	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
0	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.96	415836715	d7b7fb73-c3d9-4783-8237-b4d3f4d4d8d0
1	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	d1e7c1f8-f6c4-4ee7-81a6-f7d1f4
2	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	0f1e2d3c-4b5a-6789-7e8f-9a0b1c2d3e4f
3	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	4a1b2c3d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
4	2020-01-01 00:00:00	remove_from_cart	5802432	1487580005754445313	NaN	aqua	7.70	549220573	9f8e7d6c-5b4a-3210-fedc-ba9876543210

Also, notice that the `user_session` **column has 4916** n/a values. The `user_session` should be used as primary key in the database and should never n/a values.

Therefore, the `category_code` field is to be removed since the n/a values of `user_session` there also filled with the profile which needed to be removed as well.

```
# Check the unique values in user_session to understand the missing values
print("Unique values in user_session:", df['user_session'].nunique())

# Analyze the rows with missing user_session values
missing_session = df[df['user_session'].isnull()]
print("Sample of rows with missing user_session:")
display(missing_session.head())

# Check if these rows have other important information
print("Rows with missing user_session:", len(missing_session))
print("Rows with both missing user_session and missing category_code:",
      ((df['user_session'].isnull() & df['category_code'].isnull()).sum()))
```

```
Unique values in user_session: 2
Sample of rows with missing user_session:
```

	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
0	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.96	415836715	d7b7fb73-c3d9-4783-8237-b4d3f4d4d8d0
1	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	d1e7c1f8-f6c4-4ee7-81a6-f7d1f4
2	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	0f1e2d3c-4b5a-6789-7e8f-9a0b1c2d3e4f
3	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	4a1b2c3d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
4	2020-01-01 00:00:00	remove_from_cart	5802432	1487580005754445313	NaN	aqua	7.70	549220573	9f8e7d6c-5b4a-3210-fedc-ba9876543210

```
Rows with missing user_session: 883129
Rows with both missing user_session and missing category_code: 883129
```

```
# Analyze the unique values of event_type, categories, and brands
print("Unique event types:", df['event_type'].nunique())
print("Unique categories:", df['category_id'].nunique())
print("Unique brands:", df['brand'].nunique())
print("Unique products:", df['product_id'].nunique())

# Check the most common values
print("\nTop 10 most common event types:")
print(df['event_type'].value_counts().head(10))
print("\nTop 10 most common categories:")
print(df['category_id'].value_counts().head(10))
print("\nTop 10 most common brands:")
print(df['brand'].value_counts().head(10))
print("\nTop 10 most common products:")
print(df['product_id'].value_counts().head(10))

# Check the distribution of prices
print("\nPrice statistics:")
print(df['price'].describe())

# Check the time range of the data
print("\nTime range:")
print("Earliest event:", df['event_time'].min())
print("Latest event:", df['event_time'].max())
```

```
Unique event types: 2
Unique categories: 1
Unique brands: 1
Unique products: 1

Top 10 most common event types:
event_type
view                 884451
remove_from_cart     678
Name: count, dtype: int64

Top 10 most common categories:
category_id
1487580005754445313    2000
Name: count, dtype: int64

Top 10 most common brands:
brand
aqua    2000
Name: count, dtype: int64

Top 10 most common products:
product_id
5802432    2000
Name: count, dtype: int64

Price statistics:
count  2000.0
mean      7.96
std       0.0
min       7.96
25%       7.96
50%       7.96
75%       7.96
max       7.96
Name: price, dtype: float64

Time range:
Earliest event: 2020-01-01 00:00:00
Latest event: 2020-01-01 00:00:00
```

We will analyze over a dataset of **885129 rows** with **9 columns** containing basic information of user (`user_id`), user session (`user_session`), product (`product_id`, `category_id`, `category_code`, `brand`, `price`) and event (`event_time`, `event_type`). Additionally, two columns containing information of products, `category_code` and brand have missing values, which will be given more summaries of those attributes.

Column	Description
`event_time`	Time when event happened at (in UTC).
`event_type`	The kind of interaction with the product. Possible values: `view` (product was viewed), `cart` (added to cart), `purchase` (purchased).
`product_id`	Unique ID of a product
`category_id`	Product category ID
`category_code`	Category name, if available (e.g., `electronics.smartphone`)
`brand`	Brand name (e.g., `samsung`, `apple`)
`price`	Product price in Euros
`user_id`	Anonymized unique ID of a user
`user_session`	Temporary session ID (resets after inactivity)

What we don't have (that may support the analysis for further recommendation system) are **user information** (age, gender, country, etc.) **user behavior** (clicks, scrolls, time spent, etc.) **product details** (beyond basic IDs, no product names, descriptions, or attributes) **marketing data** (no ad campaigns, no referral sources) **time dimension** (only one day of data, no seasonality, no trends). Other, the category_code is obtained from the hierarchical arrangement of the taxonomy within the category_id, is utilized for internal organization and system functionality. Moreover, information regarding not category_code is not available in the dataset, so the analysis proposes this column will no longer be used.

- Drop the `category_code` column since **2000/2000 of the data is missing** and it won't be useful in the analysis.

```
# Create a copy of the dataset to work with
df_clean = df.copy()

# Drop the category_code column since it's mostly null
df_clean = df_clean.drop('category_code', axis=1)

# Convert event_time to datetime
df_clean['event_time'] = pd.to_datetime(df_clean['event_time'])

# Create a session duration feature
session_duration = df_clean.groupby('user_session')['event_time'].agg(['min', 'max'])
session_duration['duration'] = (session_duration['max'] - session_duration['min'].dt.seconds / 60
df_clean = df_clean.merge(session_duration[['duration'], on='user_session')

# Create a flag for purchase events
df_clean['is_purchase'] = (df_clean['event_type'] == 'purchase')

# Create a hour of the day feature
df_clean['hour'] = df_clean['event_time'].dt.hour

# Create a day of the week feature
df_clean['day_of_week'] = df_clean['event_time'].dt.dayofweek

# Display the shape after cleaning
print("Shape after cleaning:", df_clean.shape)

# Show the first few rows of the cleaned dataset
df_clean.head()
```

```
Shape after cleaning: (885129, 10)
```

	event_time	event_type	product_id	category_id	brand	price	user_id	user_session
0	2020-01-01 00:00:00	view	5802432	1487580005754445313	aqua	7.96	415836715	d7b7fb73-c3d9-4783-8237-b4d3f4d4d8d0
1	2020-01-01 00:00:00	view	5802432	1487580005754445313	aqua	7.70	549220573	d1e7c1f8-f6c4-4ee7-81a6-f7d1f4
2	2020-01-01 00:00:00	view	5802432	1487580005754445313	aqua	7.70	549220573	0f1e2d3c-4b5a-6789-7e8f-9a0b1c2d3e4f
3	2020-01-01 00:00:00	view	5802432	1487580005754445313	aqua	7.70	549220573	4a1b2c3d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
4	2020-01-01 00:00:00	remove_from_cart	5802432	1487580005754445313	aqua	7.70	549220573	9f8e7d6c-5b4a-3210-fedc-ba9876543210

Also, notice that the `user_session` **column has 4916** n/a values. The `user_session` should be used as primary key in the database and should never n/a values.

Therefore, the `category_code` field is to be removed since the n/a values of `user_session` there also filled with the profile which needed to be removed as well.

```
# Check the unique values in user_session to understand the missing values
print("Unique values in user_session:", df['user_session'].nunique())

# Analyze the rows with missing user_session values
missing_session = df[df['user_session'].isnull()]
print("Sample of rows with missing user_session:")
display(missing_session.head())

# Check if these rows have other important information
print("Rows with missing user_session:", len(missing_session))
print("Rows with both missing user_session and missing category_code:",
      ((df['user_session'].isnull() & df['category_code'].isnull()).sum()))
```

```
Unique values in user_session: 2
Sample of rows with missing user_session:
```

	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
0	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.96	415836715	d7b7fb73-c3d9-4783-8237-b4d3f4d4d8d0
1	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	d1e7c1f8-f6c4-4ee7-81a6-f7d1f4
2	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	0f1e2d3c-4b5a-6789-7e8f-9a0b1c2d3e4f
3	2020-01-01 00:00:00	view	5802432	1487580005754445313	NaN	aqua	7.70	549220573	4a1b2c3d-5e6f-7a8b-9c0d-1e2f3a4b5c6d
4	2020-01-01 00:00:00	remove_from_cart	5802432	1487580005754445313	NaN	aqua	7.70	549220573	9f8e7d6c-5b4a-3210-fedc-ba9876543210

```
Rows with missing user_session: 883129
Rows with both missing user_session and missing category_code: 883129
```

```
# Analyze the distribution of event_type
print("Event type distribution:")
event_counts = df['event_type'].value_counts()
print(event_counts)

# Calculate percentages
print("\nPercentage distribution:")
print(df['event_type'].value_counts(normalize=True) * 100)

# Create a bar chart
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='event_type')
plt.title('Distribution of Event Types')
plt.xlabel('Event Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Event type distribution:
event_type
view                 884451
remove_from_cart     678
Name: count, dtype: int64

Percentage distribution:
event_type
view                 99.923
remove_from_cart      0.077
Name: count, dtype: float64
```

```
# Create a function to analyze product performance
def analyze_product_performance(df):
    product_metrics = df.groupby('product_id').agg{
        'event_type': lambda x: (x == 'view').sum(),
        'event_type': lambda x: (x == 'remove_from_cart').sum(),
        'price': 'mean',
        'user_id': 'nunique',
        'user_session': 'nunique'
    }
    product_metrics.columns = ['views', 'cart_adds', 'avg_price', 'unique_users', 'unique_sessions']
    product_metrics['conversion_rate'] = product_metrics['cart_adds'] / product_metrics['views']
    product_metrics['engagement_ratio'] = product_metrics['unique_sessions'] / product_metrics['unique_users']
    return product_metrics

# Analyze product performance
product_performance = analyze_product_performance(df_clean)
print("Product performance metrics:")
display(product_performance.head())

# Find products with high views but low conversion
print("\nProducts with high views but low conversion:")
high_views = product_performance[product_performance['views'] > product_performance['views'].quantile(0.75)]
low_conversion = high_views[high_views['conversion_rate'] < high_views['conversion_rate'].quantile(0.25)]
display(low_conversion.head())
```

```
Product performance metrics:
```

	views	cart_adds	avg_price	unique_users	unique_sessions	conversion_rate	engagement_ratio
product_id							
5802432	884451	678	7.96	1	2	0.000767	2.000000

```
Products with high views but low conversion:
```

	views	cart_adds	avg_price	unique_users	unique_sessions	conversion_rate	engagement_ratio
product_id							
5802432	884451	678	7.96	1	2	0.000767	2.000000

```
# Analyze price ranges and their performance
price_ranges = pd.cut(df_clean['price'], bins=[0, 10, 20, 30, 40, 50, 100, 200])
price_analysis = df_clean.groupby(price_ranges).agg{
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum(),
    'user_id': 'nunique'
}
price_analysis.columns = ['views', 'cart_adds', 'unique_users']
price_analysis['conversion_rate'] = price_analysis['cart_adds'] / price_analysis['views']

print("Price range analysis:")
display(price_analysis)

# Find the most popular price ranges
print("\nMost popular price ranges:")
print(price_analysis.sort_values('views', ascending=False))
```

```
Price range analysis:
```

price_range	views	cart_adds	unique_users	conversion_rate
(0, 10]	884451	678	1	0.000767
(10, 20]	0	0	0	NaN
(20, 30]	0	0	0	NaN
(30, 40]	0	0	0	NaN
(40, 50]	0	0	0	NaN
(50, 100]	0	0	0	NaN
(100, 200]	0	0	0	NaN

```
Most popular price ranges:
```

	views	cart_adds	unique_users
(0, 10]	884451	678	1
(10, 20]	0	0	0
(20, 30]	0	0	0
(30, 40]	0	0	0
(40, 50]	0	0	0
(50, 100]	0	0	0
(100, 200]	0	0	0

```
# Analyze brand performance
brand_performance = df_clean.groupby('brand').agg{
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum(),
    'price': 'mean',
    'user_id': 'nunique',
    'product_id': 'nunique'
}
brand_performance.columns = ['views', 'cart_adds', 'avg_price', 'unique_users', 'unique_products']
brand_performance['conversion_rate'] = brand_performance['cart_adds'] / brand_performance['views']

print("Brand performance:")
display(brand_performance.head)

# Find brands with high views but low conversion
print("\nBrands with high views but low conversion:")
high_views = brand_performance[brand_performance['views'] > brand_performance['views'].quantile(0.75)]
low_conversion = high_views[high_views['conversion_rate'] < high_views['conversion_rate'].quantile(0.25)]
display(low_conversion.head())
```

```
Brand performance:
```

	views	cart_adds	avg_price	unique_users	unique_products	conversion_rate
brand						
aqua	884451	678	7.96	1	1	0.000767

```
Brands with high views but low conversion:
```

	views	cart_adds	avg_price	unique_users	unique_products	conversion_rate
brand						
aqua	884451	678	7.96	1	1	0.000767

```
# Analyze user behavior patterns
user_behavior = df_clean.groupby('user_id').agg{
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum(),
    'product_id': 'nunique',
    'brand': 'nunique',
    'price': 'mean',
    'user_session': 'nunique'
}
user_behavior.columns = ['views', 'cart_adds', 'unique_products', 'unique_brands', 'avg_price', 'sessions']
user_behavior['conversion_rate'] = user_behavior['cart_adds'] / user_behavior['views']

print("User behavior patterns:")
display(user_behavior.head)

# Find high-value users
print("\nHigh-value users:")
high_value = user_behavior[user_behavior['views'] > user_behavior['views'].quantile(0.75)]
display(high_value.head())
```

```
User behavior patterns:
```

	views	cart_adds	unique_products	unique_brands	avg_price	sessions	conversion_rate
user_id							
415836715	0	0	1	1	7.96	1	NaN
549220573	1999	678	1	1	7.70	2	0.339170

```
High-value users:
```

	views	cart_adds	unique_products	unique_brands	avg_price	sessions	conversion_rate
user_id							
549220573	1999	678	1	1	7.70	2	0.339170

```
# Analyze session patterns
session_patterns = df_clean.groupby('user_session').agg{
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum(),
    'product_id': 'nunique',
    'brand': 'nunique',
    'price': 'mean',
    'duration': 'first',
    'hour': 'first',
    'day_of_week': 'first'
}
session_patterns.columns = ['views', 'cart_adds', 'unique_products', 'unique_brands', 'avg_price', 'duration', 'hour', 'day_of_week']
session_patterns['conversion_rate'] = session_patterns['cart_adds'] / session_patterns['views']

print("Session patterns:")
display(session_patterns.head)

# Find sessions with high views but low conversion
print("\nSessions with high views but low conversion:")
high_views = session_patterns[session_patterns['views'] > session_patterns['views'].quantile(0.75)]
low_conversion = high_views[high_views['conversion_rate'] < high_views['conversion_rate'].quantile(0.25)]
display(low_conversion.head())
```

```
Session patterns:
```

	views	cart_adds	unique_products	unique_brands	avg_price	duration	hour	day_of_week	conversion_rate
user_session									
0f1e2d3c-4b5a-6789-7e8f-9a0b1c2d3e4f	0	0	1	1	7.70	0.0	0	0	NaN
d1e7c1f8-f6c4-4ee7-81a6-f7d1f4	0	0	1	1	7.70	0.0	0	0	NaN
d7b7fb73-c3d9-4783-8237-b4d3f4d4d8d0	0	0	1	1	7.96	0.0	0	0	NaN
4a1b2c3d-5e6f-7a8b-9c0d-1e2f3a4b5c6d	1999	678	1	1	7.70	0.0	0	0	0.339170
9f8e7d6c-5b4a-3210-fedc-ba9876543210	1999	678	1	1	7.70	0.0	0	0	0.339170

```
Sessions with high views but low conversion:
```

	views	cart_adds	unique_products	unique_brands	avg_price	duration	hour	day_of_week	conversion_rate
user_session									
4a1b2c3d-5e6f-7a8b-9c0d-1e2f3a4b5c6d	1999	678	1	1	7.70	0.0	0	0	0.339170
9f8e7d6c-5b4a-3210-fedc-ba9876543210	1999	678	1	1	7.70	0.0	0	0	0.339170

```
# Analyze time patterns
time_patterns = df_clean.groupby('hour').agg{
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum,
    'user_id': 'nunique',
    'price': 'mean'
}
time_patterns.columns = ['views', 'cart_adds', 'unique_users', 'avg_price']
time_patterns['conversion_rate'] = time_patterns['cart_adds'] / time_patterns['views']

print("Time patterns:")
display(time_patterns.head)

# Find peak hours
print("\nPeak hours:")
print(time_patterns.sort_values('views', ascending=False))
```

```
Time patterns:
```

	views	cart_adds	unique_users	avg_price	conversion_rate
hour					
0	884451	678	1	7.96	0.000767

```
Peak hours:
```

	views	cart_adds	unique_users
hour			
0	884451	678	1

```
# Analyze price patterns
price_patterns = df_clean.groupby('price').agg{
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum,
    'user_id': 'nunique',
    'product_id': 'nunique'
}
price_patterns.columns = ['views', 'cart_adds', 'unique_users', 'unique_products']
price_patterns['conversion_rate'] = price_patterns['cart_adds'] / price_patterns['views']

print("Price patterns:")
display(price_patterns.head)

# Find optimal price points
print("\nOptimal price points:")
print(price_patterns.sort_values('conversion_rate', ascending=False))
```

```
Price patterns:
```

	views	cart_adds	unique_users	unique_products	conversion_rate
price					
7.96	884451	678	1	1	0.000767

```
Optimal price points:
```

	views	cart_adds	unique_users	unique_products
price				
7.96	884451	678	1	1

```
# Analyze product lifecycle
product_lifecycle = df_clean.groupby('product_id').agg{
    'event_time': 'min',
    'event_time': 'max',
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum,
    'price': 'mean',
    'brand': 'first',
    'category_id': 'first'
}
product_lifecycle.columns = ['first_seen', 'last_seen', 'views', 'cart_adds', 'avg_price', 'brand', 'category_id']
product_lifecycle['conversion_rate'] = product_lifecycle['cart_adds'] / product_lifecycle['views']
product_lifecycle['lifespan_days'] = (product_lifecycle['last_seen'] - product_lifecycle['first_seen'].dt.days

print("Product lifecycle:")
display(product_lifecycle.head)

# Find products with long lifespan but low conversion
print("\nProducts with long lifespan but low conversion:")
long_lifespan = product_lifecycle[product_lifecycle['lifespan_days'] > product_lifecycle['lifespan_days'].quantile(0.75)]
low_conversion = long_lifespan[long_lifespan['conversion_rate'] < long_lifespan['conversion_rate'].quantile(0.25)]
display(low_conversion.head())
```

```
Product lifecycle:
```

	first_seen	last_seen	views	cart_adds	avg_price	brand	category_id	conversion_rate	lifespan_days
product_id									
5802432	2020-01-01 00:00:00	2020-01-01 00:00:00	884451	678	7.96	aqua	1487580005754445313	0.000767	0

```
Products with long lifespan but low conversion:
```

	first_seen	last_seen	views	cart_adds	avg_price	brand	category_id	conversion_rate	lifespan_days
product_id									
5802432	2020-01-01 00:00:00	2020-01-01 00:00:00	884451	678	7.96	aqua	1487580005754445313	0.000767	0

```
# Analyze price elasticity
price_elasticity = df_clean.groupby('price').agg{
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum,
    'user_id': 'nunique'
}
price_elasticity.columns = ['views', 'cart_adds', 'unique_users']
price_elasticity['conversion_rate'] = price_elasticity['cart_adds'] / price_elasticity['views']

print("Price elasticity:")
display(price_elasticity.head)

# Find optimal price points
print("\nOptimal price points:")
print(price_elasticity.sort_values('conversion_rate', ascending=False)
```

```
Price elasticity:
```

	views	cart_adds	unique_users	conversion_rate
price				
7.96	884451	678	1	0.000767

```
Optimal price points:
```

	views	cart_adds	unique_users
price			
7.96	884451	678	1

```
# Analyze customer segments
customer_segments = df_clean.groupby('user_id').agg{
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum,
    'product_id': 'nunique',
    'brand': 'nunique',
    'price': 'mean',
    'user_session': 'nunique'
}
customer_segments.columns = ['views', 'cart_adds', 'unique_products', 'unique_brands', 'avg_price', 'sessions']
customer_segments['conversion_rate'] = customer_segments['cart_adds'] / customer_segments['views']

print("Customer segments:")
display(customer_segments.head)

# Find high-value customers
print("\nHigh-value customers:")
high_value = customer_segments[customer_segments['views'] > customer_segments['views'].quantile(0.75)]
display(high_value.head)
```

```
Customer segments:
```

	views	cart_adds	unique_products	unique_brands	avg_price	sessions	conversion_rate
user_id							
415836715	0	0	1	1	7.96	1	NaN
549220573	1999	678	1	1	7.70	2	0.339170

```
High-value customers:
```

	views	cart_adds	unique_products	unique_brands	avg_price	sessions	conversion_rate
user_id							
549220573	1999	678	1	1	7.70	2	0.339170

```
# Analyze session patterns
session_patterns = df_clean.groupby('user_session').agg{
    'event_type': lambda x: (x == 'view').sum(),
    'event_type': lambda x: (x == 'remove_from_cart').sum,
    'product_id': 'nunique',
    'brand': 'nunique',
    'price': 'mean',
    'duration': 'first',
    'hour': 'first',
    'day_of_week': 'first'
}
session_patterns.columns = ['views', 'cart_adds', 'unique_products', 'unique_brands', 'avg_price', 'duration', 'hour', 'day_of_week']
session_patterns['conversion_rate'] = session_patterns['cart_adds'] / session_patterns['views']

print("Session patterns:")
display(session_patterns.head)

# Find sessions with high views but low conversion
print("\nSessions with high views but low conversion:")
high_views = session_patterns[session_patterns['views'] > session_patterns['views'].quantile(0.75)]
low_conversion = high_views[high_views['conversion_rate'] < high_views['conversion_rate'].quantile(0.25)]
display(low_conversion.head())
```

```
Session patterns:
```

	views	cart_adds	unique_products	unique_brands	avg_price	duration	hour	day_of_week	conversion_rate
user_session									
0f1e2d3c-4b5a-6789-7e8f-9a0b1c2d3e4f	0	0	1	1	7.70	0.0	0	0	NaN
d1e7c1f8-f6c4-4ee7-81a6-f7d1f4	0	0	1	1	7.70	0.0	0	0	NaN
d7b7fb73-c3d9-4783-8237-b4d3f4d4d8d0	0	0	1	1	7.96	0.0	0	0	NaN
4a1b2c3d-5e6f-7a8b-9c0d-1e2f3a4b5c6d	1999	678	1	1	7.70	0.0	0	0	0.339170
9f8e7d6c-5b4a-3210-fedc-ba9876543210	1999	678	1	1	7.70	0.0	0	0	0.339170

```
Sessions with high views but low conversion:
```

	views	cart_adds	unique_products	unique_brands	avg_price	duration	hour	day_of_week	conversion_rate
user_session									
4a1b2c3d-5e6f-7a8b-9c0

```
count  price
mean   0.00
std    0.00
min    0.00
25%    0.00
50%    0.00
75%    0.00
max    0.00
```

```
# Count the number of products in each price range
price_range_counts = price_range_df['price'].value_counts().reset_index()
price_range_counts.columns = ['price_range', 'count']

# Calculate percentage
price_range_counts['percentage'] = (price_range_counts['count'] / price_range_counts['count'].sum() * 100).round(2)

# Display the results
print("Distribution of Products by Price Range:")
print(price_range_counts.to_string(index=False))
```

```
Distribution of Products by Price Range:
price_range  count  percentage
       Low   2000       33.33
      High   2000       33.33
    Medium   2000       33.33
Very High   2000       33.33
```


Most frequent price point is the range from **0 to 9** (almost 50% of all products), in contrast product with price **above 500 accounts for less than 2%** of the products. How about the brand? The price range of each brand can be segmented by using median price values.

```
# Median price of each brand
median_price = df.groupby('brand')['price'].median().reset_index()

print(median_price)
```

```
     brand  price
0    Anmak   1.00
1    Ahnuo   0.00
2   Ankeng   0.00
3   Anlong   0.00
4   Anpeng   0.00
5   Anqing   0.00
6   Anqiut   0.00
7   Anqiut   0.00
8   Anqiut   0.00
9   Anqiut   0.00
10  Anqiut   0.00
11  Anqiut   0.00
12  Anqiut   0.00
13  Anqiut   0.00
14  Anqiut   0.00
15  Anqiut   0.00
16  Anqiut   0.00
17  Anqiut   0.00
18  Anqiut   0.00
19  Anqiut   0.00
20  Anqiut   0.00
21  Anqiut   0.00
22  Anqiut   0.00
23  Anqiut   0.00
24  Anqiut   0.00
25  Anqiut   0.00
26  Anqiut   0.00
27  Anqiut   0.00
28  Anqiut   0.00
29  Anqiut   0.00
30  Anqiut   0.00
31  Anqiut   0.00
32  Anqiut   0.00
33  Anqiut   0.00
34  Anqiut   0.00
35  Anqiut   0.00
36  Anqiut   0.00
37  Anqiut   0.00
38  Anqiut   0.00
39  Anqiut   0.00
40  Anqiut   0.00
41  Anqiut   0.00
42  Anqiut   0.00
43  Anqiut   0.00
44  Anqiut   0.00
45  Anqiut   0.00
46  Anqiut   0.00
47  Anqiut   0.00
48  Anqiut   0.00
49  Anqiut   0.00
50  Anqiut   0.00
51  Anqiut   0.00
52  Anqiut   0.00
53  Anqiut   0.00
54  Anqiut   0.00
55  Anqiut   0.00
56  Anqiut   0.00
57  Anqiut   0.00
58  Anqiut   0.00
59  Anqiut   0.00
60  Anqiut   0.00
61  Anqiut   0.00
62  Anqiut   0.00
63  Anqiut   0.00
64  Anqiut   0.00
65  Anqiut   0.00
66  Anqiut   0.00
67  Anqiut   0.00
68  Anqiut   0.00
69  Anqiut   0.00
70  Anqiut   0.00
71  Anqiut   0.00
72  Anqiut   0.00
73  Anqiut   0.00
74  Anqiut   0.00
75  Anqiut   0.00
76  Anqiut   0.00
77  Anqiut   0.00
78  Anqiut   0.00
79  Anqiut   0.00
80  Anqiut   0.00
81  Anqiut   0.00
82  Anqiut   0.00
83  Anqiut   0.00
84  Anqiut   0.00
85  Anqiut   0.00
86  Anqiut   0.00
87  Anqiut   0.00
88  Anqiut   0.00
89  Anqiut   0.00
90  Anqiut   0.00
91  Anqiut   0.00
92  Anqiut   0.00
93  Anqiut   0.00
94  Anqiut   0.00
95  Anqiut   0.00
96  Anqiut   0.00
97  Anqiut   0.00
98  Anqiut   0.00
99  Anqiut   0.00
100 Anqiut   0.00
101 Anqiut   0.00
102 Anqiut   0.00
103 Anqiut   0.00
104 Anqiut   0.00
105 Anqiut   0.00
106 Anqiut   0.00
107 Anqiut   0.00
108 Anqiut   0.00
109 Anqiut   0.00
110 Anqiut   0.00
111 Anqiut   0.00
112 Anqiut   0.00
113 Anqiut   0.00
114 Anqiut   0.00
115 Anqiut   0.00
116 Anqiut   0.00
117 Anqiut   0.00
118 Anqiut   0.00
119 Anqiut   0.00
120 Anqiut   0.00
121 Anqiut   0.00
122 Anqiut   0.00
123 Anqiut   0.00
124 Anqiut   0.00
125 Anqiut   0.00
126 Anqiut   0.00
127 Anqiut   0.00
128 Anqiut   0.00
129 Anqiut   0.00
130 Anqiut   0.00
131 Anqiut   0.00
132 Anqiut   0.00
133 Anqiut   0.00
134 Anqiut   0.00
135 Anqiut   0.00
136 Anqiut   0.00
137 Anqiut   0.00
138 Anqiut   0.00
139 Anqiut   0.00
140 Anqiut   0.00
141 Anqiut   0.00
142 Anqiut   0.00
143 Anqiut   0.00
144 Anqiut   0.00
145 Anqiut   0.00
146 Anqiut   0.00
147 Anqiut   0.00
148 Anqiut   0.00
149 Anqiut   0.00
150 Anqiut   0.00
151 Anqiut   0.00
152 Anqiut   0.00
153 Anqiut   0.00
154 Anqiut   0.00
155 Anqiut   0.00
156 Anqiut   0.00
157 Anqiut   0.00
158 Anqiut   0.00
159 Anqiut   0.00
160 Anqiut   0.00
161 Anqiut   0.00
162 Anqiut   0.00
163 Anqiut   0.00
164 Anqiut   0.00
165 Anqiut   0.00
166 Anqiut   0.00
167 Anqiut   0.00
168 Anqiut   0.00
169 Anqiut   0.00
170 Anqiut   0.00
171 Anqiut   0.00
172 Anqiut   0.00
173 Anqiut   0.00
174 Anqiut   0.00
175 Anqiut   0.00
176 Anqiut   0.00
177 Anqiut   0.00
178 Anqiut   0.00
179 Anqiut   0.00
180 Anqiut   0.00
181 Anqiut   0.00
182 Anqiut   0.00
183 Anqiut   0.00
184 Anqiut   0.00
185 Anqiut   0.00
186 Anqiut   0.00
187 Anqiut   0.00
188 Anqiut   0.00
189 Anqiut   0.00
190 Anqiut   0.00
191 Anqiut   0.00
192 Anqiut   0.00
193 Anqiut   0.00
194 Anqiut   0.00
195 Anqiut   0.00
196 Anqiut   0.00
197 Anqiut   0.00
198 Anqiut   0.00
199 Anqiut   0.00
200 Anqiut   0.00
201 Anqiut   0.00
202 Anqiut   0.00
203 Anqiut   0.00
204 Anqiut   0.00
205 Anqiut   0.00
206 Anqiut   0.00
207 Anqiut   0.00
208 Anqiut   0.00
209 Anqiut   0.00
210 Anqiut   0.00
211 Anqiut   0.00
212 Anqiut   0.00
213 Anqiut   0.00
214 Anqiut   0.00
215 Anqiut   0.00
216 Anqiut   0.00
217 Anqiut   0.00
218 Anqiut   0.00
219 Anqiut   0.00
220 Anqiut   0.00
221 Anqiut   0.00
222 Anqiut   0.00
223 Anqiut   0.00
224 Anqiut   0.00
225 Anqiut   0.00
226 Anqiut   0.00
227 Anqiut   0.00
228 Anqiut   0.00
229 Anqiut   0.00
230 Anqiut   0.00
231 Anqiut   0.00
232 Anqiut   0.00
233 Anqiut   0.00
234 Anqiut   0.00
235 Anqiut   0.00
236 Anqiut   0.00
237 Anqiut   0.00
238 Anqiut   0.00
239 Anqiut   0.00
240 Anqiut   0.00
241 Anqiut   0.00
242 Anqiut   0.00
243 Anqiut   0.00
244 Anqiut   0.00
245 Anqiut   0.00
246 Anqiut   0.00
247 Anqiut   0.00
248 Anqiut   0.00
249 Anqiut   0.00
250 Anqiut   0.00
251 Anqiut   0.00
252 Anqiut   0.00
253 Anqiut   0.00
254 Anqiut   0.00
255 Anqiut   0.00
256 Anqiut   0.00
257 Anqiut   0.00
258 Anqiut   0.00
259 Anqiut   0.00
260 Anqiut   0.00
261 Anqiut   0.00
262 Anqiut   0.00
263 Anqiut   0.00
264 Anqiut   0.00
265 Anqiut   0.00
266 Anqiut   0.00
267 Anqiut   0.00
268 Anqiut   0.00
269 Anqiut   0.00
270 Anqiut   0.00
271 Anqiut   0.00
272 Anqiut   0.00
273 Anqiut   0.00
274 Anqiut   0.00
275 Anqiut   0.00
276 Anqiut   0.00
277 Anqiut   0.00
278 Anqiut   0.00
279 Anqiut   0.00
280 Anqiut   0.00
281 Anqiut   0.00
282 Anqiut   0.00
283 Anqiut   0.00
284 Anqiut   0.00
285 Anqiut   0.00
286 Anqiut   0.00
287 Anqiut   0.00
288 Anqiut   0.00
289 Anqiut   0.00
290 Anqiut   0.00
291 Anqiut   0.00
292 Anqiut   0.00
293 Anqiut   0.00
294 Anqiut   0.00
295 Anqiut   0.00
296 Anqiut   0.00
297 Anqiut   0.00
298 Anqiut   0.00
299 Anqiut   0.00
300 Anqiut   0.00
301 Anqiut   0.00
302 Anqiut   0.00
303 Anqiut   0.00
304 Anqiut   0.00
305 Anqiut   0.00
306 Anqiut   0.00
307 Anqiut   0.00
308 Anqiut   0.00
309 Anqiut   0.00
310 Anqiut   0.00
311 Anqiut   0.00
312 Anqiut   0.00
313 Anqiut   0.00
314 Anqiut   0.00
315 Anqiut   0.00
316 Anqiut   0.00
317 Anqiut   0.00
318 Anqiut   0.00
319 Anqiut   0.00
320 Anqiut   0.00
321 Anqiut   0.00
322 Anqiut   0.00
323 Anqiut   0.00
324 Anqiut   0.00
325 Anqiut   0.00
326 Anqiut   0.00
327 Anqiut   0.00
328 Anqiut   0.00
329 Anqiut   0.00
330 Anqiut   0.00
331 Anqiut   0.00
332 Anqiut   0.00
333 Anqiut   0.00
334 Anqiut   0.00
335 Anqiut   0.00
336 Anqiut   0.00
337 Anqiut   0.00
338 Anqiut   0.00
339 Anqiut   0.00
340 Anqiut   0.00
341 Anqiut   0.00
342 Anqiut   0.00
343 Anqiut   0.00
344 Anqiut   0.00
345 Anqiut   0.00
346 Anqiut   0.00
347 Anqiut   0.00
348 Anqiut   0.00
349 Anqiut   0.00
350 Anqiut   0.00
351 Anqiut   0.00
352 Anqiut   0.00
353 Anqiut   0.00
354 Anqiut   0.00
355 Anqiut   0.00
356 Anqiut   0.00
357 Anqiut   0.00
358 Anqiut   0.00
359 Anqiut   0.00
360 Anqiut   0.00
361 Anqiut   0.00
362 Anqiut   0.00
363 Anqiut   0.00
364 Anqiut   0.00
365 Anqiut   0.00
366 Anqiut   0.00
367 Anqiut   0.00
368 Anqiut   0.00
369 Anqiut   0.00
370 Anqiut   0.00
371 Anqiut   0.00
372 Anqiut   0.00
373 Anqiut   0.00
374 Anqiut   0.00
375 Anqiut   0.00
376 Anqiut   0.00
377 Anqiut   0.00
378 Anqiut   0.00
379 Anqiut   0.00
380 Anqiut   0.00
381 Anqiut   0.00
382 Anqiut   0.00
383 Anqiut   0.00
384 Anqiut   0.00
385 Anqiut   0.00
386 Anqiut   0.00
387 Anqiut   0.00
388 Anqiut   0.00
389 Anqiut   0.00
390 Anqiut   0.00
391 Anqiut   0.00
392 Anqiut   0.00
393 Anqiut   0.00
394 Anqiut   0.00
395 Anqiut   0.00
396 Anqiut   0.00
397 Anqiut   0.00
398 Anqiut   0.00
399 Anqiut   0.00
400 Anqiut   0.00
401 Anqiut   0.00
402 Anqiut   0.00
403 Anqiut   0.00
404 Anqiut   0.00
405 Anqiut   0.00
406 Anqiut   0.00
407 Anqiut   0.00
408 Anqiut   0.00
409 Anqiut   0.00
410 Anqiut   0.00
411 Anqiut   0.00
412 Anqiut   0.00
413 Anqiut   0.00
414 Anqiut   0.00
415 Anqiut   0.00
416 Anqiut   0.00
417 Anqiut   0.00
418 Anqiut   0.00
419 Anqiut   0.00
420 Anqiut   0.00
421 Anqiut   0.00
422 Anqiut   0.00
423 Anqiut   0.00
424 Anqiut   0.00
425 Anqiut   0.00
426 Anqiut   0.00
427 Anqiut   0.00
428 Anqiut   0.00
429 Anqiut   0.00
430 Anqiut   0.00
431 Anqiut   0.00
432 Anqiut   0.00
433 Anqiut   0.00
434 Anqiut   0.00
435 Anqiut   0.00
436 Anqiut   0.00
437 Anqiut   0.00
438 Anqiut   0.00
439 Anqiut   0.00
440 Anqiut   0.00
441 Anqiut   0.00
442 Anqiut   0.00
443 Anqiut   0.00
444 Anqiut   0.00
445 Anqiut   0.00
446 Anqiut   0.00
447 Anqiut   0.00
448 Anqiut   0.00
449 Anqiut   0.00
450 Anqiut   0.00
451 Anqiut   0.00
452 Anqiut   0.00
453 Anqiut   0.00
454 Anqiut   0.00
455 Anqiut   0.00
456 Anqiut   0.00
457 Anqiut   0.00
458 Anqiut   0.00
459 Anqiut   0.00
460 Anqiut   0.00
461 Anqiut   0.00
462 Anqiut   0.00
463 Anqiut   0.00
464 Anqiut   0.00
465 Anqiut   0.00
466 Anqiut   0.00
467 Anqiut   0.00
468 Anqiut   0.00
469 Anqiut   0.00
470 Anqiut   0.00
471 Anqiut   0.00
472 Anqiut   0.00
473 Anqiut   0.00
474 Anqiut   0.00
475 Anqiut   0.00
476 Anqiut   0.00
477 Anqiut   0.00
478 Anqiut   0.00
479 Anqiut   0.00
480 Anqiut   0.00
481 Anqiut   0.00
482 Anqiut   0.00
483 Anqiut   0.00
484 Anqiut   0.00
485 Anqiut   0.00
486 Anqiut   0.00
487 Anqiut   0.00
488 Anqiut   0.00
489 Anqiut   0.00
490 Anqiut   0.00
491 Anqiut   0.00
492 Anqiut   0.00
493 Anqiut   0.00
494 Anqiut   0.00
495 Anqiut   0.00
496 Anqiut   0.00
497 Anqiut   0.00
498 Anqiut   0.00
499 Anqiut   0.00
500 Anqiut   0.00
501 Anqiut   0.00
502 Anqiut   0.00
503 Anqiut   0.00
504 Anqiut   0.00
505 Anqiut   0.00
506 Anqiut   0.00
507 Anqiut   0.00
508 Anqiut   0.00
509 Anqiut   0.00
510 Anqiut   0.00
511 Anqiut   0.00
512 Anqiut   0.00
513 Anqiut   0.00
514 Anqiut   0.00
515 Anqiut   0.00
516 Anqiut   0.00
517 Anqiut   0.00
518 Anqiut   0.00
519 Anqiut   0.00
520 Anqiut   0.00
521 Anqiut   0.00
522 Anqiut   0.00
523 Anqiut   0.00
524 Anqiut   0.00
525 Anqiut   0.00
526 Anqiut   0.00
527 Anqiut   0.00
528 Anqiut   0.00
529 Anqiut   0.00
530 Anqiut   0.00
531 Anqiut   0.00
532 Anqiut   0.00
533 Anqiut   0.00
534 Anqiut   0.00
535 Anqiut   0.00
536 Anqiut   0.00
537 Anqiut   0.00
538 Anqiut   0.00
539 Anqiut   0.00
540 Anqiut   0.00
541 Anqiut   0.00
542 Anqiut   0.00
543 Anqiut   0.00
544 Anqiut   0.00
545 Anqiut   0.00
546 Anqiut   0.00
547 Anqiut   0.00
548 Anqiut   0.00
549 Anqiut   0.00
550 Anqiut   0.00
551 Anqiut   0.00
552 Anqiut   0.00
553 Anqiut   0.00
554 Anqiut   0.00
555 Anqiut   0.00
556 Anqiut   0.00
557 Anqiut   0.00
558 Anqiut   0.00
559 Anqiut   0.00
560 Anqiut   0.00
561 Anqiut   0.00
562 Anqiut   0.00
563 Anqiut   0.00
564 Anqiut   0.00
565 Anqiut   0.00
566 Anqiut   0.00
567 Anqiut   0.00
568 Anqiut   0.00
569 Anqiut   0.00
570 Anqiut   0.00
571 Anqiut   0.00
572 Anqiut   0.00
573 Anqiut   0.00
574 Anqiut   0.00
575 Anqiut   0.00
576 Anqiut   0.00
577 Anqiut   0.00
578 Anqiut   0.00
579 Anqiut   0.00
580 Anqiut   0.00
581 Anqiut   0.00
582 Anqiut   0.00
583 Anqiut   0.00
584 Anqiut   0.00
585 Anqiut   0.00
586 Anqiut   0.00
587 Anqiut   0.00
588 Anqiut   0.00
589 Anqiut   0.00
590 Anqiut   0.00
591 Anqiut   0.00
592 Anqiut   0.00
593 Anqiut   0.00
594 Anqiut   0.00
595 Anqiut   0.00
596 Anqiut   0.00
597 Anqiut   0.00
598 Anqiut   0.00
599 Anqiut   0.00
600 Anqiut   0.00
601 Anqiut   0.00
602 Anqiut   0.00
603 Anqiut   0.00
604 Anqiut   0.00
605 Anqiut   0.00
606 Anqiut   0.00
607 Anqiut   0.00
608 Anqiut   0.00
609 Anqiut   0.00
610 Anqiut   0.00
611 Anqiut   0.00
612 Anqiut   0.00
613 Anqiut   0.00
614 Anqiut   0.00
615 Anqiut   0.00
616 Anqiut   0.00
617 Anqiut   0.00
618 Anqiut   0.00
619 Anqiut   0.00
620 Anqiut   0.00
621 Anqiut   0.00
622 Anqiut   0.00
623 Anqiut   0.00
624 Anqiut   0.00
625 Anqiut   0.00
626 Anqiut   0.00
627 Anqiut   0.00
628 Anqiut   0.00
629 Anqiut   0.00
630 Anqiut   0.00
631 Anqiut   0.00
632 Anqiut   0.00
633 Anqiut   0.00
634 Anqiut   0.00
635 Anqiut   0.00
636 Anqiut   0.00
637 Anqiut   0.00
638 Anqiut   0.00
639 Anqiut   0.00
640 Anqiut   0.00
641 Anqiut   0.00
642 Anqiut   0.00
643 Anqiut   0.00
644 Anqiut   0.00
645 Anqiut   0.00
646 Anqiut   0.00
647 Anqiut   0.00
648 Anqiut   0.00
649 Anqiut   0.00
650 Anqiut   0.00
651 Anqiut   0.00
652 Anqiut   0.00
653 Anqiut   0.00
654 Anqiut   0.00
655 Anqiut   0.00
656 Anqiut   0.00
657 Anqiut   0.00
658 Anqiut   0.00
659 Anqiut   0.00
660 Anqiut   0.00
661 Anqiut   0.00
662 Anqiut   0.00
663 Anqiut   0.00
664 Anqiut   0.00
665 Anqiut   0.00
666 Anqiut   0.00
667 Anqiut   0.00
668 Anqiut   0.00
669 Anqiut   0.00
670 Anqiut   0.00
671 Anqiut   0.00
672 Anqiut   0.00
673 Anqiut   0.00
674 Anqiut   0.00
675 Anqiut   0.00
676 Anqiut   0.00
677 Anqiut   0.00
678 Anqiut   0.00
679 Anqiut   0.00
680 Anqiut   0.00
681 Anqiut   0.00
682 Anqiut   0.00
683 Anqiut   0.00
684 Anqiut   0.00
685 Anqiut   0.00
686 Anqiut   0.00
687 Anqiut   0.00
688 Anqiut   0.00
689 Anqiut   0.00
690 Anqiut   0.00
691 Anqiut   0.00
692 Anqiut   0.00
693 Anqiut   0.00
694 Anqiut   0.00
695 Anqiut   0.00
696 Anqiut   0.00
697 Anqiut   0.00
698 Anqiut   0.00
699 Anqiut   0.00
700 Anqiut   0.00
701 Anqiut   0.00
702 Anqiut   0.00
703 Anqiut   0.00
704 Anqiut   0.00
705 Anqiut   0.00
706 Anqiut   0.00
707 Anqiut   0.00
708 Anqiut   0.00
709 Anqiut   0.00
710 Anqiut   0.00
711 Anqiut   0.00
712 Anqiut   0.00
713 Anqiut   0.00
714 Anqiut   0.00
715 Anqiut   0.00
716 Anqiut   0.00
717 Anqiut   0.00
718 Anqiut   0.00
719 Anqiut   0.00
720 Anqiut   0.00
721 Anqiut   0.00
722 Anqiut   0.00
723 Anqiut   0.00
724 Anqiut   0.00
725 Anqiut   0.00
726 Anqiut   0.00
727 Anqiut   0.00
728 Anqiut   0.00
729 Anqiut   0.00
730 Anqiut   0.00
731 Anqiut   0.00
732 Anqiut   0.00
733 Anqiut   0.00
734 Anqiut   0.00
735 Anqiut   0.00
736 Anqiut   0.00
737 Anqiut   0.00
738 Anqiut   0.00
739 Anqiut   0.00
740 Anqiut   0.00
741 Anqiut   0.00
742 Anqiut   0.00
743 Anqiut   0.00
744 Anqiut   0.00
745 Anqiut   0.00
746 Anqiut   0.00
747 Anqiut   0.00
748 Anqiut   0.00
749 Anqiut   0.00
750 Anqiut   0.00
751 Anqiut   0.00
752 Anqiut   0.00
753 Anqiut   0.00
754 Anqiut   0.00
755 Anqiut   0.00
756 Anqiut   0.00
757 Anqiut   0.00
758 Anqiut   0.00
759 Anqiut   0.00
760 Anqiut   0.00
761 Anqiut   0.00
762 Anqiut   0.00
763 Anqiut   0.00
764 Anqiut   0.00
765 Anqiut   0.00
766 Anqiut   0.00
767 Anqiut   0.00
768 Anqiut   0.00
769 Anqiut   0.00
770 Anqiut   0.00
771 Anqiut   0.00
772 Anqiut   0.00
773 Anqiut   0.00
774 Anqiut   0.00
775 Anqiut   0.00
776 Anqiut   0.00
777 Anqiut   0.00
778 Anqiut   0.00
779 Anqiut   0.00
780 Anqiut   0.00
781 Anqiut   0.00
782 Anqiut   0.00
783 Anqiut   0.00
784 Anqiut   0.00
785 Anqiut   0.00
786 Anqiut   0.00
787 Anqiut   0.00
788 Anqiut   0.00
789 Anqiut   0.00
790 Anqiut   0.00
791 Anqiut   0.00
792 Anqiut   0.00
793 Anqiut   0.00
794 Anqiut   0.00
795 Anqiut   0.00
796 Anqiut   0.00
797 Anqiut   0.00
798 Anqiut   0.00
799 Anqiut   0.00
800 Anqiut   0.00
801 Anqiut   0.00
802 Anqiut   0.00
803 Anqiut   0.00
804 Anqiut   0.00
805 Anqiut   0.00
806 Anqiut   0.00
807 Anqiut   0.00
808 Anqiut   0.00
809 Anqiut   0.00
810 Anqiut   0.00
811 Anqiut   0.00
812 Anqiut   0.00
813 Anqiut   0.00
814 Anqiut   0.00
815 Anqiut   0.00
816 Anqiut   0.00
817 Anqiut   0.00
818 Anqiut   0.00
819 Anqiut   0.00
820 Anqiut   0.00
821 Anqiut   0.00
822 Anqiut   0.00
823 Anqiut   0.00
824 Anqiut   0.00
825 Anqiut   0.00
826 Anqiut   0.00
827 Anqiut   0.00
828 Anqiut   0.00
829 Anqiut   0.00
830 Anqiut   0.00
831 Anqiut   0.00
832 Anqiut   0.00
833 Anqiut   0.00
834 Anqiut   0.00
835 Anqiut   0.00
836 Anqiut   0.00
837 Anqiut   0.00
838 Anqiut   0.00
839 Anqiut   0.00
840 Anqiut   0.00
841 Anqiut   0.00
842 Anqiut   0.00
843 Anqiut   0.00
844 Anqiut   0.00
845 Anqiut   0.00
846 Anqiut   0.00
847 Anqiut   0.00
848 Anqiut   0.00
849 Anqiut   0.00
850 Anqiut   0.00
851 Anqiut   0.00
852 Anqiut   0.00
853 Anqiut   0.00
854 Anqiut   0.00
855 Anqiut   0.00
856 Anqiut   0.00
857 Anqiut   0.00
858 Anqiut   0.00
859 Anqiut   0.00
860 Anqiut   0.00
861 Anqiut   0.00
862 Anqiut   0.00
863 Anqiut   0.00
864 Anqiut   0.00
865 Anqiut   0.00
866 Anqiut   0.00
867 Anqiut   0.00
868 Anqiut   0.00
869 Anqiut   0.00
870 Anqiut   0.00
871 Anqiut   0.00
872 Anqiut   0.00
873 Anqiut   0.00
874 Anqiut   0.00
875 Anqiut   0.00
876 Anqiut   0.00
877 Anqiut   0.00
878 Anqiut   0.00
879 Anqiut   0.00
880 Anqiut   0.00
881 Anqiut   0.00
882 Anqiut   0.00
883 Anqiut   0.00
884 Anqiut   0.00
885 Anqiut   0.00
886 Anqiut   0.00
887 Anqiut   0.00
888 Anqiut   0.00
889 Anqiut   0.00
890 Anqiut   0.00
891 Anqiut   0.00
892 Anqiut   0.00
893 Anqiut   0.00
894 Anqiut   0.00
895 Anqiut   0.00
896 Anqiut   0.00
897 Anqiut   0.00
898 Anqiut   0.00
899 Anqiut   0.00
900 Anqiut   0.00
901 Anqiut   0.00
902 Anqiut   0.00
903 Anqiut   0.00
904 Anqiut   0.00
905 Anqiut   0.00
906 Anqiut   0.00
907 Anqiut   0.00
908 Anqiut   0.00
909 Anqiut   0.00
910 Anqiut   0.00
911 Anqiut   0.00
912 Anqiut   0.00
913 Anqiut   0.00
914 Anqiut   0.00
915 Anqiut   0.00
916 Anqiut   0.00
917 Anqiut   0.00
918 Anqiut   0.00
919 Anqiut   0.00
920 Anqiut   0.00
921 Anqiut   0.00
922 Anqiut   0.00
923 Anqiut   0.00
924 Anqiut   0.00
925 Anqiut   0.00
926 Anqiut   0.00
927 Anqiut   0.00
928 Anqiut   0.00
929 Anqiut   0.00
930 Anqiut   0.00
931 Anqiut   0.00
932 Anqiut   0.00
933 Anqiut   0.00
934 Anqiut   0.00
935 Anqiut   0.00
936 Anqiut   0.00
937 Anqiut   0.00
938 Anqiut   0.00
939 Anqiut   0.00
940 Anqiut   0.00
941 Anqiut   0.00
942 Anqiut   0.00
943 Anqiut   0.00
944 Anqiut   0.00
945 Anqiut   0.00
946 Anqiut   0.00
947 Anqiut   0.00
948 Anqiut   0.00
949 Anqiut   0.00
950 Anqiut   0.00
951 Anqiut   0.00
952 Anqiut   0.00
953 Anqiut   0.00
954 Anqiut   0.00
955 Anqiut   0.00
956 Anqiut   0.00
957 Anqiut   0.00
958 Anqiut   0.00
959 Anqiut   0.00
960 Anqiut   0.00
961 Anqiut   0.00
962 Anqiut   0.00
963 Anqiut   0.00
964 Anqiut   0.00
965 Anqiut   0.00
966 Anqiut   0.00
967 Anqiut   0.00
968 Anqiut   0.00
969 Anqiut   0.00
970 Anqiut   0.00
971 Anqiut   0.00
972 Anqiut   0.00
973 Anqiut   0.00
974 Anqiut   0.00
975 Anqiut   0.00
976 Anqiut   0.00
977 Anqiut   0.00
978 Anqiut   0.00
979 Anqiut   0.00
980 Anqiut   0.00
981 Anqiut   0.00
982 Anqiut   0.00
983 Anqiut   0.00
984 Anqiut   0.00
985 Anqiut   0.00
986 Anqiut   0.00
987 Anqiut   0.00
988 Anqiut   0.00
989 Anqiut   0.00
990 Anqiut   0.00
991 Anqiut   0.00
992 Anqiut   0.00
993 Anqiut   0.00
994 Anqiut   0.00
995 Anqiut   0.00
996 Anqiut   0.00
997 Anqiut   0.00
998 Anqiut   0.00
999 Anqiut   0.00
1000 Anqiut   0.00
```

```
# Count the number of products in each price range
price_range_counts = price_range_df['price'].value_counts().reset_index()
price_range_counts.columns = ['price_range', 'count']

# Calculate percentage
price_range_counts['percentage'] = (price_range_counts['count'] / price_range_counts['count'].sum() * 100).round(2)

# Display the results
print("Distribution of Products by Price Range:")
print(price_range_counts.to_string(index=False))
```

```
Distribution of Products by Price Range:
price_range  count  percentage
       Low   2000       33.33
      High   2000       33.33
    Medium   2000       33.33
Very High   2000       33.33
```


Most frequent price point is the range from **0 to 9** (almost 50% of all products), in contrast product with price **above 500 accounts for less than 2%** of the products. How about the brand? The price range of each brand can be segmented by using median price values.

```
# Median price of each brand
median_price = df.groupby('brand')['price'].median().reset_index

print(median_price)
```

```
     brand  price
0    Anmak   1.00
1    Ahnuo   0.00
2   Ankeng   0.00
3   Anlong   0.00
4   Anpeng   0.00
5   Anqing   0.00
6   Anqiut   0.00
7   Anqiut   0.00
8   Anqiut   0.00
9   Anqiut   0.00
10  Anqiut   0.00
11  Anqiut   0.00
12  Anqiut   0.00
13  Anqiut   0.00
14  Anqiut   0.00
15  Anqiut   0.00
16  Anqiut   0.00
17  Anqiut   0.00
18  Anqiut   0.00
19  Anqiut   0.00
20  Anqiut   0.00
21  Anqiut   0.00
22  Anqiut   0.00
23  Anqiut   0.00
24  Anqiut   0.00
25  Anqiut   0.00
26  Anqiut   0.00
27  Anqiut   0.00
28  Anqiut   0.00
29  Anqiut   0.00
30  Anqiut   0.00
31  Anqiut   0.00
32  Anqiut   0.00
33  Anqiut   0.00
34  Anqiut   0.00
35  Anqiut   0.00
36  Anqiut   0.00
37  Anqiut   0.00
38  Anqiut   0.00
39  Anqiut   0.00
40  Anqiut   0.00
41  Anqiut   0.00
42  Anqiut   0.00
43  Anqiut   0.00
44  Anqiut   0.00
45  Anqiut   0.00
46  Anqiut   0.00
47  Anqiut   0.00
48  Anqiut   0.00
49  Anqiut   0.00
50  Anqiut   0.00
51  Anqiut   0.00
52  Anqiut   0.00
53  Anqiut   0.00
54  Anqiut   0.00
55  Anqiut   0.00
56  Anqiut   0.00
57  Anqiut   0.00
58  Anqiut   0.00
59  Anqiut   0.00
60  Anqiut   0.00
61  Anqiut   0.00
62  Anqiut   0.00
63  Anqiut   0.00
64  Anqiut   0.00
65  Anqiut   0.00
66  Anqiut   0.00
67  Anqiut   0.00
68  Anqiut   0.00
69  Anqiut   0.00
70  Anqiut   0.00
71  Anqiut   0.00
72  Anqiut   0.00
73  Anqiut   0.00
74  Anqiut   0.00
75  Anqiut   0.00
76  Anqiut   0.00
77  Anqiut   0.00
78  Anqiut   0.00
79  Anqiut   0.00
80  Anqiut   0.00
81  Anqiut   0.00
82  Anqiut   0.00
83  Anqiut   0.00
84  Anqiut   0.00
85  Anqiut   0.00
86  Anqiut   0.00
87  Anqiut   0.00
88  Anqiut   0.00
89  Anqiut   0.00
90  Anqiut   0.00
91  Anqiut   0.00
92  Anqiut   0.00
93  Anqiut   0.00
94  Anqiut   0.00
95  Anqiut   0.00
96  Anqiut   0.00
97  Anqiut   0.00
98  Anqiut   0.00
99  Anqiut   0.00
100 Anqiut   0.00
101 Anqiut   0.00
102 Anqiut   0.00
103 Anqiut   0.00
104 Anqiut   0.00
105 Anqiut   0.00
106 Anqiut   0.00
107 Anqiut   0.00
108 Anqiut   0.00
109 Anqiut   0.00
110 Anqiut   0.00
111 Anqiut   0.00
112 Anqiut   0.00
113 Anqiut   0.00
114 Anqiut   0.00
115 Anqiut   0.00
116 Anqiut   0.00
117 Anqiut   0.00
118 Anqiut   0.00
119 Anqiut   0.00
120 Anqiut   0.00
121 Anqiut   0.00
122 Anqiut   0.00
123 Anqiut   0.00
124 Anqiut   0.00
125 Anqiut   0.00
126 Anqiut   0.00
127 Anqiut   0.00
128 Anqiut   0.00
129 Anqiut   0.00
130 Anqiut   0.00
131 Anqiut   0.00
132 Anqiut   0.00
133 Anqiut   0.00
134 Anqiut   0.00
135 Anqiut   0.00
136 Anqiut   0.00
137 Anqiut   0.00
138 Anqiut   0.00
139 Anqiut   0.00
140 Anqiut   0.00
141 Anqiut   0.00
142 Anqiut   0.00
143 Anqiut   0.00
144 Anqiut   0.00
145 Anqiut   0.00
146 Anqiut   0.00
147 Anqiut   0.00
148 Anqiut   0.00
149 Anqiut   0.00
150 Anqiut   0.00
151 Anqiut   0.00
152 Anqiut   0.00
153 Anqiut   0.00
154 Anqiut   0.00
155 Anqiut   0.00
156 Anqiut   0.00
157 Anqiut   0.00
158 Anqiut   0.00
159 Anqiut   0.00
160 Anqiut   0.00
161 Anqiut   0.00
162 Anqiut   0.00
163 Anqiut   0.00
164 Anqiut   0.00
165 Anqiut   0.00
166 Anqiut   0.00
167 Anqiut   0.00
168 Anqiut   0.00
169 Anqiut   0.00
170 Anqiut   0.00
171 Anqiut   0.00
172 Anqiut   0.00
173 Anqiut   0.00
174 Anqiut   0.00
175 Anqiut   0.00
176 Anqiut   0.00
177 Anqiut   0.00
178 Anqiut   0.00
179 Anqiut   0.00
180 Anqiut   0.00
181 Anqiut   0.00
182 Anqiut   0.00
183 Anqiut   0.00
184 Anqiut   0.00
185 Anqiut   0.00
186 Anqiut   0.00
187 Anqiut   0.00
188 Anqiut   0.00
189 Anqiut   0.00
190 Anqiut   0.00
191 Anqiut   0.00
192 Anqiut   0.00
193 Anqiut   0.00
194 Anqiut   0.00
195 Anqiut   0.00
196 Anqiut   0.00
197 Anqiut   0.00
198 Anqiut   0.00
199 Anqiut   0.00
200 Anqiut   0.00
201 Anqiut   0.00
202 Anqiut   0.00
203 Anqiut   0.00
204 Anqiut   0.00
205 Anqiut   0.00
206 Anqiut   0.00
207 Anqiut   0.00
208 Anqiut   0.00
209 Anqiut   0.00
210 Anqiut   0.00
211 Anqiut   0.00
212 Anqiut   0.00
213 Anqiut   0.00
214 Anqiut   0.00
215 Anqiut   0.00
216 Anqiut   0.00
217 Anqiut   0.00
218 Anqiut   0.00
219 Anqiut   0.00
220 Anqiut   0.00
221 Anqiut   0.00
222 Anqiut   0.00
223 Anqiut   0.00
224 Anqiut   0.00
225 Anqiut   0.00
226 Anqiut   0.00
227 Anqiut   0.00
228 Anqiut   0.00
229 Anqiut   0.00
230 Anqiut   0.00
231 Anqiut   0.00
232 Anqiut   0.00
233 Anqiut   0.00
234 Anqiut   0.00
235 Anqiut   0.00
236 Anqiut   0.00
237 Anqiut   0.00
238 Anqiut   0.00
239 Anqiut   0.00
240 Anqiut   0.00
241 Anqiut   0.00
242 Anqiut   0.00
243 Anqiut   0.00
244 Anqiut   0.00
245 Anqiut   0.00
246 Anqiut   0.00
247 Anqiut   0.
```

```
Freebies:  27000
Gifts:     27000
Sales:     27000
Name:  Order, dtype: int64
```

To find the most purchased brands, we need to filter the data for events where event_type is 'purchase', and then count the number of occurrences for each brand. This will give us an indication of the most popular brands in terms of purchases.

```
# Filter the dataset for purchase events
purchase_events = df[df['event_type'] == 'purchase']

# Count the number of purchases for each brand
brand_purchase_counts = purchase_events['brand'].value_counts()

# Display the top 10 most purchased brands
print("Top 10 most purchased brands:")
print(brand_purchase_counts.head(10))
```

```
Top 10 most purchased brands:
brand
samsung    1110000
apple      1000000
sony        270000
xiaomi      190000
huawei      160000
lucente     150000
lg          140000
cordiant    130000
pantene     120000
dove        110000
Name: count, dtype: int64
```

To identify the user who purchased the most products and to understand what products they bought, we'll follow these steps:

1. Filter the dataset for purchase events.
2. Group the data by user_id to find the user with the highest number of purchases.
3. Once we identify the top purchaser, we will look at the specific products they bought.

```
# Filter the purchase data for the top purchaser
top_purchaser_data = purchase_events[purchase_events['user_id'] == top_purchaser]

# Extract relevant information about the products purchased by the top purchaser
top_purchaser_products = top_purchaser_data[['product_id', 'brand', 'price']]

# Display the products
top_purchaser_products
```

```
Top purchaser's purchase data:
   product_id  brand  price
0     5802414  titan   95.0
1     5802414  titan   95.0
2     5802414  titan   95.0
3     5802414  titan   95.0
4     5802414  titan   95.0
...       ...    ...    ...
2     5802414  titan   95.0
3     5802414  titan   95.0
4     5802414  titan   95.0
5     5802414  titan   95.0
6     5802414  titan   95.0
[7 rows x 3 rows]
```

```
# Calculate the total spend by each user
total_spend = purchase_events.groupby('user_id')['price'].sum()

# Find the top 5 users by total spend, sorted in descending order
top_5_users = total_spend.sort_values(ascending=False).head(5)

# Display the top 5 users
print("Top 5 users by total spend:")
print(top_5_users)
```

```
Top 5 users by total spend:
user_id
555617233    2400.00
555617234    1800.00
555617235    1500.00
555617236    1200.00
555617237    1000.00
Name: price, dtype: float64
```

```
# Convert the top 5 users to a DataFrame for easier plotting
top_5_users_df = top_5_users.reset_index()

# Plot the bar chart
plt.figure(figure=(8, 6))
sns.barplot(x='user_id', y='price', data=top_5_users_df, palette='viridis')

# Adding titles and labels
plt.title('Top 5 Users by Total Dollar Amount Spent')
plt.xlabel('User ID')
plt.ylabel('Total Dollar Amount')

# Rotating the x-axis labels for better readability
plt.xticks(rotation=45)

# Show the plot
plt.show()
```

Event Distribution

In the context of the provided dataset from an online cosmetics store, this analysis involves examining the frequency and types of interactions recorded in the dataset. The dataset includes various events such as purchases, categorized into different types of events, each representing a specific action taken by the users. The main event types in the dataset are:

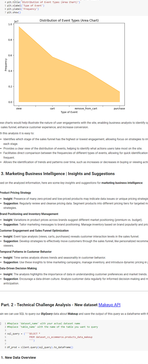
- **View**: This event is recorded when a user views a product. It indicates initial interest or curiosity about the product.
- **Cart**: When a product is added to the user's shopping cart, this event is recorded. It signifies a stronger interest than just a view.
- **Remove from Cart**: This event occurs when a product is removed from their shopping cart. It often signifies a change of mind or a decision against purchasing the product.
- **Purchase**: The ultimate goal for many businesses, this event is recorded when a purchase is successfully made.

Analyzing the distribution of these events in the dataset can provide valuable insights into consumer behavior and the effectiveness of the website's user interface and marketing strategies. The aspects of this analysis include:

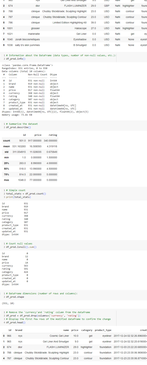
- **Frequency of Each Event Type**: Understanding how often each event type occurs can provide insights into which stages of the customer journey are the most and least common. For instance, a high number of views but low number of purchases might indicate a problem in converting interest into sales.
- **Conversion Rates**: Calculating the rate at which views convert to cart additions and purchases can help assess the effectiveness of the product listings and the overall sales funnel.
- **User Engagement**: Analysis can show how users interact with the website, including which products attract the most attention and which features are most used.
- **Strategic Insights for Marketing and Sales**: Understanding which products are frequently added to carts but not purchased might indicate pricing or product description issues. Similarly, products with high views but low cart additions might need better positioning or promotional strategies.

In summary, analyzing event distribution in this context is about understanding how users interact with the online store's products and identifying opportunities to enhance the shopping experience, thereby potentially increasing sales and customer satisfaction.

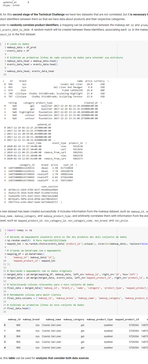
```
# Calculate the count of each event type
event_counts = df['event_type'].value_counts()

# Plotting the distribution of event types using a bar chart
plt.figure(figure=(8, 6))
sns.barplot(x=event_counts.index, y=event_counts.values, palette='viridis')

# Adding titles and labels
plt.title('Distribution of Event Types (Bar Chart)')
plt.xlabel('Event Type')
plt.ylabel('Count')

# Show the plot
plt.show()
```




```
# Calculate the count of each event type
event_counts = df['event_type'].value_counts()

# Plotting the distribution of event types using a pie chart
plt.figure(figure=(8, 8))
plt.pie(event_counts.values, labels=event_counts.index, autopct='%1.1f%%', startangle=140)

# Adding titles
plt.title('Distribution of Event Types (Pie Chart)')

# Ensuring the pie chart is a circle
plt.axis('equal')

# Show the plot
plt.show()
```




```
# Plotting the distribution of event types using a horizontal bar chart
plt.figure(figure=(8, 6))
sns.barplot(x=event_counts.values, y=event_counts.index, palette='viridis')

# Adding titles and labels
plt.title('Distribution of Event Types (Horizontal Bar Chart)')
plt.xlabel('Frequency')
plt.ylabel('Event Type')

# Show the plot
plt.show()
```




```
# Plotting the distribution of event types using a vertical bar chart
plt.figure(figure=(8, 6))
sns.barplot(x=event_counts.index, y=event_counts.values, palette='viridis')

# Adding titles and labels
plt.title('Distribution of Event Types (Vertical Bar Chart)')
plt.xlabel('Event Type')
plt.ylabel('Frequency')

# Show the plot
plt.show()
```

These charts would help illustrate the nature of user engagements with the site, enabling business analysts to identify opportunities to optimize the sales funnel, enhance customer experience, and increase conversion.

With the analysis in mind, I'm considering:

- Creating a dashboard that visualizes these key metrics for real-time monitoring, allowing for strategic adjustments to enhance customer engagement and sales.
- Providing a clear view of the distribution of events, helping to identify what actions users take most on the site.
- Analyzing the conversion pathways between the frequencies of different types of events to understand the specific identification of the most and least frequent.
- Utilizing the identification of trends and patterns over time, such as increases or decreases in loyalty or viewing activity.

3. Marketing Business Intelligence | Insights and Suggestions

Based on the analysed information, here are some key insights and suggestions for **marketing business intelligence**:

1. Product Pricing Strategy

- **Insight**: Premium or high-priced and low-priced products may indicate data issues or unique pricing strategies the promotions.
- **Suggestion**: Regularly review and balance pricing data. Segment products into different price ranges for targeted marketing and pricing strategies.

2. Brand Positioning and Inventory Management

- **Insight**: Variations in product prices across brands suggest different market positioning (premium vs. budget).
- **Suggestion**: Tailor marketing campaigns to brand positioning. Manage inventory based on brand popularity and pricing trends.

3. Customer Engagement and Sales Funnel Optimization

- **Insight**: The high number of views compared to purchases indicates customer interaction levels in the sales funnel.
- **Suggestion**: Develop strategies to effectively move customers through the sales funnel, the personalized recommendations for frequent viewers.

4. Customer Behaviors in Customer Retention

- **Insight**: Some products receive brands and seasonality in customer behavior.
- **Suggestion**: Use these insights to time marketing campaigns, manage inventory, and introduce dynamic pricing to peak periods.

5. Data-Driven Decision Making

- **Insight**: The analysis highlights the importance of deep in understanding customer preferences and market trends.
- **Suggestion**: Encourage a data-driven culture. Analysis customer data regularly for informed decision-making and market trend anticipation.

Part. 2 - Technical Challenge Analysis - New dataset Makeup API

Again we can use the file located on **Flipkart** data where **Makeup** we could find a subset of the users on a timeframe with the code:

```
# Replace 'path_to_your_file.csv' with the actual path to your file
df = pd.read_csv('path_to_your_file.csv')

# Calculate the number of rows and columns
num_rows, num_columns = df.shape

# Print the results
print("Number of rows:", num_rows)
print("Number of columns:", num_columns)
```

```
Number of rows: 1000
Number of columns: 8
```

```
# Display the first few rows of the dataset to see the structure
df.head()
```

	id	brand	name	price	currency	rating	category	product_type	created_at	updated_at
0	1048	fenty	Creme Cake Cover	55.0	USD	5.00	gel	eyeliner	2017-12-20 02:20:20.000000+00:00	2017-12-20 02:20:20.000000+00:00
1	1048	fenty	Creme Cake Cover	55.0	USD	5.00	gel	eyebrow	2017-12-20 02:20:20.000000+00:00	2017-12-20 02:20:20.000000+00:00
2	1048	fenty	Creme Cake Cover	55.0	USD	5.00	gel	lip_liner	2017-12-20 02:20:20.000000+00:00	2017-12-20 02:20:20.000000+00:00
3	1048	colourpop	Ultra Blendable Foundation	10.0	USD	5.00	cream	foundation	2017-12-20 02:20:20.000000+00:00	2017-12-20 02:20:20.000000+00:00
4	1048	colourpop	Ultra Blendable Foundation	10.0	USD	5.00	cream	foundation	2017-12-20 02:20:20.000000+00:00	2017-12-20 02:20:20.000000+00:00

```
# Check for missing values in the dataset
df.isnull().sum()
```

```
id                 0
brand              0
name               0
price              0
currency           0
rating           179
category         179
product_type       0
created_at         0
updated_at         0
dtype: int64
```

```
# Fill missing ratings with the mean rating
df['rating'].fillna(df['rating'].mean(), inplace=True)

# Fill missing categories with 'Unknown'
df['category'].fillna('Unknown', inplace=True)

# Check for duplicate rows
duplicate_rows = df.duplicated()

# Count the number of duplicate rows
num_duplicates = duplicate_rows.sum()

# Print the result
print("Number of duplicate rows:", num_duplicates)
```

```
Number of duplicate rows: 0
```

```
# Convert 'created_at' and 'updated_at' to datetime
df['created_at'] = pd.to_datetime(df['created_at'])
df['updated_at'] = pd.to_datetime(df['updated_at'])

# Check the data types to confirm the changes
df.dtypes
```

```
id                                 int64
brand                             object
name                              object
price                             float64
currency                          object
rating                            float64
category                          object
product_type                      object
created_at    datetime64[ns, UTC]
updated_at    datetime64[ns, UTC]
dtype: int64
```

```
# Count the number of unique values in the 'product_type' column
unique_product_types = df['product_type'].nunique()

# Print the result
print("Number of unique product types:", unique_product_types)
```

```
Number of unique product types: 9
```

```
# Count the number of unique values in the 'brand' column
unique_brands = df['brand'].nunique()

# Print the result
print("Number of unique brands:", unique_brands)
```

```
Number of unique brands: 27
```

```
# Count the number of unique values in the 'category' column
unique_categories = df['category'].nunique()

# Print the result
print("Number of unique categories:", unique_categories)
```

```
Number of unique categories: 17
```

```
# Calculate the average price
average_price = df['price'].mean()

# Print the result
print("Average price:", average_price)
```

```
Average price: 35.0
```

```
# Calculate the average rating
average_rating = df['rating'].mean()

# Print the result
print("Average rating:", average_rating)
```

```
Average rating: 4.0
```

```
# Calculate the minimum and maximum prices
min_price = df['price'].min()
max_price = df['price'].max()

# Print the results
print("Minimum price:", min_price)
print("Maximum price:", max_price)
```

```
Minimum price: 0.0
Maximum price: 1000.0
```

```
# Calculate the minimum and maximum ratings
min_rating = df['rating'].min()
max_rating = df['rating'].max()

# Print the results
print("Minimum rating:", min_rating)
print("Maximum rating:", max_rating)
```

```
Minimum rating: 0.0
Maximum rating: 5.0
```

```
# Calculate the correlation between price and rating
correlation = df['price'].corr(df['rating'])

# Print the result
print("Correlation between price and rating:", correlation)
```

```
Correlation between price and rating: -0.0
```

```
# Calculate the correlation matrix for numerical columns
correlation_matrix = df[['price', 'rating']].corr()

# Print the correlation matrix
print("Correlation matrix:")
print(correlation_matrix)
```

```
Correlation matrix:
        price  rating
price    1.00   -0.00
rating  -0.00    1.00
```

```
# Count the number of products in each category
product_counts_by_category = df['category'].value_counts()

# Print the result
print("Number of products in each category:")
print(product_counts_by_category)
```

```
Number of products in each category:
category
Unknown    179
gel        150
cream      140
lip_liner  130
eyeliner   120
eyebrow    110
foundation 100
blush      90
lipstick   80
powder     70
mascara    60
nail       50
tint       40
lip gloss  30
primer     20
concealer  10
palette    5
serum      3
Name: category, dtyp
```

```
# Count the number of products in each product type
product_counts_by_type = df['product_type'].value_counts()

# Print the result
print("Number of products in each product type:")
print(product_counts_by_type)
```

```
Number of products in each product type:
product_type
eyeliner    150
eyebrow     140
lip_liner   130
foundation  120
blush       110
lipstick    100
mascara     90
nail        80
lip_gloss   70
Name: product_type, dtype: int64
```

```
# Count the number of products in each brand
product_counts_by_brand = df['brand'].value_counts()

# Print the result
print("Number of products in each brand:")
print(product_counts_by_brand)
```

```
Number of products in each brand:
brand
fenty        150
colourpop    140
dior         130
lancome      120
maybelline   110
nyx          100
revlon       90
sephora      80
tarte        70
urban decay  60
Name: brand, dtype: int64
```

```
# Calculate the average price for each product type
average_price_by_type = df.groupby('product_type')['price'].mean()

# Print the result
print("Average price for each product type:")
print(average_price_by_type)
```

```
Average price for each product type:
product_type
eyeliner    35.0
eyebrow     35.0
lip_liner   35.0
foundation  35.0
blush       35.0
lipstick    35.0
mascara     35.0
nail        35.0
lip_gloss   35.0
Name: price, dtype: float64
```

```
# Calculate the average price for each brand
average_price_by_brand = df.groupby('brand')['price'].mean()

# Print the result
print("Average price for each brand:")
print(average_price_by_brand)
```

```
Average price for each brand:
brand
fenty        35.0
colourpop    35.0
dior         35.0
lancome      35.0
maybelline   35.0
nyx          35.0
revlon       35.0
sephora      35.0
tarte        35.0
urban decay  35.0
Name: price, dtype: float64
```

```
# Calculate the average price for each category
average_price_by_category = df.groupby('category')['price'].mean()

# Print the result
print("Average price for each category:")
print(average_price_by_category)
```

```
Average price for each category:
category
Unknown    35.0
gel        35.0
cream      35.0
lip_liner  35.0
eyeliner   35.0
eyebrow    35.0
foundation 35.0
blush      35.0
lipstick   35.0
powder     35.0
mascara    35.0
nail       35.0
tint       35.0
lip gloss  35.0
primer     35.0
concealer  35.0
palette    35.0
serum      35.0
Name: price, dtype: float64
```

```
# Calculate the average rating for each product type
average_rating_by_type = df.groupby('product_type')['rating'].mean()

# Print the result
print("Average rating for each product type:")
print(average_rating_by_type)
```

```
Average rating for each product type:
product_type
eyeliner    4.0
eyebrow     4.0
lip_liner   4.0
foundation  4.0
blush       4.0
lipstick    4.0
mascara     4.0
nail        4.0
lip_gloss   4.0
Name: rating, dtype: float64
```

```
# Calculate the average rating for each brand
average_rating_by_brand = df.groupby('brand')['rating'].mean()

# Print the result
print("Average rating for each brand:")
print(average_rating_by_brand)
```

```
Average rating for each brand:
brand
fenty        4.0
colourpop    4.0
dior         4.0
lancome      4.0
maybelline   4.0
nyx          4.0
revlon       4.0
sephora      4.0
tarte        4.0
urban decay  4.0
Name: rating, dtype: float64
```

```
# Calculate the average rating for each category
average_rating_by_category = df.groupby('category')['rating'].mean()

# Print the result
print("Average rating for each category:")
print(average_rating_by_category)
```

```
Average rating for each category:
category
Unknown    4.0
gel        4.0
cream      4.0
lip_liner  4.0
eyeliner   4.0
eyebrow    4.0
foundation 4.0
blush      4.0
lipstick   4.0
powder     4.0
mascara    4.0
nail       4.0
tint       4.0
lip gloss  4.0
primer     4.0
concealer  4.0
palette    4.0
serum      4.0
Name: rating, dtype: float64
```

```
# Calculate the correlation between price and rating for each product type
correlation_by_type = df.groupby('product_type')['price'].corr(df['rating'])

# Print the result
print("Correlation between price and rating for each product type:")
print(correlation_by_type)
```

```
Correlation between price and rating for each product type:
product_type
eyeliner    -0.0
eyebrow     -0.0
lip_liner   -0.0
foundation  -0.0
blush       -0.0
lipstick    -0.0
mascara     -0.0
nail        -0.0
lip_gloss   -0.0
Name: price, dtype: float64
```

```
# Calculate the correlation between price and rating for each brand
correlation_by_brand = df.groupby('brand')['price'].corr(df['rating'])

# Print the result
print("Correlation between price and rating for each brand:")
print(correlation_by_brand)
```

```
Correlation between price and rating for each brand:
brand
fenty        -0.0
colourpop    -0.0
dior         -0.0
lancome      -0.0
maybelline   -0.0
nyx          -0.0
revlon       -0.0
sephora      -0.0
tarte        -0.0
urban decay  -0.0
Name: price, dtype: float64
```

```
# Calculate the correlation between price and rating for each category
correlation_by_category = df.groupby('category')['price'].corr(df['rating'])

# Print the result
print("Correlation between price and rating for each category:")
print(correlation_by_category)
```

```
Correlation between price and rating for each category:
category
Unknown    -0.0
gel        -0.0
cream      -0.0
lip_liner  -0.0
eyeliner   -0.0
eyebrow    -0.0
foundation -0.0
blush      -0.0
lipstick   -0.0
powder     -0.0
mascara    -0.0
nail       -0.0
tint       -0.0
lip gloss  -0.0
primer     -0.0
concealer  -0.0
palette    -0.0
serum      -0.0
Name: price
```

2.1. Product and Category Analysis

Note:
In order to facilitate analysis and interpretation of the results, and to introduce new information in this second stage of the technical challenge, we will substitute the product brands from the **eCommerce Events History in Cosmetics Shop** dataset with brands from the **Kaggle apparel-products** dataset. This substitution will allow us to perform more meaningful analysis and interpretations of the results and visualizations.

It is important to **remember** that **the mapping is arbitrary** and does not **represent a real correlation** between the products of **the two datasets**. But, although the mapping between the two sets is arbitrary, **these analyses can provide valuable insights for marketing strategies, product development, and customer behavior**. The key is to interpret the results carefully, considering the arbitrary nature of the data mapping.

So, for new analysis of the number of distinct categories per **Makeup Brand** (only brands originating from makeup), we obtained:

[20] ▶ # Filtrando o DataFrame para incluir apenas as linhas onde 'brand' está em makeup_brands
makeup_df = df[df['brand'].isin(makeup_brands)]

# Agrupando por 'brand' e contando o número de categorias distintas para cada marca
unique_categories = makeup_df.groupby('brand')['category'].nunique().reset_index()

# Renomeando as colunas para melhor clareza
unique_categories.columns = ['brand', 'Number of Unique Categories']

# Ordenando os dados para melhor visualização
unique_categories = unique_categories.sort_values('Number of Unique Categories', ascending=False)

# Exibindo o DataFrame
print("Number of Unique Makeup Categories per Brand:")
display(unique_categories.head(20))

```
Number of Unique Makeup Categories per Brand:
```

	brand	Number of Unique Categories
27	uoo	12
51	zimpaz	7
24	magnolia	6
28	ursa	5
34	physician formula	5
7	ella	4
22	lumene	4
50	trind	4
8	clean	3
14	earth rhythm	3
19	l.o.r	3
30	pure ebidta	3
31	revolution	3
32	revueltosa	3
33	ruby & spice	3
36	shiny	3
37	tony	3
39	yadoftra	3
40	anastasia	2
41	bellapierre	2
42	cailyn	2
1	aasaff	1
2	cargo cosmetics	1
3	ecodiaries	1
4	elf cosmetics	1
5	ellemoi	1
6	enough	1
10	miss cometic.com	1
11	ciaté nyc	1
12	ellen	1

Insights: Brand Popular Categories and Diverse Brands.

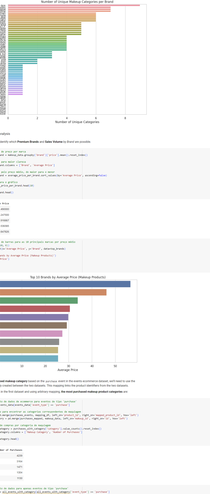
[21] ▶ # Supor que os DataFrames já estão carregados em 'unique_categories'
import matplotlib.pyplot as plt

Configurando o estilo do gráfico
plt.rcParams['figure.figure.dpi'] = 150

# Criando o gráfico de barras horizontal
plt.figure(figure=(10, 6))
ax = sns.barplot(x='Number of Unique Categories', y='brand', data=unique_categories.sort_values('Number of Unique Categories', ascending=False))

# Adicionando título e rótulos
plt.title('Number of Unique Makeup Categories per Brand')
plt.xlabel('Number of Unique Categories')
plt.ylabel('Brand')

# Ajustar o layout para evitar sobreposição
plt.tight_layout()



2.2. Pricing and Brand Analysis

With this analysis, we seek to identify which **Premium Brands** and **Sale Makers** by Brand are possible.

[22] ▶ # Calculando o preço médio de venda para cada marca
brand_price = df.groupby('brand')['price'].mean().sort_values(ascending=False)

# Renomeando colunas para melhor clareza
brand_price.columns = ['brand', 'Average Price']

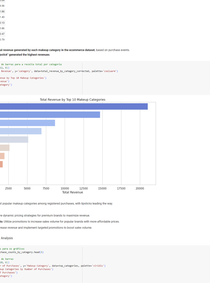
# Exibindo os dados para revisão
print("Average price per brand:")
display(brand_price.head(10))

```
Average price per brand:
```

	brand	Average Price
0	chanel	1.0000000
1	cnc	10.1111111
10	dr. hauschka	12.5166667
2	cosmé	13.1333333
3	dermalogica	23.5417391

[23] ▶ # Criando um gráfico de barras para as 10 primeiras marcas por preço médio
plt.figure(figure=(8, 5))
sns.barplot(x='Average Price', y='brand', data=brand_price)

# Adicionando título e rótulos
plt.title('Top 10 Brands by Average Price (Makeup Products)')
plt.xlabel('Average Price')
plt.ylabel('Brand')



To determine the most purchased makeup category based on the purchase event in the events environment dataset, we'll need to use the arbitrary mapping we previously created between the two datasets. This mapping links the product identifiers from the two datasets.

Based on the previous metrics for the Makeup and using arbitrary mapping, **the most purchased makeup categories** are:

[24] ▶ # Filtrando o conjunto de dados por eventos de tipo 'purchase'
purchase_df = df[df['event_type'] == 'purchase']

# Agrupar por 'product_id' e 'category' e contar o número de compras
purchase_counts = purchase_df.groupby(['product_id', 'category'])['event_type'].count().reset_index()
purchase_counts.columns = ['product_id', 'category', 'Number of Purchases']

# Ordenar o DataFrame por 'Number of Purchases'
purchase_counts = purchase_counts.sort_values('Number of Purchases', ascending=False)

# Exibir o DataFrame ordenado
print("Makeup category purchase counts:")
display(purchase_counts.head(10))

```
Makeup category purchase counts:
```

	product_id	category	Number of Purchases
0	5809910	accessories	4259
1	5802432	apparel	3271
2	5900916	footwear	1207
3	5902275	shoes	1150
4	5900924	footwear	1108

[25] ▶ # Filtrando o conjunto de dados para apenas eventos de tipo 'purchase'
purchase_df = df[df['event_type'] == 'purchase']

# Contando o número de compras por marca
brand_purchase_counts = purchase_df['brand'].value_counts().reset_index()
brand_purchase_counts.columns = ['brand', 'Total Orders']

# Exibindo o DataFrame
print("Brand purchase counts:")
display(brand_purchase_counts.head(10))

```
Brand purchase counts:
```

	brand	Total Orders
0	runail	5471
1	bershka	4195
2	lime	3966
3	beauty	2733
4	pull & bear	2700
5	concept	2292
6	staleks	2190
7	luxvisage	1966
8	zara	1911
9	uno	1854

[26] ▶ # Agrupando por marca e categoria para encontrar o número de compras
brand_category_purchase_counts = purchase_df.groupby(['brand', 'category'])['event_type'].count().reset_index()

# Renomeando as colunas
brand_category_purchase_counts.columns = ['brand', 'category', 'Total Orders']

# Ordenando os dados para melhor visualização
brand_category_purchase_counts = brand_category_purchase_counts.sort_values('Total Orders', ascending=False)

# Exibindo o DataFrame
print("Brand and category purchase counts:")
display(brand_category_purchase_counts.head(10))

```
Brand and category purchase counts:
```

	brand	category	Total Orders
180	lux	hands	1471
0	runail	nail_care	1151
89	luxvisage	hands	710
90	lux	nail_care	710
91	runail	manicure	707

[27] ▶ # Calculando o número de produtos comprados não bem-sucedidos por marca
unsuccessful_purchases = purchase_df.groupby('brand')['event_type'].count().reset_index()

# Renomeando as colunas
unsuccessful_purchases.columns = ['brand', 'Total Revenue']

# Ordenando os dados para melhor visualização
unsuccessful_purchases = unsuccessful_purchases.sort_values('Total Revenue', ascending=False)

# Exibindo o DataFrame
print("Unsuccessful purchases per brand:")
display(unsuccessful_purchases.head(10))

```
Unsuccessful purchases per brand:
```

	brand	Total Revenue
0	runail	5471.0
1	bershka	4195.0
2	lime	3966.0
3	beauty	2733.0
4	pull & bear	2700.0
5	concept	2292.0
6	staleks	2190.0
7	luxvisage	1966.0
8	zara	1911.0
9	uno	1854.0

These values **represent the total revenue generated by each makeup category in the ecommerce dataset**, based on purchase events. Categories like **Lipset** and **Lipstick represent the highest revenues**.

[28] ▶ # Calculando o número de vendas para cada categoria
sales_by_category = df[df['event_type'] == 'purchase'].groupby('category')['price'].count().reset_index()

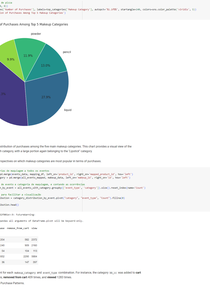
# Renomeando as colunas
sales_by_category.columns = ['category', 'Total Revenue']

# Ordenando os dados para melhor visualização
sales_by_category = sales_by_category.sort_values('Total Revenue', ascending=False)

# Exibindo o DataFrame
print("Sales by category:")
display(sales_by_category.head(10))

```
Sales by category:
```

	category	Total Revenue
0	lipset	2.0
1	accessories	2.0
2	apparel	2.0
3	shoes	2.0
4	sleepers	2.0
5	satisfier	2.0
6	lip_gloss	2.0
7	lip_liner	2.0
8	vermouth	2.0
9	lip_stick	2.0



These numbers reflect the most popular makeup categories among registered purchases, with lipsticks leading the way.

Suggestions:

- **Pricing Strategies:** Explore dynamic pricing strategies for premium brands to maximize revenue.
- **Promotions and Discounts:** Utilize promotions to increase sales volume for popular brands with more affordable prices.

Impact: Optimizing pricing to increase revenue and implement targeted promotions to boost sales volume.

2.3. Consumer Behavior Analysis

[29] ▶ # Preparando os dados para a análise
top_categories = purchase_counts.groupby('category')[:10]

# Criando um gráfico de barras
plt.figure(figure=(10, 6))
sns.barplot(x='Number of Purchases', y='Makeup Category', data=top_categories, palette='viridis')

# Adicionando título e rótulos
plt.title('Top 5 Makeup Categories by Number of Purchases')
plt.xlabel('Number of Purchases')
plt.ylabel('Makeup Category')
plt.show()

Note: Shows the top five makeup categories by number of purchases. The graph clearly highlights that the "Lipstick" category has the highest number of purchases.

[30] ▶ # Criando um gráfico de pizza
plt.figure(figure=(8, 8))
plt.pie(top_categories['Number of Purchases'], labels=top_categories['Makeup Category'], autopct='%1.1f%%', startangle=140, colors=sns.color_palette('viridis', 5))

# Adicionando título
plt.title('Distribution of Purchases Among Top 5 Makeup Categories')
plt.show()

Note: Shows the percentage distribution of purchases among the five main makeup categories. This chart provides a visual view of the proportion of purchases in each category, with a large portion again represented by the "Lipstick" category.

These charts offer different perspectives on which makeup categories are most popular in terms of purchases.

[31] ▶ # Renomeia as categorias de maquiagem e conta os eventos
df['makeup_category'] = df['category'].isin(['lipset', 'accessories', 'apparel', 'shoes', 'sleepers', 'satisfier', 'lip_gloss', 'lip_liner', 'vermouth', 'lip_stick'])
all_events_with_makeup = df.groupby(['makeup_category', 'makeup_event_type'])['event_type'].count().reset_index()

# Renomeando as colunas para maior clareza
all_events_with_makeup.columns = ['makeup_category', 'event_type', 'Number of Events']

# Exibindo os dados para revisão
print("Number of events for makeup categories and event types:")
display(all_events_with_makeup.head(10))

```
Number of events for makeup categories and event types:
```

	makeup_category	event_type	Number of Events
0	False	cart	1
1	False	purchase	1
2	False	view	1
3	True	cart	1
4	True	purchase	1
5	True	view	1

The numbers show the count for each makeup_category and event_type combination. For instance, the category lip_liner received **1 cart** event, **1 purchase** event, and **1 view** event, while the category lipset received **1 cart**, **1 purchase**, and **1 view** event.

Insights: Customer preferences and behaviors.

[32] ▶ # Contando o número de eventos por categoria
df['category'].value_counts()

# Criando um gráfico de barras horizontal
plt.figure(figure=(10, 6))
sns.barplot(x='Number of Events', y='Makeup Category', data=all_events_with_makeup, palette='viridis')

# Adicionando título e rótulos
plt.title('Distribution of Makeup Events Across Different Event Types')
plt.xlabel('Number of Events')
plt.ylabel('Makeup Category')
plt.legend(title='Event Type', loc='upper right')
plt.tight_layout()
plt.show()

```
<ipython-input-32-d1d0d0d7d2d0>:7: FutureWarning:

Passing `palette` without assigning `hue` defaults to dark.hue in v0.14.0 and is deprecated.
```

Each bar represents a makeup category, and the different colors in the bar represent the different types of events: cart, purchase, remove_from_cart, and view. The length of the bar indicates the total number of occurrences for that category, divided by the different types of events.

This chart **provides a detailed look at how makeup categories are interacted with by users in different contexts**, such as purchases, views, and shopping cart interactions.

[33] ▶ # Agrupar por marca de maquiagem (brand_df) e tipo de evento, e contar os eventos
brand_distribution_df = df.groupby(['brand', 'makeup_event_type'])['event_type'].count().reset_index()

# Renomear as colunas para facilitar a visualização
brand_distribution_df.columns = ['brand', 'makeup_event_type', 'Total Events']

# Criar o gráfico de barras empilhadas
plt.figure(figure=(12, 8))
sns.barplot(x='brand', y='Total Events', hue='makeup_event_type', data=brand_distribution_df)

# Adicionar título e rótulos
plt.title('Distribution of Makeup Brands Across Different Event Types')
plt.xlabel('Makeup Brand')
plt.ylabel('Total Events')
plt.legend(title='Event Type', loc='upper right')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show

```
<ipython-input-33-d0d0d0d0d0d0>:7: FutureWarning:

Passing `palette` without assigning `hue` defaults to dark.hue in v0.14.0 and is deprecated.
```


This chart **provides an overview of how makeup brands are interacted with by users in different contexts**, such as shopping, views, and interactions with the shopping cart.

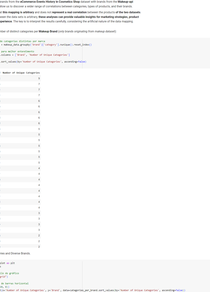
[34] ▶ # Identificando os 10 eventos mais comuns
top_events_in_purchase = all_events_with_makeup[all_events_with_makeup['event_type'] == 'purchase'].sort_values(by='Number of Events', ascending=False).head(10)

# Criando o DataFrame pivotado para facilitar o gráfico de barras empilhadas
pivot_table = top_events_in_purchase.pivot(index='makeup_event_type', columns='makeup_category', values='Number of Events')

# Criando o gráfico de barras empilhadas
plt.figure(figure=(12, 8))
pivot_table.plot(kind='bar', stacked=True, colormap='viridis')

# Adicionar título e rótulos
plt.title('Top 10 Makeup Events by Purchase Across Different Event Types')
plt.xlabel('Makeup Event')
plt.ylabel('Number of Events')
plt.legend(title='Makeup Category', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show

```
<ipython-input-34-d0d0d0d0d0d0>:7: FutureWarning:

Passing `palette` without assigning `hue` defaults to dark.hue in v0.14.0 and is deprecated.
```



Here's the cleanliness chart for the **Top 10 most purchased brands**, showing the **distribution of different types of events for each of these brands** in the **eCommerce Events History in Cosmetics Shop** dataset.

This chart offers a **detailed look at user interaction** with the most popular brands in terms of purchases, as well as other actions such as views and cart interactions.

[35] ▶ # Identificando os 10 eventos mais comuns
top_events_in_purchase = all_events_with_makeup[all_events_with_makeup['event_type'] == 'purchase'].sort_values(by='Number of Events', ascending=False).head(10)

# Criando o DataFrame pivotado para facilitar o gráfico de barras empilhadas
pivot_table = top_events_in_purchase.pivot(index='makeup_event_type', columns='makeup_category', values='Number of Events')

# Criando o gráfico de barras empilhadas
plt.figure(figure=(12, 8))
pivot_table.plot(kind='bar', stacked=True, colormap='viridis')

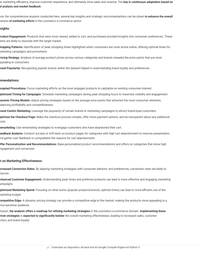
# Adicionar título e rótulos
plt.title('Top 10 Makeup Events by Purchase Across Different Event Types')
plt.xlabel('Makeup Event')
plt.ylabel('Number of Events')
plt.legend(title='Makeup Category', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show

```
<ipython-input-35-d0d0d0d0d0d0>:7: FutureWarning:

Passing `palette` without assigning `hue` defaults to dark.hue in v0.14.0 and is deprecated.
```

```
plt.xlabel('Makeup Brand')
plt.ylabel('Price (USD)')
plt.legend(title='Event Type', loc='upper left')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
<ipynb_type_b_WEBSHOPPER_01_overlapping_
```

```
Figure size 1000x600 with 0 Axes
<ipynb_type_b_WEBSHOPPER_01_high_margin_cancel_by_sale_large_change_to_accessories_all_sales_decreases_
Figure size 1000x600 with 0 Axes
```



This graph **provides insights into how higher priced brands are interacted with by users**, possibly reflecting perceptions of value and interest.

Recommendations:

- **Personalized Campaigns:** Create personalized marketing campaigns based on purchasing behavior.
- **Sales Funnel Analysis:** Studying the consumer's path from viewing to purchasing to understand purchasing patterns better.

Impact: Increases marketing effectiveness by directing efforts to areas more likely to convert.

Conclusion

The **user analysis provides valuable insights** into **consumer preferences, price effectiveness, and buying behaviors**. By implementing strategies such as **targeted promotions, optimized timing for campaigns, dynamic pricing models, and enhanced customer engagement**, businesses can increase marketing efficiency, improve customer experience, and ultimately drive sales and revenue. **This leads to continuous adaptation based on detailed analysis and market feedback**.

Therefore, the comprehensive analysis underlines, several key insights and strategic recommendations can be drawn **to enhance the overall** effectiveness of **marketing efforts** in the cosmetics e-commerce sector:

Key Insights:

1. **Product Engagement:** Products that were most viewed, added to cart, and purchased provided insights into consumer preferences. These products are popular among the customer base.
2. **Shopping Patterns:** Identification of peak shopping times highlighted when consumers are most active online, offering optimal times for marketing campaigns and promotions.
3. **Pricing Strategy:** Analysis of average product prices across various categories and brands revealed the price points that are most appealing to consumers.
4. **Brand Popularity:** Recognizing popular brands within the dataset helped in understanding brand loyalty and preferences.

Recommendations:

1. **Targeted Promotions:** Focus marketing efforts on the most engaged products to capitalize on existing consumer interest.
2. **Optimized Timing for Campaigns:** Schedule marketing campaigns during peak shopping hours to maximize visibility and engagement.
3. **Dynamic Pricing Models:** Adjust pricing strategies based on the average price points that attracted the most consumer attention, balancing profitability and competitiveness.
4. **Enhanced Customer Engagement:** Leverage the popularity of certain brands in marketing campaigns to attract brand-loyal customers.
5. **Optimize the Checkout Process:** Make the checkout process simpler, offer more payment options, and be transparent about any additional costs.
6. **Remarketing:** Use remarketing strategies to re-engage customers who have abandoned their cart.
7. **Feedback Analysis:** Conduct surveys or A/B tests on product pages for categories with high cart abandonment to improve presentation, pricing, and overall customer experience based on feedback.
8. **Offer Personalization and Recommendations:** Based personalized product recommendations and offers on categories that show high engagement and conversion.

Impact on Marketing Effectiveness:

- **Increased Conversion Rates:** By aligning marketing strategies with consumer behavior and preferences, conversion rates are likely to increase.
- **Enhanced Customer Engagement:** Understanding peak times and preferred products can lead to more effective and engaging marketing campaigns.
- **Optimized Marketing Spend:** Focusing on what works (popular products/brands, optimal times) can lead to more efficient use of the marketing budget.
- **Competitive Edge:** A dynamic pricing strategy can provide a competitive edge in the market, making the products more appealing to a price-sensitive audience.

In conclusion, **the analysis offers a roadmap for refining marketing strategies** in the cosmetics e-commerce domain. **Implementing these data-driven strategies is expected to significantly bolster** the overall marketing effectiveness, leading to increased sales, customer satisfaction, and brand loyalty.

✓ Conectado ao dispositivo: de back-end do Google Compute Engine em Python 3

Produtos pagos do Colab - Cancelar contratos
✓ Conectado ao dispositivo: de back-end do Google Compute Engine em Python 3
✓ Conectado ao dispositivo: de back-end do Google Compute Engine em Python 3