# Sales Analytics

In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
```

In [2]:
```
#importing the dataset
df1=pd.read_csv("Sales_Data.csv")
df2=pd.read_csv("Glance_Views.csv")
```

## Columns in Sales Data

- SKU_NAME: Unique identifier for the product.
- FEED_DATE: Date of the sales record
- CATEGORY: Broad category of the product.
- SUB_CATEGORY: More specific classification within the category.
- ORDERED_REVENUE: Total revenue generated from orders.
- ORDERED_UNITS: Number of units sold.
- REP_OOS: Reported Out of Stock instances.

## Columns in Glance Views

- SKU_NAME: Unique identifier for the product.
- FEED_DATE: Date of the record.
- VIEWS: Number of product page views.
- UNITS: Number of units sold based on views

In [3]:
```
df1
```

Out[3]:

| | SKU_NAME | FEED_DATE | CATEGORY | SUB_CATEGORY | ORDERED_REVENUE | ORDERED_UNITS | REP_OOS |
|---|---|---|---|---|---|---|---|
| **0** | B12020KBUI | 5/18/2019 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 |
| **1** | B12020KBUI | 5/19/2019 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 |
| **2** | B12020KBUI | 5/22/2019 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 |
| **3** | B12020KBUI | 5/23/2019 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 |
| **4** | B12020KBUI | 5/27/2019 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **41110** | D29S5IMRDI | 8/27/2019 | 1000 Inputs | 1007 Other Input Devices | 129.99 | 1 | 0.0 |
| **41111** | D29S5IMRDI | 8/28/2019 | 1000 Inputs | 1007 Other Input Devices | 259.98 | 2 | 0.0 |
| **41112** | D29S5IMRDI | 8/29/2019 | 1000 Inputs | 1007 Other Input Devices | 259.98 | 2 | 0.0 |
| **41113** | D29S5IMRDI | 8/30/2019 | 1000 Inputs | 1007 Other Input Devices | 259.98 | 2 | 0.0 |
| **41114** | D29S5IMRDI | 8/31/2019 | 1000 Inputs | 1007 Other Input Devices | 0.00 | 0 | 0.0 |

41115 rows × 7 columns

In [4]:
```
df2
```

Out[4]:

| | SKU_NAME | FEED_DATE | VIEWS | UNITS |
|---|---|---|---|---|
| **0** | B1212:PZ:V | 5/1/2019 | 455 | 16 |
| **1** | B1212:PZ:V | 5/2/2019 | 478 | 12 |
| **2** | B1212:PZ:V | 5/3/2019 | 681 | 42 |
| **3** | B1212:PZ:V | 5/4/2019 | 662 | 70 |
| **4** | B1212:PZ:V | 5/5/2019 | 568 | 33 |
| **...** | ... | ... | ... | ... |
| **40740** | C08N8KVJDZ | 8/27/2019 | 225 | -1 |
| **40741** | C08N8KVJDZ | 8/28/2019 | 219 | 0 |
| **40742** | C08N8KVJDZ | 8/29/2019 | 264 | 0 |
| **40743** | C08N8KVJDZ | 8/30/2019 | 260 | 8 |
| **40744** | C08N8KVJDZ | 8/31/2019 | 254 | 3 |

40745 rows × 4 columns

## Descriptive Statistics

In [5]:
```
#shape of the data
df1.shape
```

Out[5]: (41115, 7)

In [6]:
```
df2.shape
```

Out[6]: (40745, 4)

In [7]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41115 entries, 0 to 41114
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   SKU_NAME         41115 non-null  object 
 1   FEED_DATE        41115 non-null  object 
 2   CATEGORY         41115 non-null  object 
 3   SUB_CATEGORY     41115 non-null  object 
 4   ORDERED_REVENUE  41115 non-null  float64
 5   ORDERED_UNITS    41115 non-null  int64  
 6   REP_OOS          40426 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 2.2+ MB
```

There are null values in REP_OOS column

In [8]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40745 entries, 0 to 40744
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   SKU_NAME   40745 non-null  object
 1   FEED_DATE  40745 non-null  object
 2   VIEWS      40745 non-null  int64 
 3   UNITS      40745 non-null  int64 
dtypes: int64(2), object(2)
memory usage: 1.2+ MB
```

## Convert Feed Date to datetime format

In [9]:
```
df1['FEED_DATE'] = pd.to_datetime(df1['FEED_DATE'])
df2['FEED_DATE'] = pd.to_datetime(df2['FEED_DATE'])
```

In [10]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41115 entries, 0 to 41114
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   SKU_NAME         41115 non-null  object
 1   FEED_DATE        41115 non-null  datetime64[ns]
 2   CATEGORY         41115 non-null  object
 3   SUB_CATEGORY     41115 non-null  object
 4   ORDERED_REVENUE  41115 non-null  float64
 5   ORDERED_UNITS    41115 non-null  int64
 6   REP_OOS          40426 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 2.2+ MB
```

In [11]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40745 entries, 0 to 40744
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   SKU_NAME   40745 non-null  object
 1   FEED_DATE  40745 non-null  datetime64[ns]
 2   VIEWS      40745 non-null  int64
 3   UNITS      40745 non-null  int64
dtypes: datetime64[ns](1), int64(2), object(1)
memory usage: 1.2+ MB
```

In [12]:
```
for i in df1.columns:
    print('Data type of the column {} is {}'.format(i,df1[i].dtype))
    print('Number of unique values in the column {} is {}'.format(i,df1[i].nunique()))
    print('Null values in the column {} is {}'.format(i,df1[i].isnull().sum()))
    print('='*90)
```

```
Data type of the column SKU_NAME is object
Number of unique values in the column SKU_NAME is 465
Null values in the column SKU_NAME is 0
==========================================================================================
Data type of the column FEED_DATE is datetime64[ns]
Number of unique values in the column FEED_DATE is 123
Null values in the column FEED_DATE is 0
==========================================================================================
Data type of the column CATEGORY is object
Number of unique values in the column CATEGORY is 10
Null values in the column CATEGORY is 0
==========================================================================================
Data type of the column SUB_CATEGORY is object
Number of unique values in the column SUB_CATEGORY is 24
Null values in the column SUB_CATEGORY is 0
==========================================================================================
Data type of the column ORDERED_REVENUE is float64
Number of unique values in the column ORDERED_REVENUE is 15506
Null values in the column ORDERED_REVENUE is 0
==========================================================================================
Data type of the column ORDERED_UNITS is int64
Number of unique values in the column ORDERED_UNITS is 1011
Null values in the column ORDERED_UNITS is 0
==========================================================================================
Data type of the column REP_OOS is float64
Number of unique values in the column REP_OOS is 3388
Null values in the column REP_OOS is 689
==========================================================================================
```

In [13]:
```
for i in df2.columns:
    print('Data type of the column {} is {}'.format(i,df2[i].dtype))
    print('Number of unique values in the column {} is {}'.format(i,df2[i].nunique()))
    print('Null values in the column {} is {}'.format(i,df2[i].isnull().sum()))
    print('='*90)
```

```
Data type of the column SKU_NAME is object
Number of unique values in the column SKU_NAME is 452
Null values in the column SKU_NAME is 0
==========================================================================================
Data type of the column FEED_DATE is datetime64[ns]
Number of unique values in the column FEED_DATE is 123
Null values in the column FEED_DATE is 0
==========================================================================================
Data type of the column VIEWS is int64
Number of unique values in the column VIEWS is 4370
Null values in the column VIEWS is 0
==========================================================================================
Data type of the column UNITS is int64
Number of unique values in the column UNITS is 1011
Null values in the column UNITS is 0
==========================================================================================
```

## Checking Missing values

In [14]:
```
df1.isna().sum()
```

Out[14]:
```
SKU_NAME            0
FEED_DATE           0
CATEGORY            0
SUB_CATEGORY        0
ORDERED_REVENUE     0
ORDERED_UNITS       0
REP_OOS           689
dtype: int64
```

REP_OOS contains 689 null values

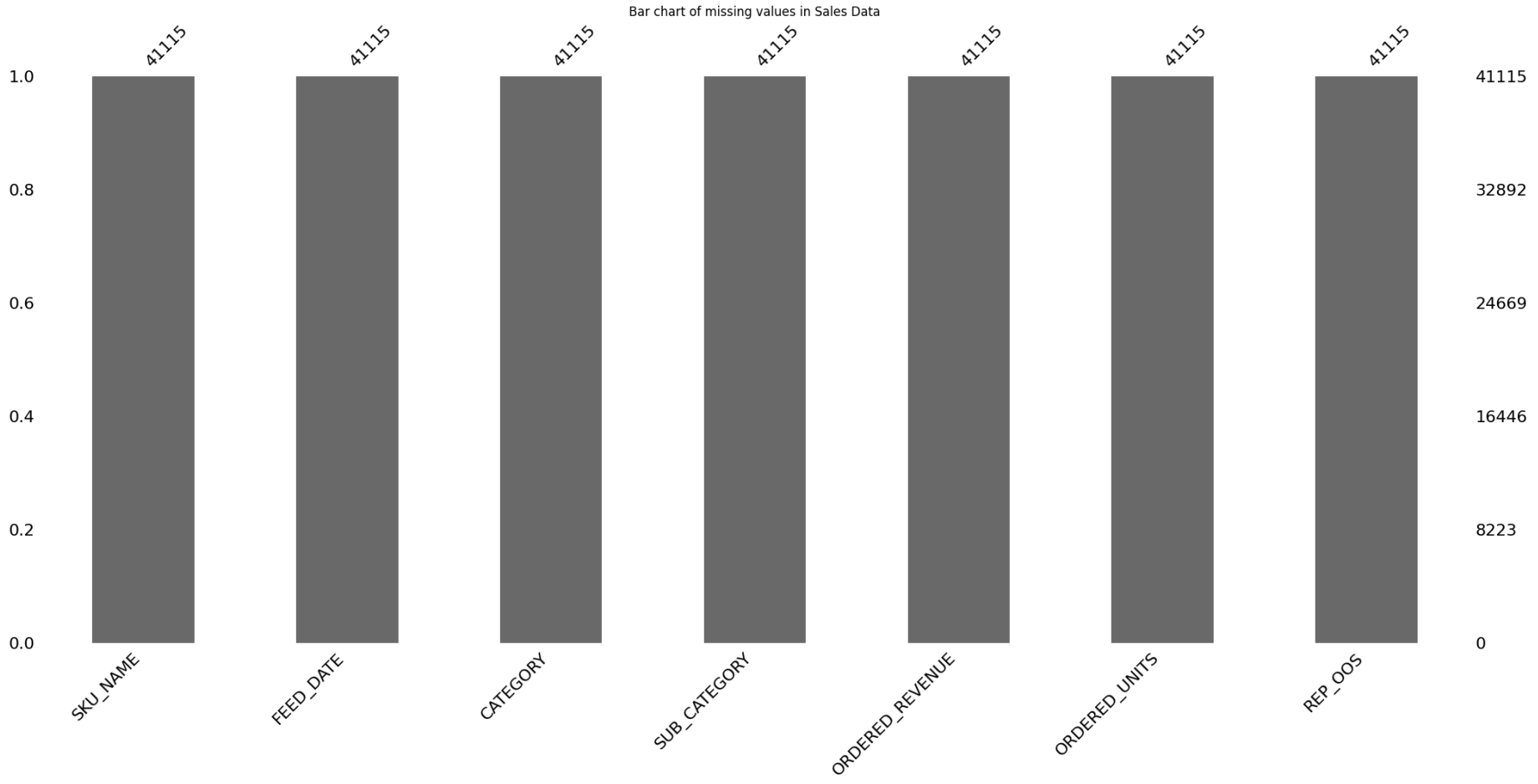
Zero out of stock events

In [15]:
```
df1['REP_OOS'] = df1['REP_OOS'].fillna(0)
```

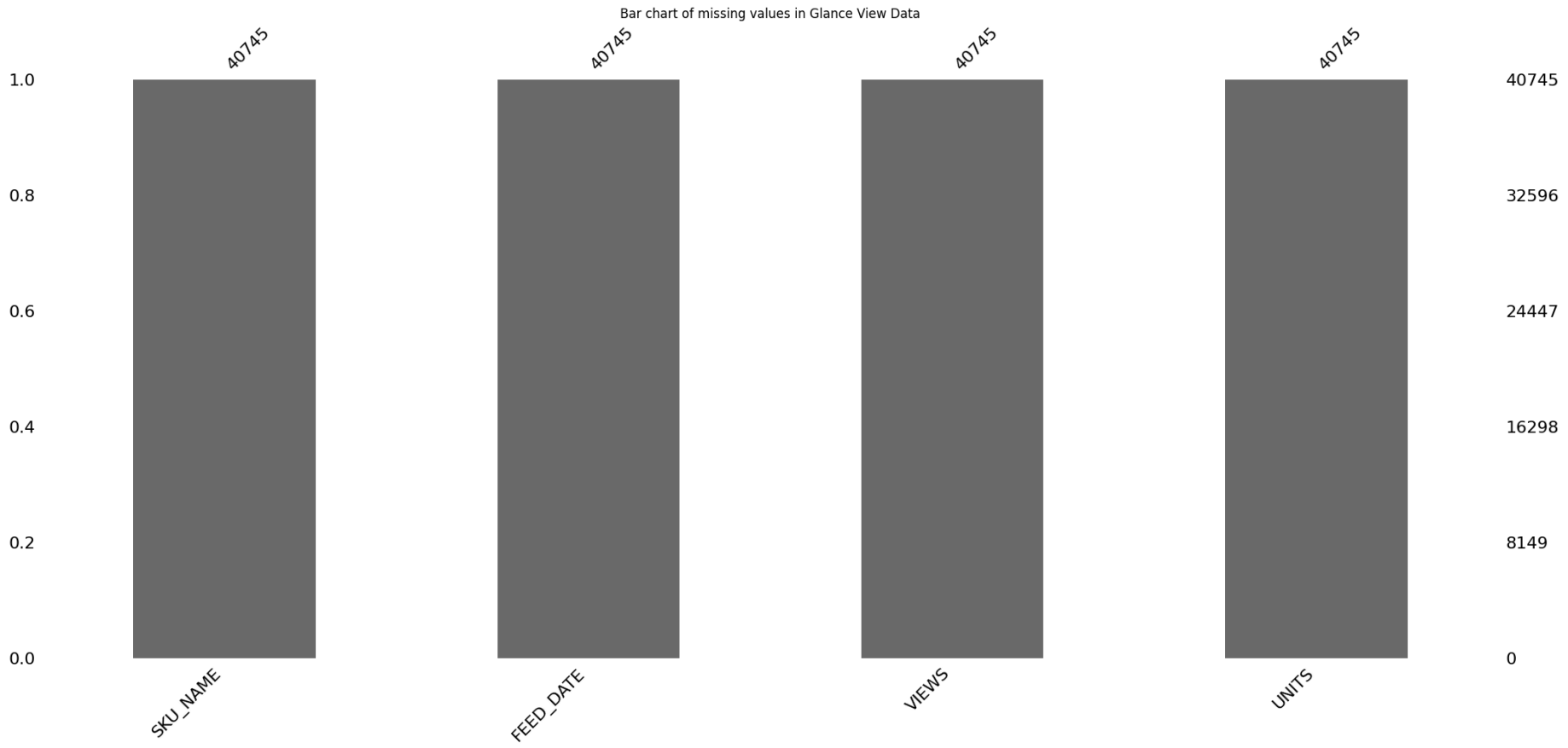
In [16]:
```
df2.isna().sum()
```

Out[16]:
```
SKU_NAME     0
FEED_DATE    0
VIEWS        0
UNITS        0
dtype: int64
```

In [17]:
```
msno.bar(df1)
plt.title('Bar chart of missing values in Sales Data')
plt.show()
```




In [18]:
```
msno.bar(df2)
plt.title('Bar chart of missing values in Glance View Data')
plt.show()
```




In [19]:
```
df1.describe(include='all')
```

Out[19]:

| | SKU_NAME | FEED_DATE | CATEGORY | SUB_CATEGORY | ORDERED_REVENUE | ORDERED_UNITS | REP_OOS |
|---|---|---|---|---|---|---|---|
| **count** | 41115 | 41115 | 41115 | 41115 | 4.111500e+04 | 41115.000000 | 41115.000000 |
| **unique** | 465 | NaN | 10 | 24 | NaN | NaN | NaN |
| **top** | D2869MTWCQ | NaN | 1000 Inputs | 1002 Mice | NaN | NaN | NaN |
| **freq** | 123 | NaN | 26943 | 8692 | NaN | NaN | NaN |
| **mean** | NaN | 2019-07-01 17:44:36.950018304 | NaN | NaN | 2.206692e+03 | 50.904804 | 8.655594 |
| **min** | NaN | 2019-05-01 00:00:00 | NaN | NaN | -3.565780e+04 | -934.000000 | 0.000000 |
| **25%** | NaN | 2019-06-01 00:00:00 | NaN | NaN | 0.000000e+00 | 0.000000 | 0.000000 |
| **50%** | NaN | 2019-07-02 00:00:00 | NaN | NaN | 4.472100e+02 | 6.000000 | 3.750000 |
| **75%** | NaN | 2019-08-02 00:00:00 | NaN | NaN | 2.123820e+03 | 34.000000 | 10.890000 |
| **max** | NaN | 2019-08-31 00:00:00 | NaN | NaN | 1.121838e+06 | 16367.000000 | 118.520000 |
| **std** | NaN | NaN | NaN | NaN | 9.405537e+03 | 210.131201 | 17.322095 |

In [20]:
```
df2.describe(include='all')
```

Out[20]:

| | SKU_NAME | FEED_DATE | VIEWS | UNITS |
|---|---|---|---|---|
| **count** | 40745 | 40745 | 40745.00000 | 40745.000000 |
| **unique** | 452 | NaN | NaN | NaN |
| **top** | B1212:PZ:V | NaN | NaN | NaN |
| **freq** | 123 | NaN | NaN | NaN |
| **mean** | NaN | 2019-07-01 14:20:21.587924736 | 852.54495 | 51.400982 |
| **min** | NaN | 2019-05-01 00:00:00 | 1.00000 | -934.000000 |
| **25%** | NaN | 2019-06-01 00:00:00 | 169.00000 | 0.000000 |
| **50%** | NaN | 2019-07-02 00:00:00 | 397.00000 | 6.000000 |
| **75%** | NaN | 2019-08-01 00:00:00 | 995.00000 | 35.000000 |
| **max** | NaN | 2019-08-31 00:00:00 | 176162.00000 | 16367.000000 |
| **std** | NaN | NaN | 2100.94248 | 211.019576 |

## Duplicate Check

In [21]:
```
df1.duplicated().sum()
```

Out[21]:
```
0
```

In [22]:
```
df2.duplicated().sum()
```

Out[22]: 0

No duplicates in the both datasets

---

## Univariate Analysis

In [23]:
```
merged_data = pd.merge(df1, df2, on=['SKU_NAME', 'FEED_DATE'], how='inner')
#remove units column
merged_data = merged_data.drop('UNITS', axis=1)
```

In [24]:
```
merged_data
```

Out[24]:

| | SKU_NAME | FEED_DATE | CATEGORY | SUB_CATEGORY | ORDERED_REVENUE | ORDERED_UNITS | REP_OOS | VIEWS |
|---|---|---|---|---|---|---|---|---|
| **0** | B12020KBUI | 2019-05-18 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 | 8 |
| **1** | B12020KBUI | 2019-05-19 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 | 5 |
| **2** | B12020KBUI | 2019-05-22 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 | 8 |
| **3** | B12020KBUI | 2019-05-23 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 | 4 |
| **4** | B12020KBUI | 2019-05-27 | 1000 Inputs | 1002 Mice | 0.00 | 0 | 0.0 | 9 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **40421** | D29S5IMRDI | 2019-08-27 | 1000 Inputs | 1007 Other Input Devices | 129.99 | 1 | 0.0 | 104 |
| **40422** | D29S5IMRDI | 2019-08-28 | 1000 Inputs | 1007 Other Input Devices | 259.98 | 2 | 0.0 | 108 |
| **40423** | D29S5IMRDI | 2019-08-29 | 1000 Inputs | 1007 Other Input Devices | 259.98 | 2 | 0.0 | 86 |
| **40424** | D29S5IMRDI | 2019-08-30 | 1000 Inputs | 1007 Other Input Devices | 259.98 | 2 | 0.0 | 89 |
| **40425** | D29S5IMRDI | 2019-08-31 | 1000 Inputs | 1007 Other Input Devices | 0.00 | 0 | 0.0 | 59 |

40426 rows × 8 columns

In [25]:
```
merged_data['YEAR'] = merged_data['FEED_DATE'].dt.year
merged_data['MONTH'] = merged_data['FEED_DATE'].dt.month
merged_data['DAY'] = merged_data['FEED_DATE'].dt.day
merged_data['WEEKDAY'] = merged_data['FEED_DATE'].dt.weekday
```

In [26]:
```
merged_data.describe()
```

Out[26]:

| | FEED_DATE | ORDERED_REVENUE | ORDERED_UNITS | REP_OOS | VIEWS | YEAR | MONTH | DAY | WI |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 40426 | 4.042600e+04 | 40426.000000 | 40426.000000 | 40426.000000 | 40426.0 | 40426.000000 | 40426.000000 | 40426 |
| **mean** | 2019-07-01 16:45:17.142433024 | 2.244341e+03 | 51.772696 | 8.803116 | 855.868006 | 2019.0 | 6.525132 | 15.930738 | 3 |
| **min** | 2019-05-01 00:00:00 | -3.565780e+04 | -934.000000 | 0.000000 | 1.000000 | 2019.0 | 5.000000 | 1.000000 | 0 |
| **25%** | 2019-06-01 00:00:00 | 0.000000e+00 | 0.000000 | 0.000000 | 170.000000 | 2019.0 | 6.000000 | 8.000000 | 1 |
| **50%** | 2019-07-02 00:00:00 | 4.783800e+02 | 6.000000 | 3.980000 | 399.000000 | 2019.0 | 7.000000 | 16.000000 | 3 |
| **75%** | 2019-08-02 00:00:00 | 2.187637e+03 | 36.000000 | 11.040000 | 1003.000000 | 2019.0 | 8.000000 | 24.000000 | 5 |
| **max** | 2019-08-31 00:00:00 | 1.121838e+06 | 16367.000000 | 118.520000 | 176162.000000 | 2019.0 | 8.000000 | 31.000000 | 6 |
| **std** | NaN | 9.480891e+03 | 211.808279 | 17.431879 | 2103.969243 | 0.0 | 1.121014 | 8.847382 | 1 |

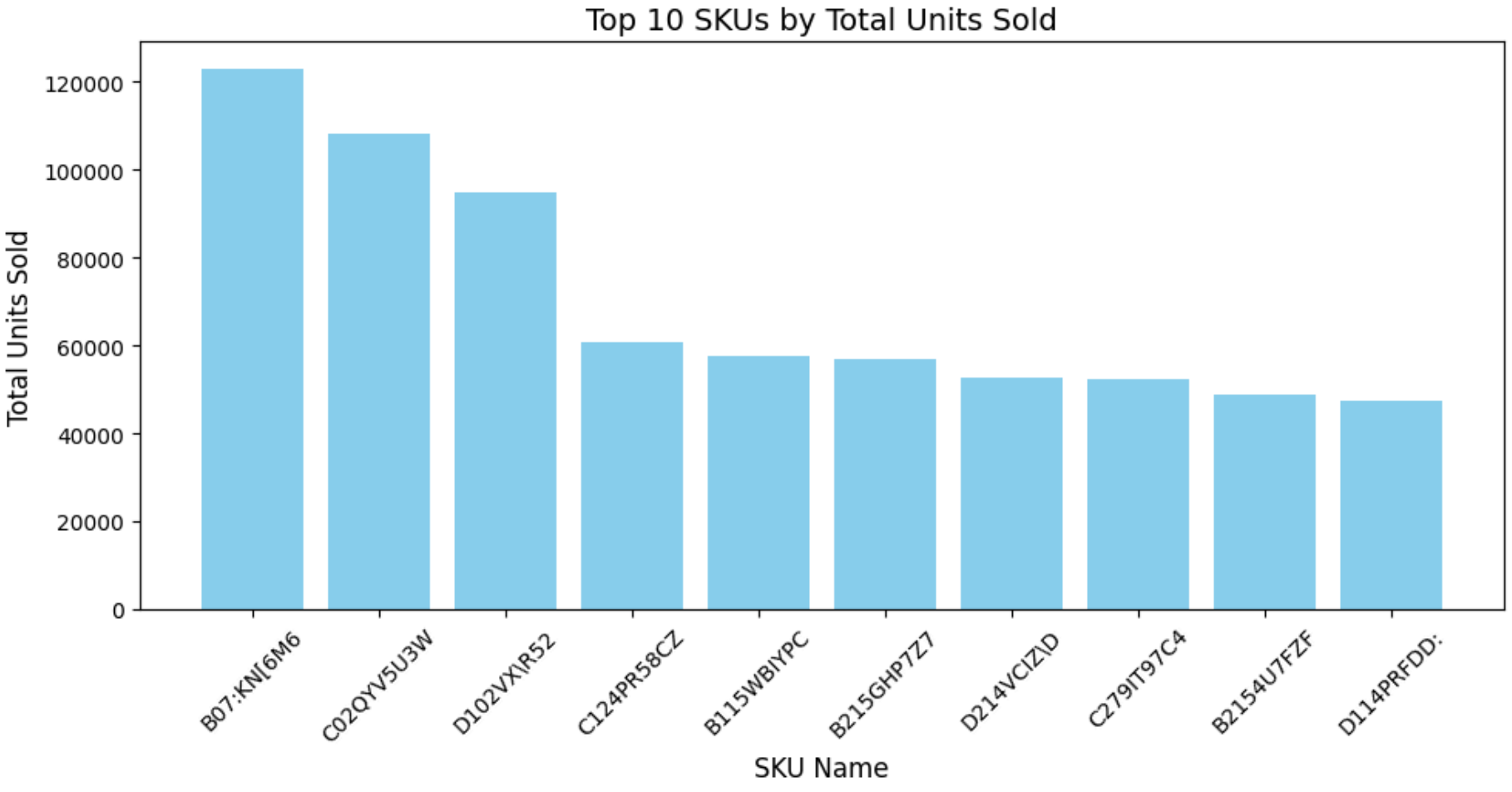
In [27]:
```
categorical_analysis = {
    col: merged_data[col].value_counts()
    for col in merged_data.select_dtypes(include=['object']).columns
}

categorical_analysis
```

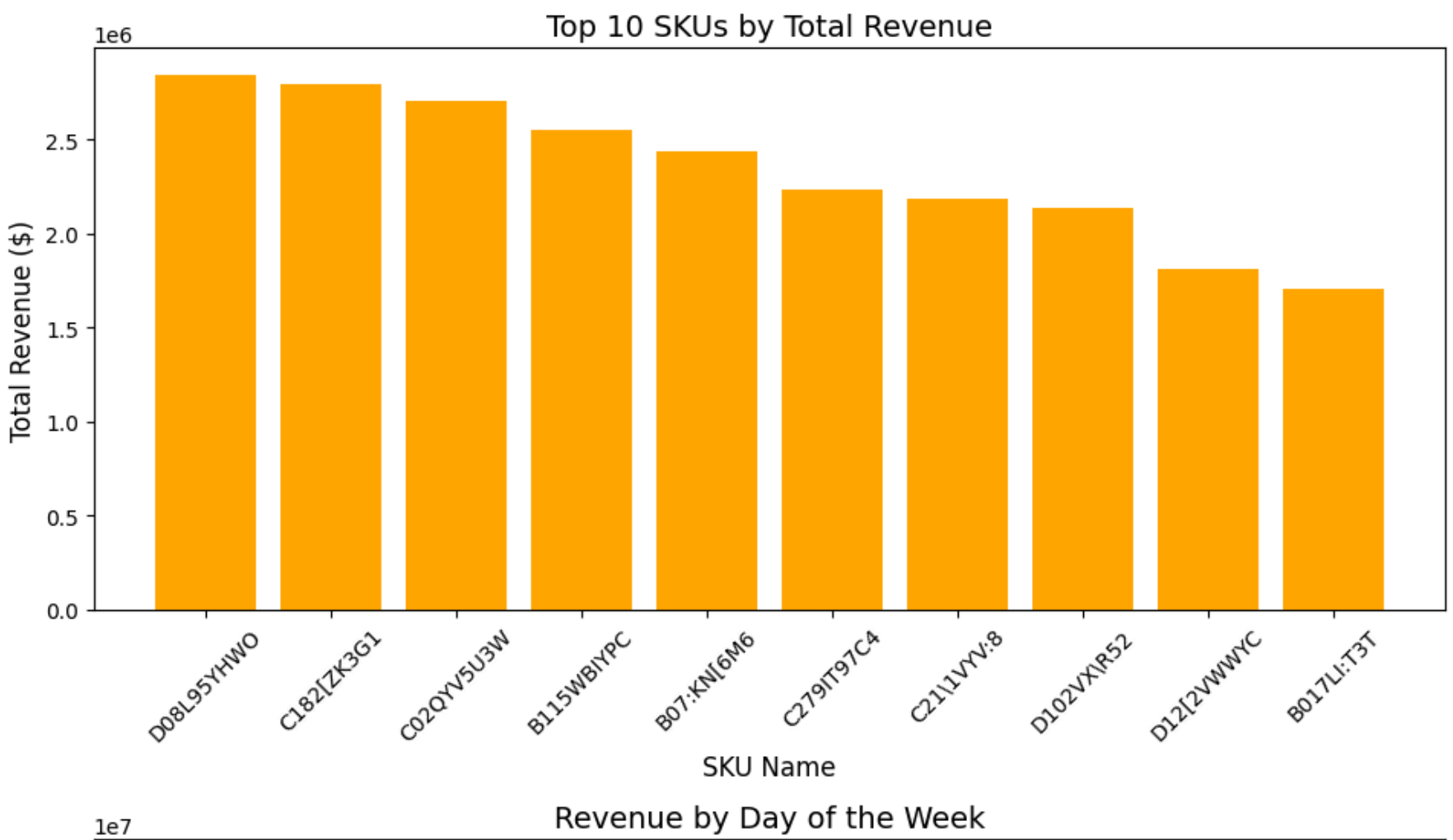
Out[27]:
```
{'SKU_NAME': SKU_NAME
 B11J0HXCQI    123
 C03CBL[721    123
 B28D3XMS37    123
 D27B3YDDMX    123
 D11DRPRJ84    123
               ...
 D01CLVJV9A      1
 B123P7CEQM      1
 D235O9[O\M      1
 C12;C;8OPZ      1
 B21E\9ZKDH      1
 Name: count, Length: 452, dtype: int64,
 'CATEGORY': CATEGORY
 1000 Inputs                          26714
 5000 Portable Media Players           5658
 5300 Headphones                       1849
 5600 Video Components                 1607
 1500 Tablet Accessories               1536
 10800 Xbox One Accessories            1399
 1600 Sony PSP Games and Software       780
 0400 Computer Peripherals              597
 6200 PC Accessories                    235
 0100 Wireless Phones                    51
 Name: count, dtype: int64,
 'SUB_CATEGORY': SUB_CATEGORY
 1002 Mice                                    8509
 1001 Keyboards                               5844
 5045 Media Speaker Systems                   4979
 1004 Computer Headsets and Mics              4190
 1003 Computer Speakers                       3043
 1005 Webcams                                 2673
 1006 Gamepads and Controllers                2275
 5310 Headphones                              1849
 5610 A/V Remote Controls                     1607
 10830 Headsets                               1399
 1590 Other Tablet Accessories                1046
 1610 Classic Games & RetroArcade              780
 5010 Other Portable Audio                     679
 1501 Tablet Carrying Cases & Style            375
 0430 Computer Headsets and Mics - DELETED     252
 6230 Headsets                                 235
 1007 Other Input Devices                      159
 0435 Webcams - DELETED                        128
 0455 Keyboards - DELETED                      119
 1504 Tablet Stands and Docks                  115
 0499 Computer Peripherals Other - DELETED      94
 0191 Connected Wearables                       51
 1008 Computer Peripherals Other                21
 0460 Mice - DELETED                             4
 Name: count, dtype: int64}
```

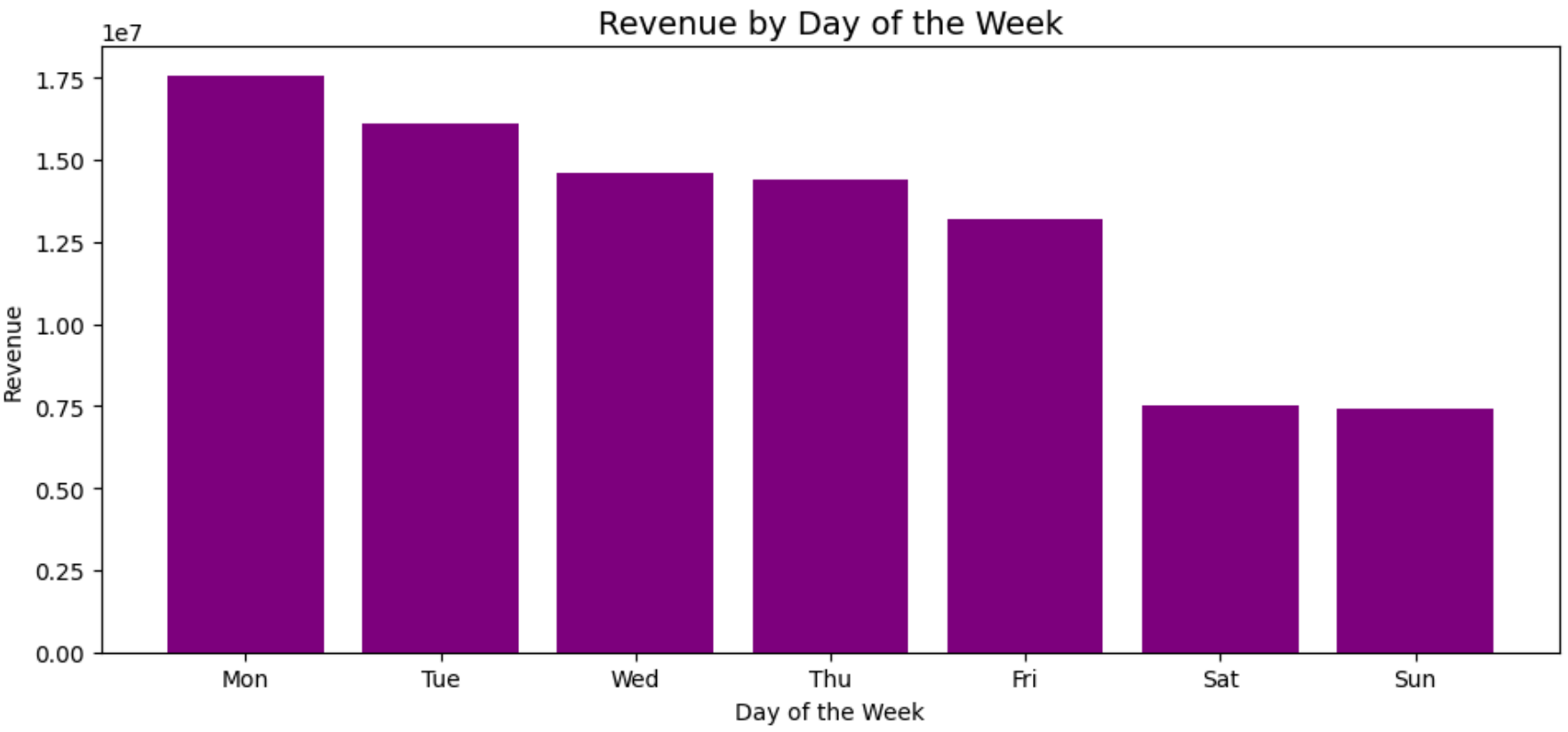
- **Revenue**: Ranges from negative values (likely anomalies) to a maximum of $1,121,838.
- **Units Sold**: Includes negative values, which need further review for validity.
- **Views**: Averages 856 views per product, with a maximum of 176,162 views.
- **SKU_NAME**: 452 unique SKUs, with the most frequent SKUs appearing 123 times.
- **CATEGORY**: The most common category is "1000 Inputs", accounting for over 26,000 records.
- **SUB_CATEGORY**: The top subcategory is "1002 Mice", with 8,509 records, followed by Keyboards and Media Speaker Systems.

In [28]:
```python
fig, axes = plt.subplots(3, 1, figsize=(10, 15))

# Aggregating data to find the top SKUs by total units sold
top_skus = merged_data.groupby('SKU_NAME')['ORDERED_UNITS'].sum().sort_values(ascending=False).head(10)

# Plotting the top SKUs by total units sold
axes[0].bar(top_skus.index, top_skus.values, color='skyblue')
axes[0].set_title('Top 10 SKUs by Total Units Sold', fontsize=14)
axes[0].set_xlabel('SKU Name', fontsize=12)
axes[0].set_ylabel('Total Units Sold', fontsize=12)
axes[0].tick_params(axis='x', rotation=45)

# Aggregating data to find the top SKUs by total revenue
top_revenue_skus = merged_data.groupby('SKU_NAME')['ORDERED_REVENUE'].sum().sort_values(ascending=False).head(10)

# Plotting the top SKUs by total revenue
axes[1].bar(top_revenue_skus.index, top_revenue_skus.values, color='orange')
axes[1].set_title('Top 10 SKUs by Total Revenue', fontsize=14)
axes[1].set_xlabel('SKU Name', fontsize=12)
axes[1].set_ylabel('Total Revenue ($)', fontsize=12)
axes[1].tick_params(axis='x', rotation=45)

# Weekday Revenue
weekday_revenue = merged_data.groupby('WEEKDAY')['ORDERED_REVENUE'].sum().reset_index()
weekday_revenue = weekday_revenue.sort_values('ORDERED_REVENUE', ascending=False)
```

```
axes[2].bar(weekday_revenue['WEEKDAY'], weekday_revenue['ORDERED_REVENUE'], color='purple')
axes[2].set_title('Revenue by Day of the Week', fontsize=14)
axes[2].set_xticks(range(7))
axes[2].set_xticklabels(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'])
axes[2].set_xlabel('Day of the Week')
axes[2].set_ylabel('Revenue')

plt.tight_layout()
plt.show()
```




## Insights

- Top 10 SKUs by Total Units Sold:

- The first subplot highlights the most popular SKUs based on total units sold.
- A few SKUs significantly outperform the rest, indicating high demand or effective promotions.
- SKU dominance might suggest targeted marketing or restocking priorities for these products.

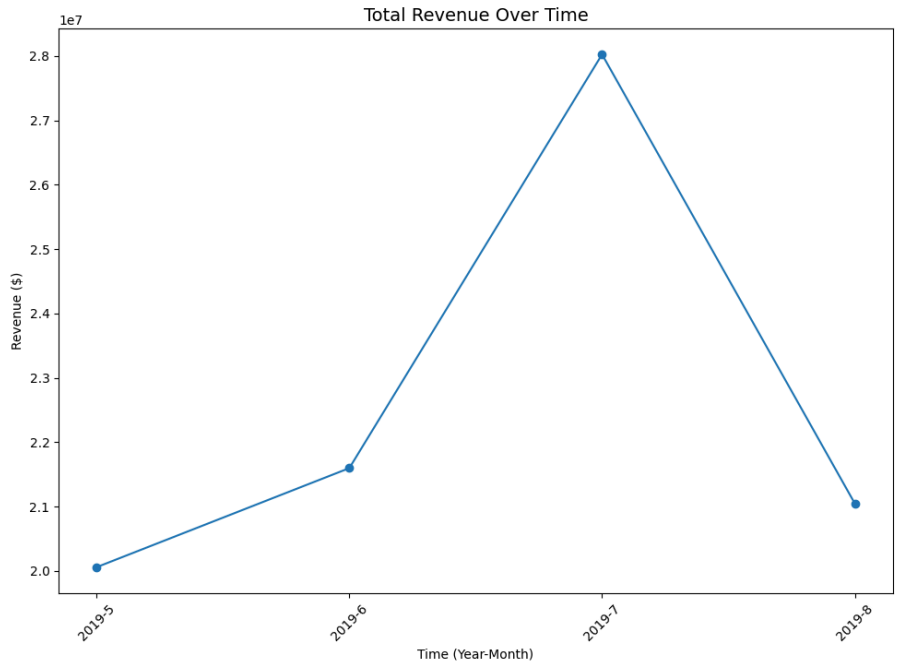
- Top 10 SKUs by Total Revenue:

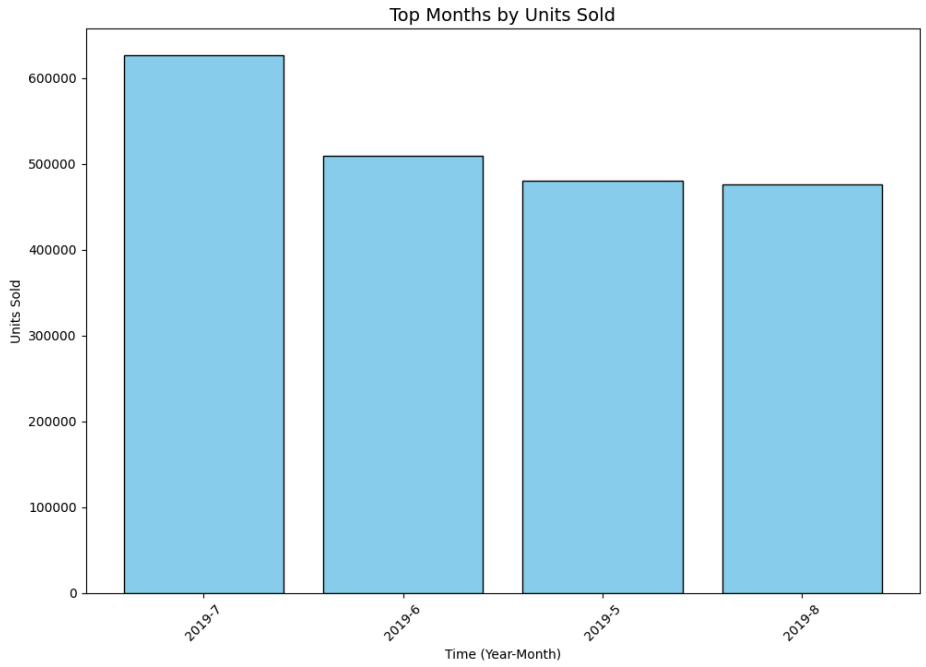
- The second subplot shows SKUs contributing the most revenue.
- There may be overlap between high-unit SKUs and high-revenue SKUs, though revenue may also be driven by high-price items.
- Products with high revenue but lower units sold could represent premium offerings.

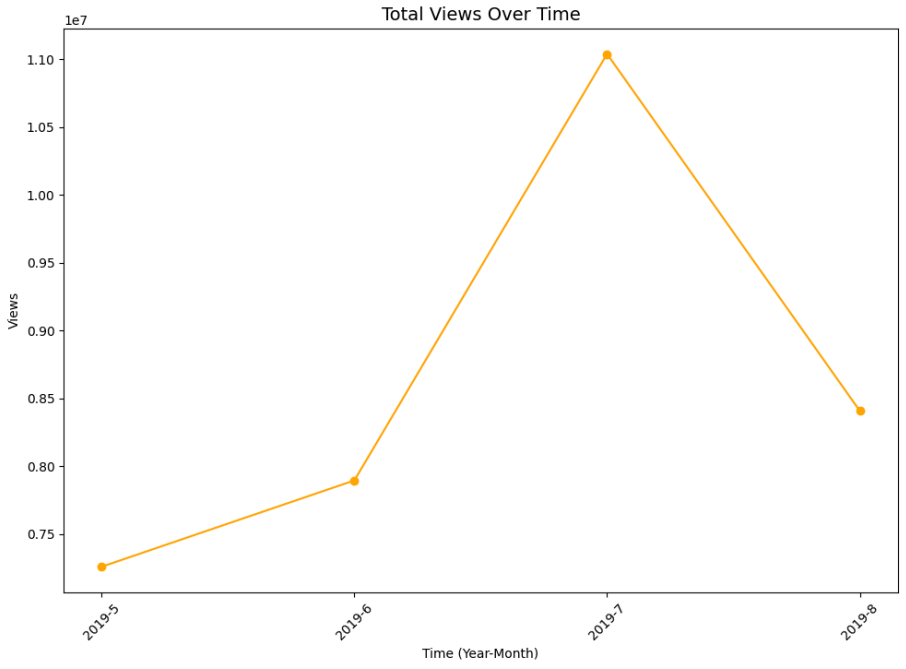
- Revenue by Day of the Week:

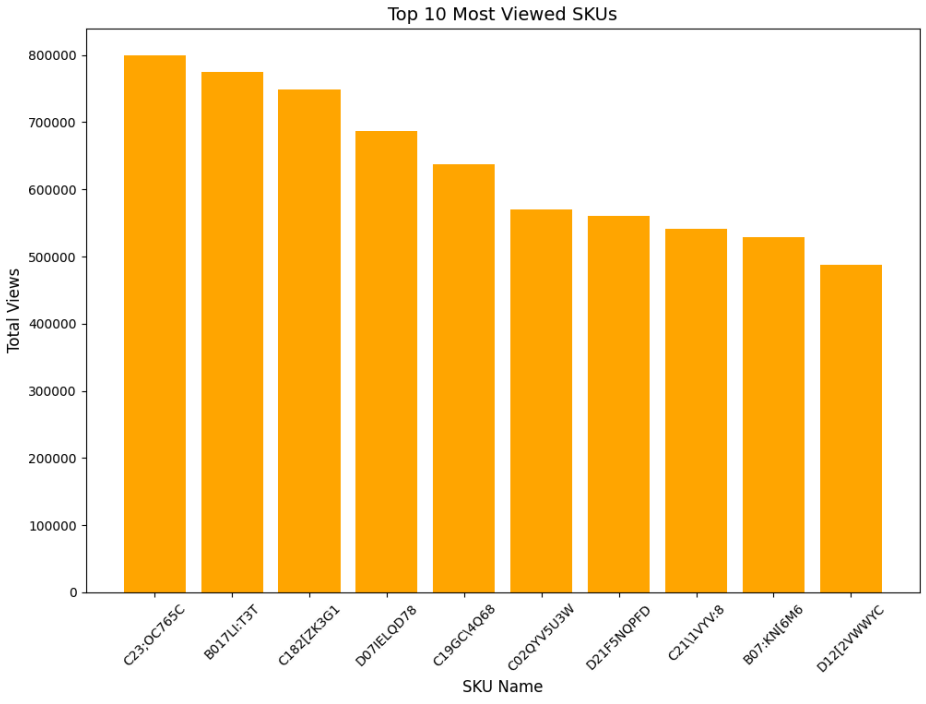
- The third subplot indicates revenue patterns across the week.
- Peaks and troughs suggest consumer shopping behavior, such as higher purchases during weekends or specific weekdays.
- If weekdays show consistent dips, weekday-specific promotions could be an area of improvement.

In [29]:
```
# Aggregate data for insights
monthly_sales = merged_data.groupby(['YEAR', 'MONTH'])[['ORDERED_REVENUE', 'ORDERED_UNITS']].sum()
monthly_views = merged_data.groupby(['YEAR', 'MONTH'])['VIEWS'].sum()

# Flatten the index for monthly_sales to create a single "Year-Month" column
monthly_sales = monthly_sales.reset_index()
monthly_sales['YEAR_MONTH'] = monthly_sales['YEAR'].astype(str) + '-' + monthly_sales['MONTH'].astype(str)

# Plot the data
fig, axs = plt.subplots(2, 2, figsize=(20, 15))

# 1. Revenue Over Time
axs[0, 0].plot(monthly_sales['YEAR_MONTH'], monthly_sales['ORDERED_REVENUE'], marker='o')
axs[0, 0].set_title("Total Revenue Over Time", fontsize=14)
axs[0, 0].set_xlabel("Time (Year-Month)")
axs[0, 0].set_ylabel("Revenue ($)")
axs[0, 0].tick_params(axis='x', rotation=45)

# 2. Units Sold (Sorted Bar Graph)
sorted_units = monthly_sales.sort_values('ORDERED_UNITS', ascending=False)
axs[0, 1].bar(sorted_units['YEAR_MONTH'].head(10), sorted_units['ORDERED_UNITS'].head(10), color='skyblue', edgecolor='black')
axs[0, 1].set_title("Top Months by Units Sold", fontsize=14)
axs[0, 1].set_xlabel("Time (Year-Month)")
axs[0, 1].set_ylabel("Units Sold")
axs[0, 1].tick_params(axis='x', rotation=45)

# 3. Views Over Time (Line Graph)
monthly_views = monthly_views.reset_index()
monthly_views['YEAR_MONTH'] = monthly_views['YEAR'].astype(str) + '-' + monthly_views['MONTH'].astype(str)
axs[1, 0].plot(monthly_views['YEAR_MONTH'], monthly_views['VIEWS'], marker='o', color='orange')
axs[1, 0].set_title("Total Views Over Time", fontsize=14)
axs[1, 0].set_xlabel("Time (Year-Month)")
axs[1, 0].set_ylabel("Views")
axs[1, 0].tick_params(axis='x', rotation=45)

most_viewed_skus = merged_data.groupby('SKU_NAME')['VIEWS'].sum().sort_values(ascending=False).head(10)

# Most Viewed SKUs (already aggregated earlier)
axs[1, 1].bar(most_viewed_skus.index, most_viewed_skus.values, color='orange')
axs[1, 1].set_title('Top 10 Most Viewed SKUs', fontsize=14)
axs[1, 1].set_xlabel('SKU Name', fontsize=12)
axs[1, 1].set_ylabel('Total Views', fontsize=12)
axs[1, 1].tick_params(axis='x', rotation=45)


# Improve layout
plt.tight_layout()
plt.show()
```



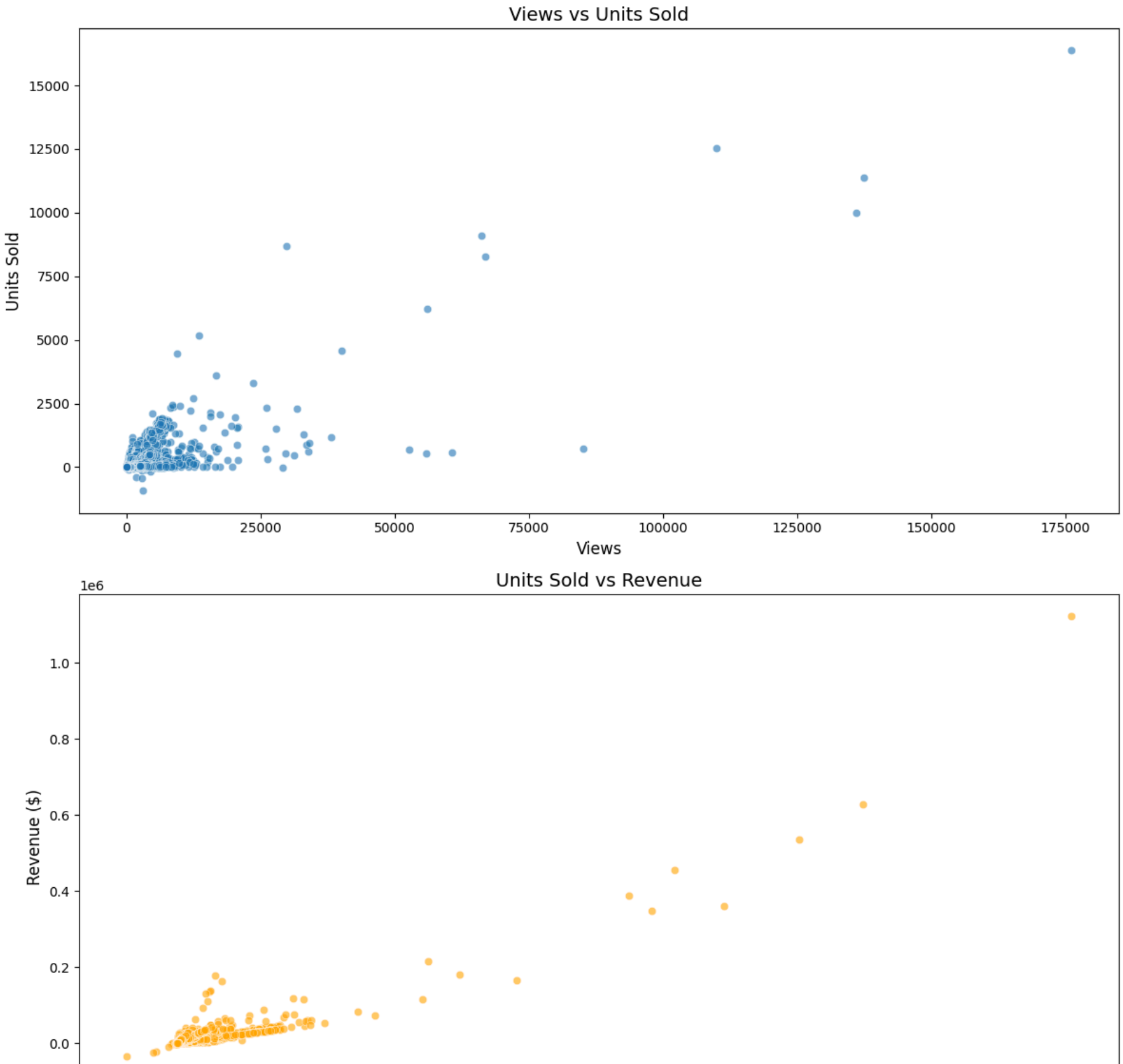

Insights:

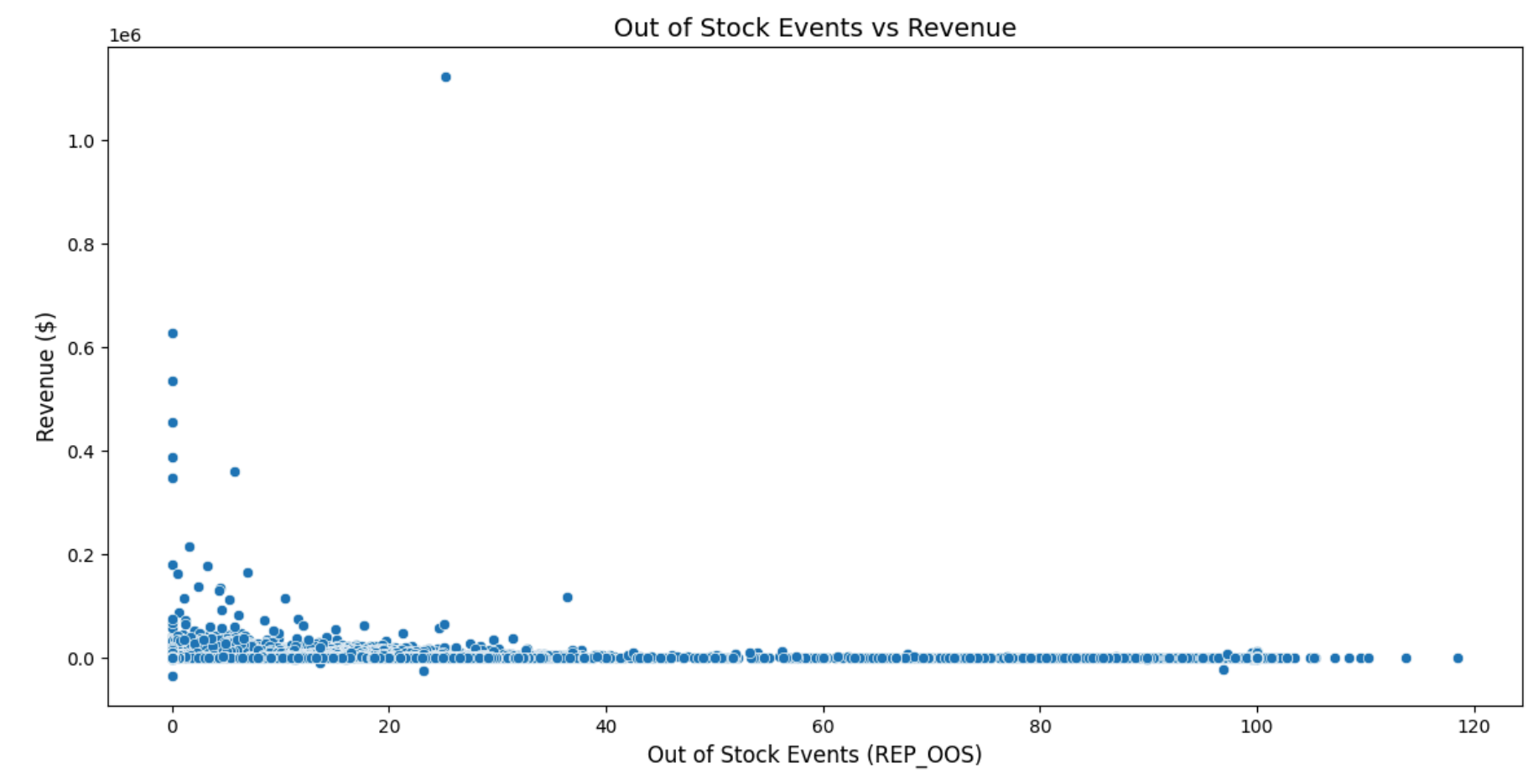
- **Revenue Over Time:**
  - Revenue shows seasonal peaks, indicating high sales periods possibly due to promotions, holidays, or sales events.
  - Highest Revenue achieved in the month of July
- **Views Over Time:**
  - Views follow a trend similar to revenue, suggesting customer interest is directly linked to sales performance.
  - Highest total views was also achieved in the month of July
- **Top 10 Most viewed SKU Names**
  - C23:OC765C product is the most viewed product

---

## Bivariate Analysis

In [30]:
```
fig, axes = plt.subplots(3, 1, figsize=(12, 18))

# 1. Relationship between Views and Units Sold
sns.scatterplot(data=merged_data, x='VIEWS', y='ORDERED_UNITS', ax=axes[0], alpha=0.6)
axes[0].set_title('Views vs Units Sold', fontsize=14)
axes[0].set_xlabel('Views', fontsize=12)
axes[0].set_ylabel('Units Sold', fontsize=12)

# 2. Revenue vs Units Sold
sns.scatterplot(data=merged_data, x='ORDERED_UNITS', y='ORDERED_REVENUE', ax=axes[1], alpha=0.6, color='orange')
axes[1].set_title('Units Sold vs Revenue', fontsize=14)
axes[1].set_xlabel('Units Sold', fontsize=12)
axes[1].set_ylabel('Revenue ($)', fontsize=12)

# 3. Out of Stock Events (REP_OOS) vs Revenue
sns.scatterplot(data=merged_data, x='REP_OOS', y='ORDERED_REVENUE', ax=axes[2])
axes[2].set_title('Out of Stock Events vs Revenue', fontsize=14)
axes[2].set_xlabel('Out of Stock Events (REP_OOS)', fontsize=12)
axes[2].set_ylabel('Revenue ($)', fontsize=12)

plt.tight_layout()
plt.show()
```







**Insights:**

- **Views vs Units Sold:**
  - A positive correlation is expected: SKUs with higher views typically result in higher units sold.

- Exceptions (e.g., high views with low units sold) may indicate issues like poor conversion rates, pricing mismatches, or lack of availability.

- **Units Sold vs Revenue:**
  - A strong positive relationship: More units sold generally lead to higher revenue.
  - Outliers (high revenue with low units sold) likely represent premium-priced SKUs.
- **Out of Stock Events (REP_OOS) vs Revenue:**
  - Higher out-of-stock events could result in lower revenue due to lost sales opportunities.

---

## Correlation

In [31]:
```
merged_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40426 entries, 0 to 40425
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   SKU_NAME         40426 non-null  object
 1   FEED_DATE        40426 non-null  datetime64[ns]
 2   CATEGORY         40426 non-null  object
 3   SUB_CATEGORY     40426 non-null  object
 4   ORDERED_REVENUE  40426 non-null  float64
 5   ORDERED_UNITS    40426 non-null  int64
 6   REP_OOS          40426 non-null  float64
 7   VIEWS            40426 non-null  int64
 8   YEAR             40426 non-null  int32
 9   MONTH            40426 non-null  int32
 10  DAY              40426 non-null  int32
 11  WEEKDAY          40426 non-null  int32
dtypes: datetime64[ns](1), float64(2), int32(4), int64(2), object(3)
memory usage: 3.1+ MB
```

In [32]:
```
merged_data.head()
```

Out[32]:

| | SKU_NAME | FEED_DATE | CATEGORY | SUB_CATEGORY | ORDERED_REVENUE | ORDERED_UNITS | REP_OOS | VIEWS | YEAR | MONTH | DAY | W |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | B12020KBUI | 2019-05-18 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 8 | 2019 | 5 | 18 | |
| **1** | B12020KBUI | 2019-05-19 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 5 | 2019 | 5 | 19 | |
| **2** | B12020KBUI | 2019-05-22 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 8 | 2019 | 5 | 22 | |
| **3** | B12020KBUI | 2019-05-23 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 4 | 2019 | 5 | 23 | |
| **4** | B12020KBUI | 2019-05-27 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 9 | 2019 | 5 | 27 | |

In [33]:
```
numerical_columns = [
    col for col in merged_data.columns
    if merged_data[col].dtype in ['float64', 'int64', 'int32'] and col.lower() not in ['year','day','month','weekday']
]

# Display the resulting numerical columns
numerical_columns
```

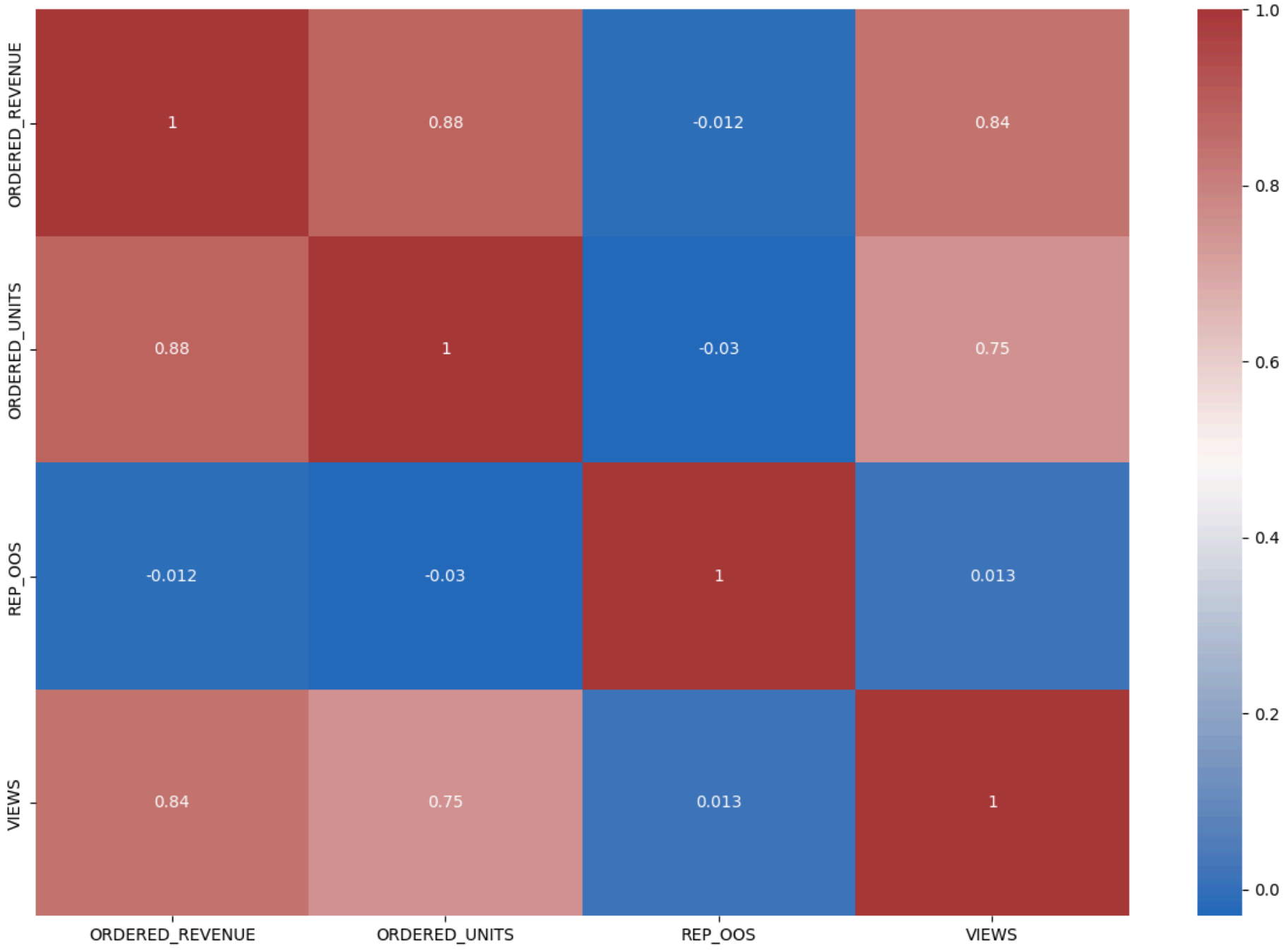
Out[33]:
```
['ORDERED_REVENUE', 'ORDERED_UNITS', 'REP_OOS', 'VIEWS']
```

In [34]:
```
merged_data[numerical_columns].corr()
```

Out[34]:

| | ORDERED_REVENUE | ORDERED_UNITS | REP_OOS | VIEWS |
|---|---|---|---|---|
| **ORDERED_REVENUE** | 1.000000 | 0.881947 | -0.012417 | 0.837448 |
| **ORDERED_UNITS** | 0.881947 | 1.000000 | -0.030230 | 0.749537 |
| **REP_OOS** | -0.012417 | -0.030230 | 1.000000 | 0.013096 |
| **VIEWS** | 0.837448 | 0.749537 | 0.013096 | 1.000000 |

In [35]:
```
plt.figure(figsize=(15,10))

sns.heatmap(merged_data[numerical_columns].corr(), annot=True,cmap='vlag')
plt.show()
```




**Insights:**

- **Highly Correlated Variables**
  - Ordered Units and Ordered Revenue has the highest correlation
  - Ordered Revenue and Views
  - Ordered Units and Views
- **Weakly Correlated Variables**
  - REP_OOS and Ordered Revenue has the least correlation
  - REP_OOS and views
  - REP_OOS and Ordered units

---

# 1. Identify the most expensive SKU, on average, over the entire time period.

In [36]:
```
merged_data.head()
```

Out[36]:

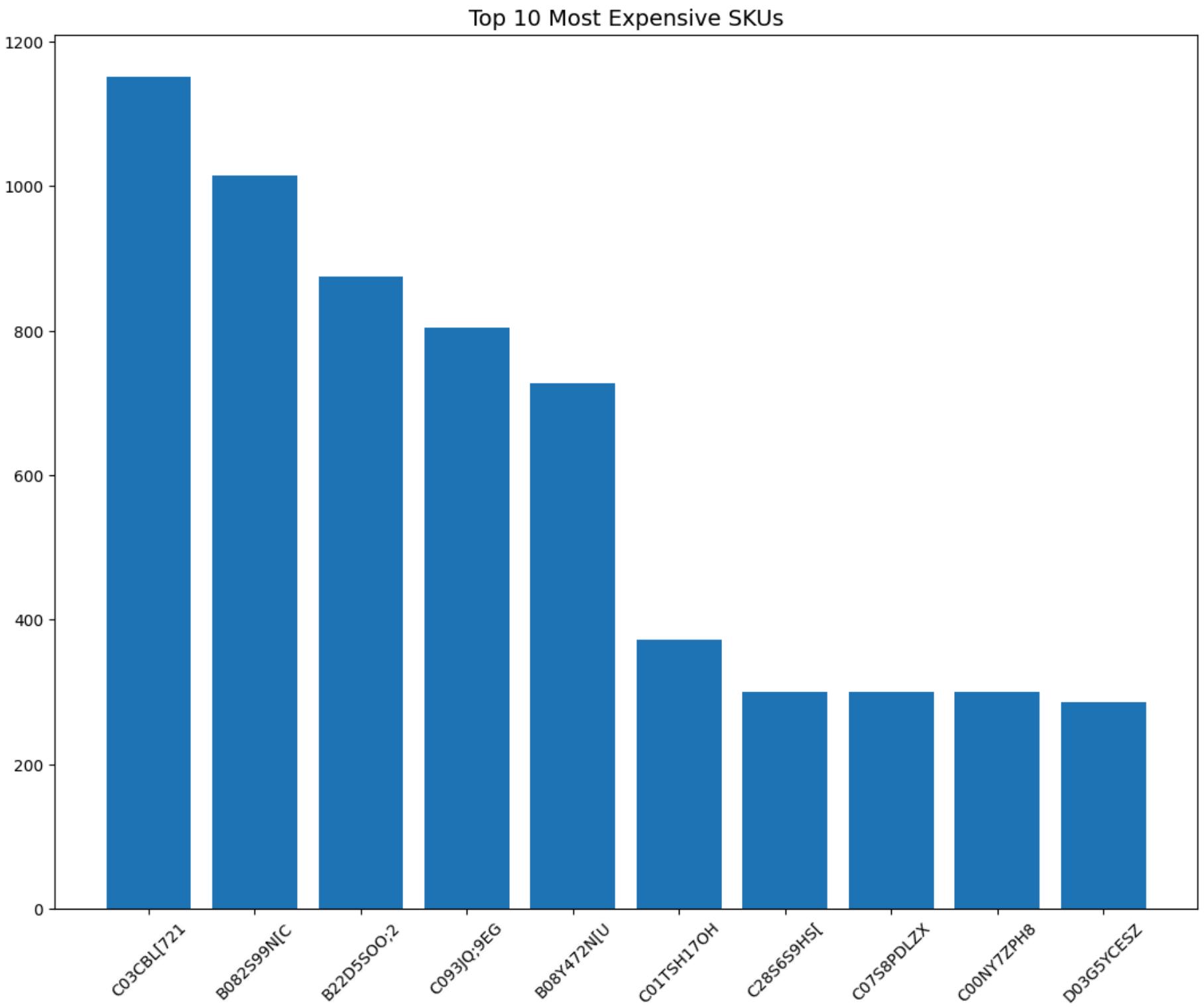
| | SKU_NAME | FEED_DATE | CATEGORY | SUB_CATEGORY | ORDERED_REVENUE | ORDERED_UNITS | REP_OOS | VIEWS | YEAR | MONTH | DAY |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | B12020KBUI | 2019-05-18 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 8 | 2019 | 5 | 18 |
| **1** | B12020KBUI | 2019-05-19 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 5 | 2019 | 5 | 19 |
| **2** | B12020KBUI | 2019-05-22 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 8 | 2019 | 5 | 22 |
| **3** | B12020KBUI | 2019-05-23 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 4 | 2019 | 5 | 23 |
| **4** | B12020KBUI | 2019-05-27 | 1000 Inputs | 1002 Mice | 0.0 | 0 | 0.0 | 9 | 2019 | 5 | 27 |

In [37]:
```
merged_data['PRICE'] = merged_data.apply(
    lambda row: row['ORDERED_REVENUE'] / row['ORDERED_UNITS'] if row['ORDERED_UNITS'] > 0 else None, axis=1
)

# Compute the average price for each SKU across the dataset
average_price_by_sku = merged_data.groupby('SKU_NAME')['PRICE'].mean()
```

```
# Identify the most expensive SKU and its average price
most_expensive_sku = average_price_by_sku.idxmax()
max_average_price = average_price_by_sku.max()

print('Most expensive SKU:', most_expensive_sku, '\nAverage price:', max_average_price)
```

```
Most expensive SKU: C03CBL[721
Average price: 1151.8587273143523
```

In [38]:
```
#Top 10 most expensive SKUs
top_10_expensive_skus = average_price_by_sku.sort_values(ascending=False).head(10)


plt.figure(figsize=(13,10))
plt.title('Top 10 Most Expensive SKUs', fontsize=14)
plt.tick_params(axis='x', rotation=45)
plt.bar(top_10_expensive_skus.index, top_10_expensive_skus.values)
plt.show()
```




---

## 2. What % of SKUs have generated some revenue in this time period? (brownie points - can you identify SKUs that stopped selling completely after July?)

In [39]:
```
sku_revenue = merged_data.groupby('SKU_NAME')['ORDERED_REVENUE'].sum()
revenue_generating_skus = sku_revenue[sku_revenue > 0].index
total_skus = merged_data['SKU_NAME'].nunique()
percentage_revenue_generating = (len(revenue_generating_skus) / total_skus) * 100

print(f"Percentage of SKUs generating revenue: {percentage_revenue_generating:.2f}%")
```

```
Percentage of SKUs generating revenue: 80.97%
```

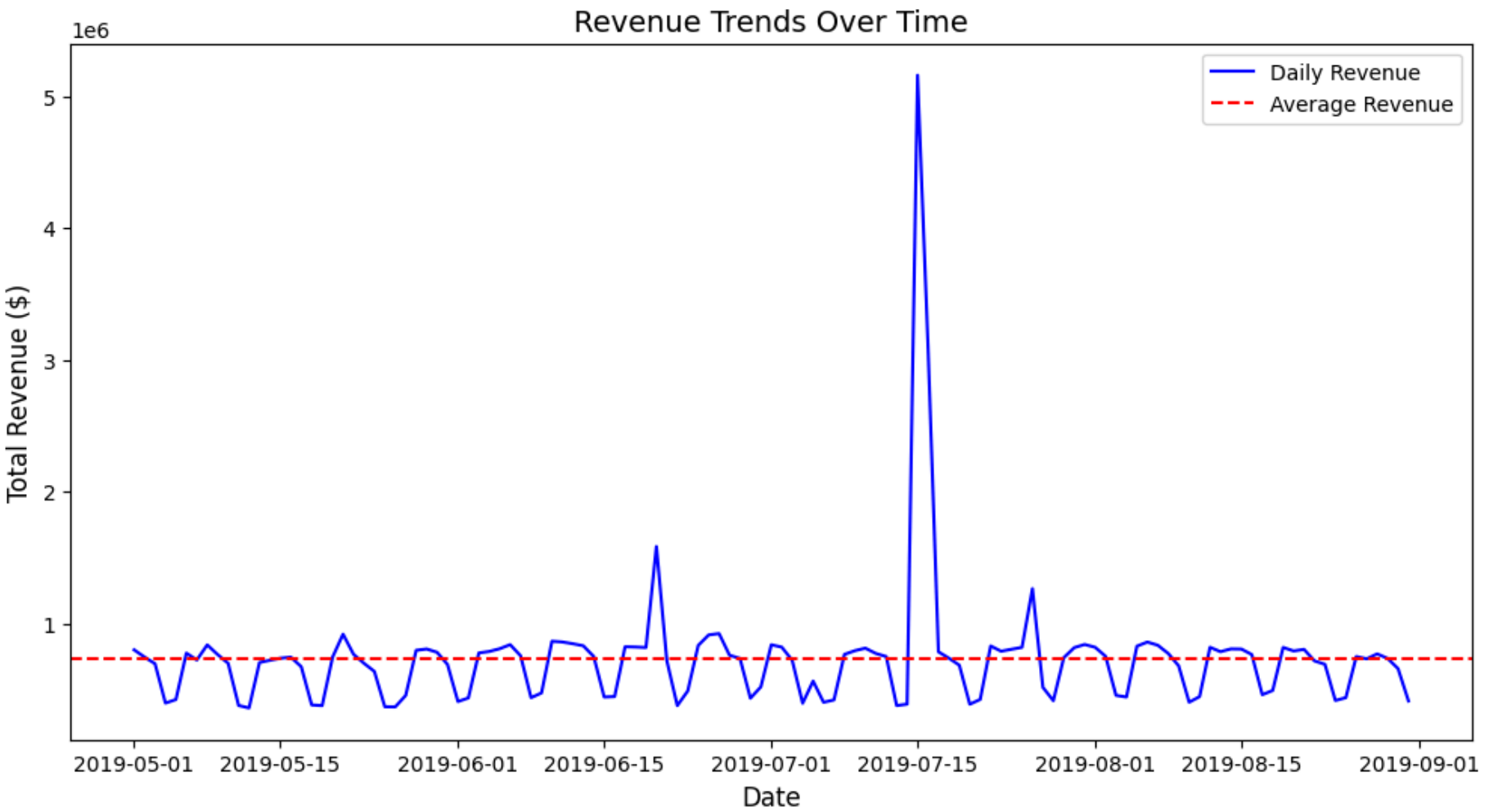
In [40]:
```
skus_selling_after_july = merged_data[merged_data['MONTH'] > 7]['SKU_NAME'].unique()
skus_stopped_selling = set(revenue_generating_skus) - set(skus_selling_after_july)

skus_stopped_selling
```

Out[40]:
```
{'B09FIZS5TZ',
 'B10LJIXFL0',
 'C02228YPPT',
 'C07S8PDLZX',
 'C12GZK3L49',
 'C17NEDU7P[',
 'C19T:CGV3L',
 'C28S6S9HS[',
 'C29LCOGDHZ',
 'D278[PCGZN',
 'D27IO46C5P'}
```

---

## 3. Somewhere in this timeframe, there was a Sale Event. Identify the dates

In [41]:
```
# Aggregating data by date to find patterns or potential sale events
daily_revenue = merged_data.groupby('FEED_DATE')['ORDERED_REVENUE'].sum()

# Plotting the revenue trends to identify sale event dates
plt.figure(figsize=(12, 6))
plt.plot(daily_revenue.index, daily_revenue.values, label='Daily Revenue', color='blue')
plt.title('Revenue Trends Over Time', fontsize=14)
plt.xlabel('Date', fontsize=12)
plt.ylabel('Total Revenue ($)', fontsize=12)
plt.axhline(daily_revenue.mean(), color='red', linestyle='--', label='Average Revenue')
plt.legend()
plt.show()
```



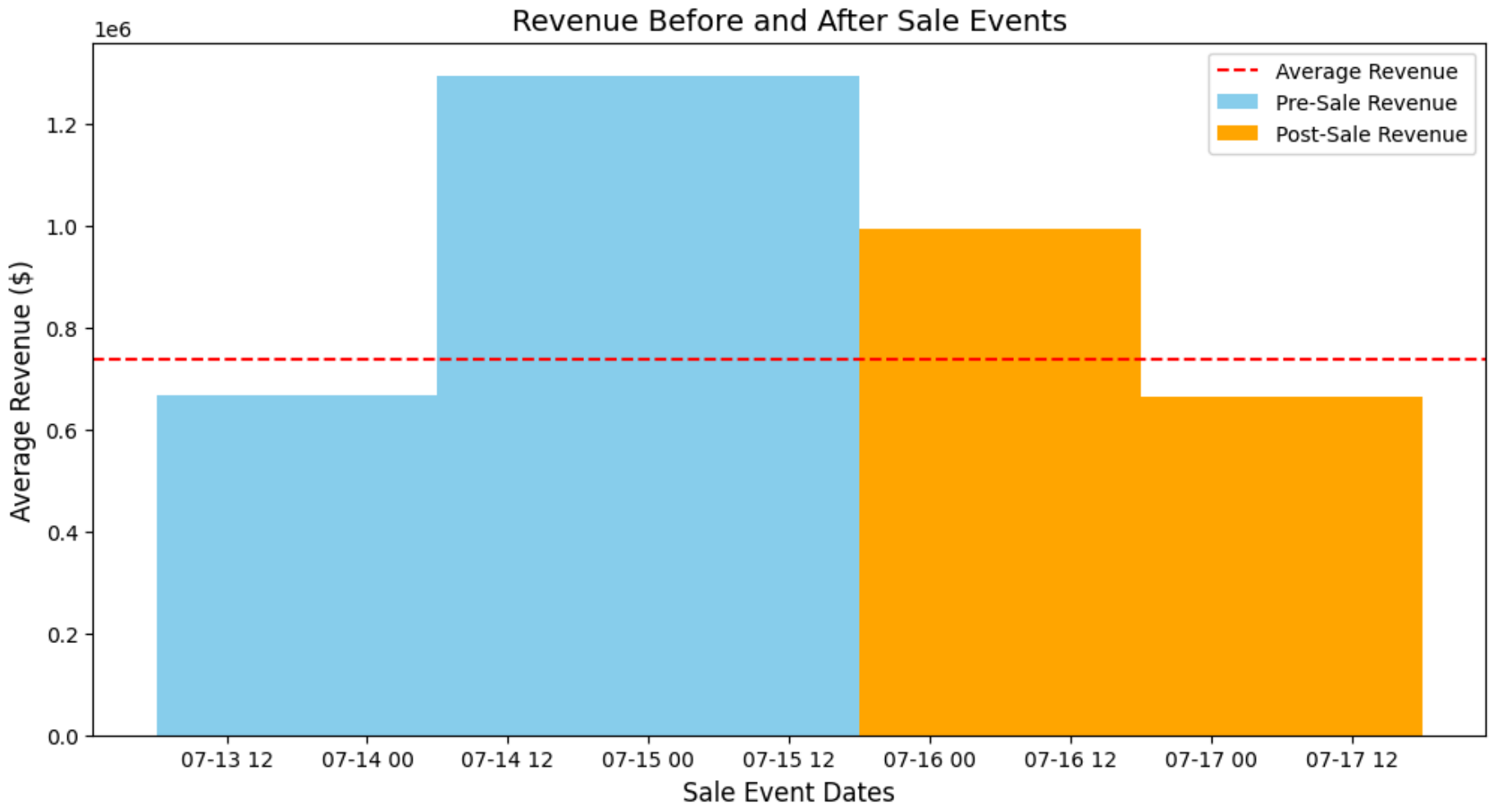

- On 15-Jul-2019 to 31-Jul-2019, Sale Event happened as there was a highest peak in sale over the time.

---

## 4. Does having a sale event cannibalize sales in the immediate aftermath? Highlighting a few examples would suffice (brownie points - determine a statistical metric to prove/disprove this).

In [42]:
```
# Step 1: Identify sale event dates
threshold = daily_revenue.mean() + 2 * daily_revenue.std()
sale_event_dates = daily_revenue[daily_revenue > threshold].index

# Step 2: Analyze revenue before and after sale events (7 days window)
results = []
for date in sale_event_dates:
    pre_sale_period = daily_revenue[(daily_revenue.index >= date - pd.Timedelta(days=7)) & (daily_revenue.index < date)]
    post_sale_period = daily_revenue[(daily_revenue.index > date) & (daily_revenue.index <= date + pd.Timedelta(days=7))]

    # Calculate mean revenue before and after sale events
    pre_sale_avg = pre_sale_period.mean()
    post_sale_avg = post_sale_period.mean()
    results.append((date, pre_sale_avg, post_sale_avg, post_sale_avg - pre_sale_avg))

# Convert results to a DataFrame for visualization
```

```
cannibalization_data = pd.DataFrame(results, columns=['Sale Event Date', 'Pre-Sale Avg Revenue', 'Post-Sale Avg Revenue', 'Cha

# Visualization: Revenue before and after sale events
plt.figure(figsize=(12, 6))
for date, pre, post, _ in results:
    plt.bar(date - pd.Timedelta(days=1), pre, width=1.5, color='skyblue', label='Pre-Sale Revenue' if date == sale_event_dates
    plt.bar(date + pd.Timedelta(days=1), post, width=1.5, color='orange', label='Post-Sale Revenue' if date == sale_event_date

plt.axhline(daily_revenue.mean(), color='red', linestyle='--', label='Average Revenue')
plt.title('Revenue Before and After Sale Events', fontsize=14)
plt.xlabel('Sale Event Dates', fontsize=12)
plt.ylabel('Average Revenue ($)', fontsize=12)
plt.legend()
plt.show()

# Statistical Metric: Perform a paired t-test to compare pre- and post-sale revenues
from scipy.stats import ttest_rel
pre_sale_revenues = cannibalization_data['Pre-Sale Avg Revenue']
post_sale_revenues = cannibalization_data['Post-Sale Avg Revenue']
t_stat, p_value = ttest_rel(pre_sale_revenues, post_sale_revenues)

cannibalization_data, t_stat, p_value
```




Out[42]:
```
(  Sale Event Date  Pre-Sale Avg Revenue  Post-Sale Avg Revenue         Change
 0      2019-07-15          6.687664e+05          995908.044286  327141.611429
 1      2019-07-16          1.295948e+06          665417.572857 -630530.170000,
 0.31679805592565335,
 0.8046877725151356)
```

## 5. In each category, find the subcategory that has grown slowest relative to the category it is present in. If you were handling the entire portfolio, which of these subcategories would you be most concerned with?

In [43]:
```
subcategory_revenue = merged_data.groupby(['CATEGORY', 'SUB_CATEGORY'])['ORDERED_REVENUE'].sum()
subcategory_revenue
```

Out[43]:
```
CATEGORY                          SUB_CATEGORY
0100 Wireless Phones              0191 Connected Wearables                       29115.84
0400 Computer Peripherals         0430 Computer Headsets and Mics - DELETED     337838.28
                                  0435 Webcams - DELETED                         65663.07
                                  0455 Keyboards - DELETED                       75269.93
                                  0460 Mice - DELETED                                0.00
                                  0499 Computer Peripherals Other - DELETED     160657.14
1000 Inputs                       1001 Keyboards                              22425163.19
                                  1002 Mice                                   23290337.91
                                  1003 Computer Speakers                       4830543.40
                                  1004 Computer Headsets and Mics              8151852.98
                                  1005 Webcams                                12230209.80
                                  1006 Gamepads and Controllers                5855070.12
                                  1007 Other Input Devices                      626510.40
                                  1008 Computer Peripherals Other                    0.00
10800 Xbox One Accessories        10830 Headsets                              1206332.62
1500 Tablet Accessories           1501 Tablet Carrying Cases & Style            27952.04
                                  1504 Tablet Stands and Docks                  13131.97
                                  1590 Other Tablet Accessories               1130959.06
1600 Sony PSP Games and Software  1610 Classic Games & RetroArcade            1042615.09
5000 Portable Media Players       5010 Other Portable Audio                    314094.73
                                  5045 Media Speaker Systems                  5765041.39
5300 Headphones                   5310 Headphones                              835263.46
5600 Video Components             5610 A/V Remote Controls                    2200516.21
6200 PC Accessories               6230 Headsets                                115588.36
Name: ORDERED_REVENUE, dtype: float64
```

In [44]:
```
subcategory_revenue.iloc[:2]
```

Out[44]:
```
CATEGORY                  SUB_CATEGORY
0100 Wireless Phones      0191 Connected Wearables                      29115.84
0400 Computer Peripherals 0430 Computer Headsets and Mics - DELETED    337838.28
Name: ORDERED_REVENUE, dtype: float64
```

In [45]:
```
category_revenue=merged_data.groupby(['CATEGORY'])['ORDERED_REVENUE'].sum()
category_revenue
```

Out[45]:
```
CATEGORY
0100 Wireless Phones                29115.84
0400 Computer Peripherals          639428.42
1000 Inputs                      77409687.80
10800 Xbox One Accessories        1206332.62
1500 Tablet Accessories           1172043.07
1600 Sony PSP Games and Software  1042615.09
5000 Portable Media Players       6079136.12
5300 Headphones                    835263.46
5600 Video Components             2200516.21
6200 PC Accessories                115588.36
Name: ORDERED_REVENUE, dtype: float64
```

In [46]:
```
rel_growth=(subcategory_revenue/category_revenue)
rel_growth
```

Out[46]:
```
CATEGORY                          SUB_CATEGORY
0100 Wireless Phones              0191 Connected Wearables                    1.000000
0400 Computer Peripherals         0430 Computer Headsets and Mics - DELETED   0.528344
                                  0435 Webcams - DELETED                      0.102690
                                  0455 Keyboards - DELETED                    0.117714
                                  0460 Mice - DELETED                         0.000000
                                  0499 Computer Peripherals Other - DELETED   0.251251
1000 Inputs                       1001 Keyboards                              0.289695
                                  1002 Mice                                   0.300871
                                  1003 Computer Speakers                      0.062402
                                  1004 Computer Headsets and Mics             0.105308
                                  1005 Webcams                                0.157993
                                  1006 Gamepads and Controllers               0.075637
                                  1007 Other Input Devices                    0.008093
                                  1008 Computer Peripherals Other             0.000000
10800 Xbox One Accessories        10830 Headsets                              1.000000
1500 Tablet Accessories           1501 Tablet Carrying Cases & Style          0.023849
                                  1504 Tablet Stands and Docks                0.011204
                                  1590 Other Tablet Accessories               0.964947
1600 Sony PSP Games and Software  1610 Classic Games & RetroArcade            1.000000
5000 Portable Media Players       5010 Other Portable Audio                   0.051668
                                  5045 Media Speaker Systems                  0.948332
5300 Headphones                   5310 Headphones                             1.000000
5600 Video Components             5610 A/V Remote Controls                    1.000000
6200 PC Accessories               6230 Headsets                               1.000000
Name: ORDERED_REVENUE, dtype: float64
```

In [47]:
```
slow_moving=rel_growth[rel_growth<0.2]
slow_moving
```

Out[47]:
```
CATEGORY                    SUB_CATEGORY
0400 Computer Peripherals   0435 Webcams - DELETED               0.102690
                            0455 Keyboards - DELETED             0.117714
                            0460 Mice - DELETED                  0.000000
1000 Inputs                 1003 Computer Speakers               0.062402
                            1004 Computer Headsets and Mics      0.105308
                            1005 Webcams                         0.157993
                            1006 Gamepads and Controllers        0.075637
                            1007 Other Input Devices             0.008093
                            1008 Computer Peripherals Other      0.000000
1500 Tablet Accessories     1501 Tablet Carrying Cases & Style   0.023849
                            1504 Tablet Stands and Docks         0.011204
5000 Portable Media Players 5010 Other Portable Audio            0.051668
Name: ORDERED_REVENUE, dtype: float64
```

In [ ]:
```
```

## These are the slowest moving subcategories when compared to relative growth in the categories

## 6. Highlight any anomalies/mismatches in the data that you see, if any. (In terms of data quality issues)

- There are some negative values in the units sold
- Compare SKUs in sales_data and glance_views to find mismatches

In [48]:
```
negative_values = merged_data[
    (merged_data['ORDERED_REVENUE'] < 0) |
    (merged_data['ORDERED_UNITS'] < 0) |
    (merged_data['VIEWS'] < 0) ]


negative_values
```

Out[48]:

| | SKU_NAME | FEED_DATE | CATEGORY | SUB_CATEGORY | ORDERED_REVENUE | ORDERED_UNITS | REP_OOS | VIEWS | YEAR | MONTH |
|---|---|---|---|---|---|---|---|---|---|---|
| **686** | C211F62H36 | 2019-07-03 | 1000 Inputs | 1001 Keyboards | -57.99 | -1 | 0.00 | 427 | 2019 | 7 |
| **1951** | B004FMWNKW | 2019-06-21 | 0400 Computer Peripherals | 0455 Keyboards - DELETED | -134.85 | -15 | 0.00 | 535 | 2019 | 6 |
| **1952** | B004FMWNKW | 2019-06-22 | 0400 Computer Peripherals | 0455 Keyboards - DELETED | -80.91 | -9 | 0.00 | 439 | 2019 | 6 |
| **1953** | B004FMWNKW | 2019-06-23 | 0400 Computer Peripherals | 0455 Keyboards - DELETED | -107.88 | -12 | 0.00 | 410 | 2019 | 6 |
| **1954** | B004FMWNKW | 2019-06-24 | 0400 Computer Peripherals | 0455 Keyboards - DELETED | -80.91 | -9 | 0.00 | 562 | 2019 | 6 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **40262** | D28QU2Q7[: | 2019-06-04 | 1000 Inputs | 1003 Computer Speakers | -399.98 | -2 | 0.00 | 20 | 2019 | 6 |
| **40264** | D28QU2Q7[: | 2019-06-06 | 1000 Inputs | 1003 Computer Speakers | -199.99 | -1 | 0.00 | 10 | 2019 | 6 |
| **40266** | D28QU2Q7[: | 2019-06-08 | 1000 Inputs | 1003 Computer Speakers | -199.99 | -1 | 0.00 | 5 | 2019 | 6 |
| **40274** | D28QU2Q7[: | 2019-06-16 | 1000 Inputs | 1003 Computer Speakers | -199.99 | -1 | 4.55 | 44 | 2019 | 6 |
| **40368** | C19T:CGV3L | 2019-05-27 | 5300 Headphones | 5310 Headphones | -29.99 | -1 | 0.00 | 5 | 2019 | 5 |

400 rows × 13 columns

In [49]:
```
sales_skus = set(df1['SKU_NAME'])
views_skus = set(df2['SKU_NAME'])
sku_mismatches = sales_skus.symmetric_difference(views_skus)

sku_mismatches
```

Out[49]:
```
{'B013WPQZ2U',
 'B13I6J4KZO',
 'B205J5RPJQ',
 'B213K5FIPU',
 'B21LO8TNNY',
 'C02KP:UMZK',
 'C10N5I1QJI',
 'C214SL86Q[',
 'D016LM4SES',
 'D126YHPGCS',
 'D12LO8TJJ4',
 'D21LP:UMOR',
 'D237YHQCNZ'}
```

---

## 7. For SKU Name C120[H:8NV, discuss whether Unit Conversion (Units/Views) is affected by Average Selling Price

In [50]:
```
sku_data = merged_data[merged_data['SKU_NAME'] == 'C120[H:8NV']
sku_data['UNITS_CONVERTED'] = sku_data['ORDERED_UNITS'] / sku_data['VIEWS']
sku_data['AVG_SP']=sku_data['ORDERED_REVENUE']/sku_data['ORDERED_UNITS']
sku_data['AVG_SP'] = sku_data['AVG_SP'].fillna(0)

sku_data
```

```
C:\Users\mohit\AppData\Local\Temp\ipykernel_35868\3486845960.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  sku_data['UNITS_CONVERTED'] = sku_data['ORDERED_UNITS'] / sku_data['VIEWS']
C:\Users\mohit\AppData\Local\Temp\ipykernel_35868\3486845960.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  sku_data['AVG_SP']=sku_data['ORDERED_REVENUE']/sku_data['ORDERED_UNITS']
C:\Users\mohit\AppData\Local\Temp\ipykernel_35868\3486845960.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  sku_data['AVG_SP'] = sku_data['AVG_SP'].fillna(0)
```

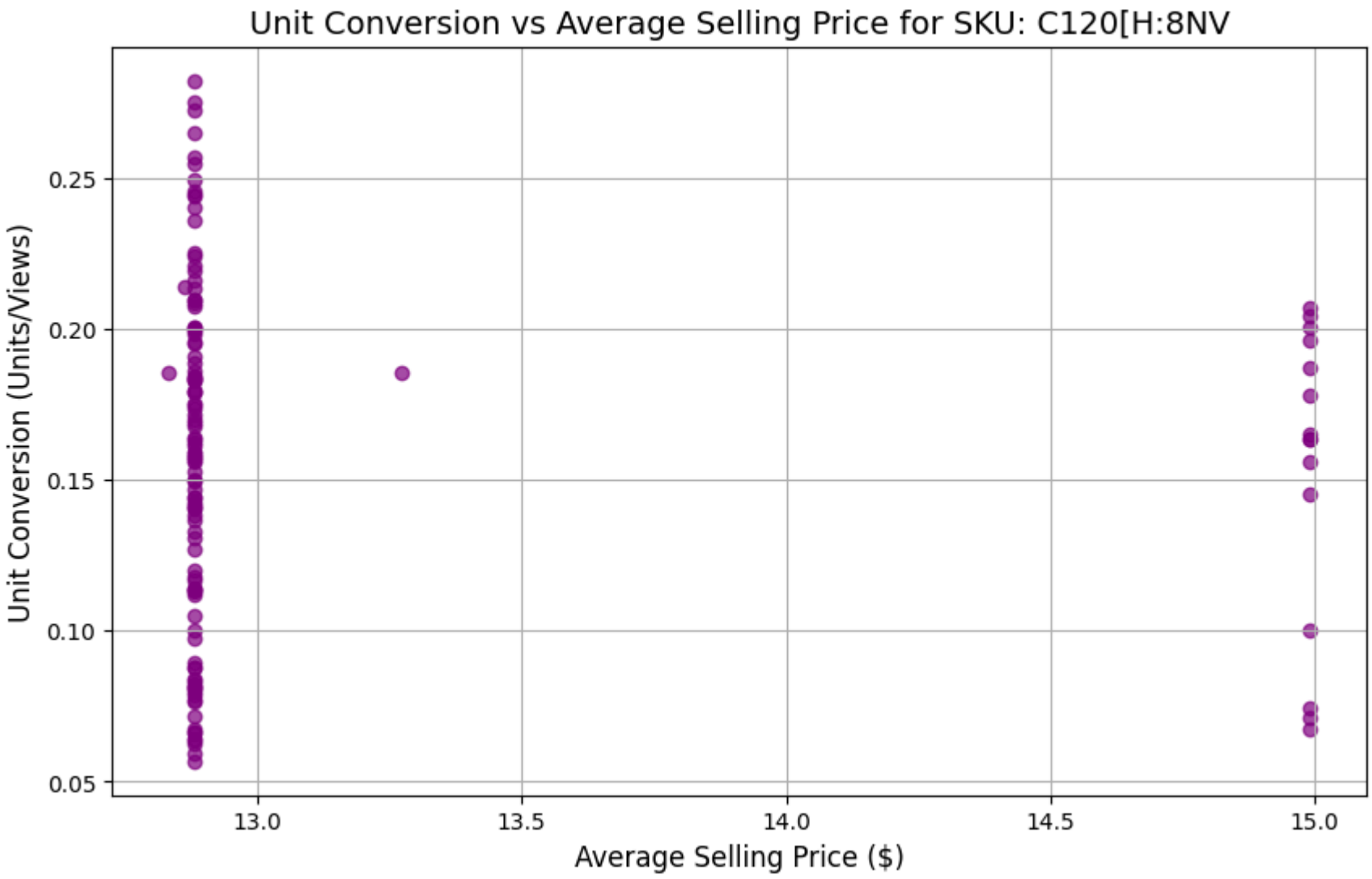
Out[50]:

| | SKU_NAME | FEED_DATE | CATEGORY | SUB_CATEGORY | ORDERED_REVENUE | ORDERED_UNITS | REP_OOS | VIEWS | YEAR | MONTH | DAY |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **382** | C120[H:8NV | 2019-05-01 | 1000 Inputs | 1003 Computer Speakers | 1019.32 | 68 | 4.70 | 468 | 2019 | 5 | 1 |
| **383** | C120[H:8NV | 2019-05-02 | 1000 Inputs | 1003 Computer Speakers | 1064.29 | 71 | 5.06 | 435 | 2019 | 5 | 2 |
| **384** | C120[H:8NV | 2019-05-03 | 1000 Inputs | 1003 Computer Speakers | 1079.28 | 72 | 5.22 | 441 | 2019 | 5 | 3 |
| **385** | C120[H:8NV | 2019-05-04 | 1000 Inputs | 1003 Computer Speakers | 389.74 | 26 | 7.43 | 350 | 2019 | 5 | 4 |
| **386** | C120[H:8NV | 2019-05-05 | 1000 Inputs | 1003 Computer Speakers | 344.77 | 23 | 8.53 | 340 | 2019 | 5 | 5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **500** | C120[H:8NV | 2019-08-27 | 1000 Inputs | 1003 Computer Speakers | 940.24 | 73 | 3.63 | 468 | 2019 | 8 | 27 |
| **501** | C120[H:8NV | 2019-08-28 | 1000 Inputs | 1003 Computer Speakers | 1056.16 | 82 | 6.14 | 505 | 2019 | 8 | 28 |
| **502** | C120[H:8NV | 2019-08-29 | 1000 Inputs | 1003 Computer Speakers | 953.12 | 74 | 8.42 | 404 | 2019 | 8 | 29 |
| **503** | C120[H:8NV | 2019-08-30 | 1000 Inputs | 1003 Computer Speakers | 927.36 | 72 | 7.48 | 401 | 2019 | 8 | 30 |
| **504** | C120[H:8NV | 2019-08-31 | 1000 Inputs | 1003 Computer Speakers | 437.92 | 34 | 5.19 | 289 | 2019 | 8 | 31 |

123 rows × 15 columns

In [51]:
```
plt.figure(figsize=(10, 6))
plt.scatter(sku_data['AVG_SP'], sku_data['UNITS_CONVERTED'], alpha=0.7, color='purple')
```

```
plt.title('Unit Conversion vs Average Selling Price for SKU: C120[H:8NV', fontsize=14)
plt.xlabel('Average Selling Price ($)', fontsize=12)
plt.ylabel('Unit Conversion (Units/Views)', fontsize=12)
plt.grid(True)
plt.show()
```

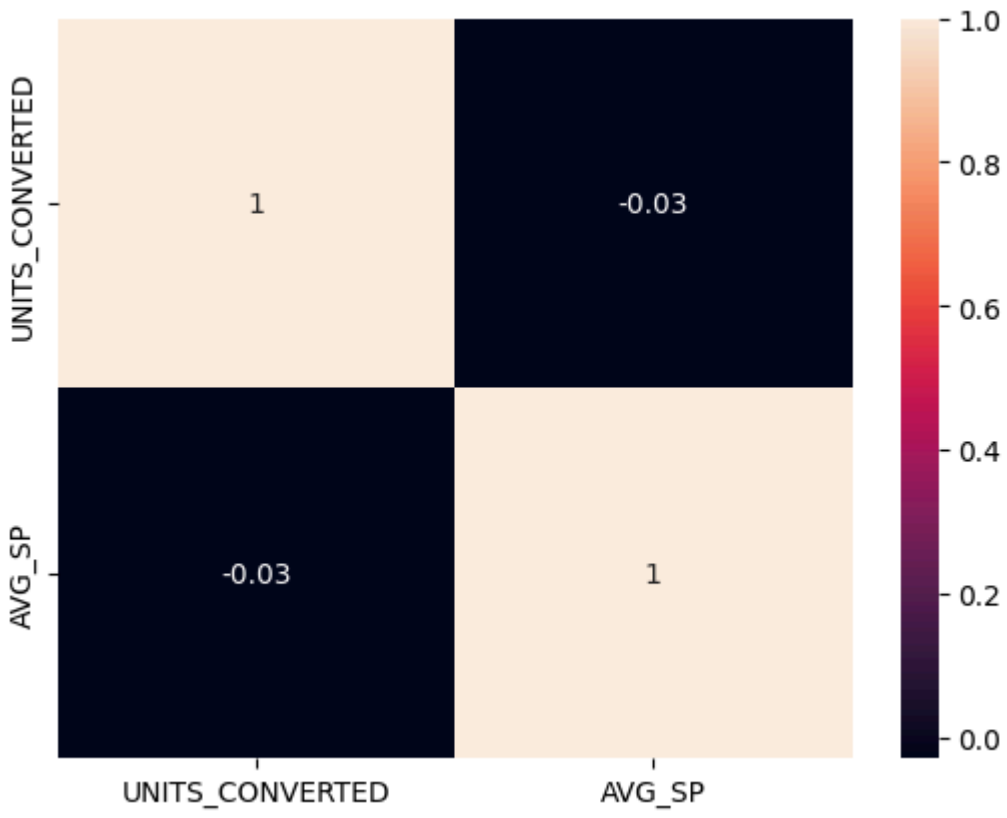



In [52]:
```
correlation = sku_data[['UNITS_CONVERTED', 'AVG_SP']].corr().iloc[0, 1]
correlation
```

Out[52]: -0.029776602675128886

In [53]:
```
sns.heatmap(sku_data[['UNITS_CONVERTED', 'AVG_SP']].corr(), annot=True)
```

Out[53]: <Axes: >




No correlation between Average Selling Price and Unit Conversion

# Summary

- General Metrics:
  - Revenue: Ranges from anomalies (negative values) to a maximum of $1,121,838.
  - Units Sold: Includes negative values; these need further review.
  - Views: Average of 856 per SKU, with a maximum of 176,162 views.
  - SKU_NAME: 452 unique SKUs; most frequent appears 123 times.
  - CATEGORY: Most common is "1000 Inputs" with over 26,000 records.

    - SUB_CATEGORY: Top subcategory is "1002 Mice," followed by Keyboards and Media Speaker Systems.

Insights by Analysis:

- Top SKUs by Units Sold and Revenue:
    - High-performing SKUs are likely ideal candidates for targeted marketing.
    - Premium offerings show high revenue despite low units sold.
- Revenue Trends Over Time:
    - Peaks in July indicate a likely sales event, confirmed between July 15-31, 2019.
- Views vs Units Sold:
    - Positive correlation: Higher views often translate to higher sales.
    - Exceptions highlight potential issues like pricing mismatches or availability problems.
- Slowest-Growing Subcategories:
    - Categories like "0460 Mice - DELETED" and "1008 Computer Peripherals Other" show zero relative growth, indicating areas of concern.
- Out-of-Stock Impact:
    - Negative correlation between out-of-stock events and revenue indicates inventory issues.
- Correlations:
    - Strong Correlations:
        - Ordered Units and Ordered Revenue
        - Views and Revenue
    - Weak Correlations:
        - REP_OOS and sales metrics like revenue and views.
- SKU-Level Insights:
    - Most Expensive SKU: C03CBL[721 with an average price of $1151.86.
    - Revenue-Generating SKUs: 80.97% of SKUs contributed revenue.
    - SKUs Stopped Selling After July: Identified SKUs, such as C17NEDU7P[, showed complete inactivity.
- Anomalies:
    - Negative Values: Found in revenue and units sold, indicating potential data quality issues.
    - Missing Values: 689 missing values in REP_OOS likely imply no out-of-stock events.
    - Duplicate Rows: Investigate to ensure data consistency.
- Other Observations:
    - No Correlation was found between average selling price and unit conversion for SKU C120[H:8NV.