In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

'''Import Libraries:Importing the required libraries: numpy, pandas, matplotlib, and s
These libraries are commonly used for data analysis and visualization tasks.'''
```

In [ ]:

```
#File structure columns with description
'''
event_time---> Time when event happened at (in UTC).
event_type---> Only one kind of event: purchase.
product_id---> ID of a product
category_id---> Product's category ID
category_code---> Product's category taxonomy (code name) if it was possible to make i
brand---> Downcased string of brand name. Can be missed.
price---> Float price of a product. Present.
user_id---> Permanent user ID.
** user_session**--->Temporary user's session ID. Same for each user's session. Is cha
```

In [3]:

```
# The code loads a dataset from a CSV file named '2020-Jan.csv' into a pandas DataFram

data=pd.read_csv('2020-Jan.csv')
data
```

Out[3]:

	event_time	event_type	product_id	category_id	category_code	brand	pri
0	2020-01-01 00:00:00 UTC	view	5809910	1602943681873052386	NaN	grattol	5.2
1	2020-01-01 00:00:09 UTC	view	5812943	1487580012121948301	NaN	kinetics	3.9
2	2020-01-01 00:00:19 UTC	view	5798924	1783999068867920626	NaN	zinger	3.9
3	2020-01-01 00:00:24 UTC	view	5793052	1487580005754995573	NaN	NaN	4.9
4	2020-01-01 00:00:25 UTC	view	5899926	2115334439910245200	NaN	NaN	3.9
...	...	...	...	...	...	...	
4264747	2020-01-31 23:59:52 UTC	remove_from_cart	5886774	1487580006317032337	NaN	NaN	1.5
4264748	2020-01-31 23:59:52 UTC	remove_from_cart	5886774	1487580006317032337	NaN	NaN	1.5
4264749	2020-01-31 23:59:53 UTC	view	5875432	2084144451428549153	NaN	NaN	2.0
4264750	2020-01-31 23:59:57 UTC	remove_from_cart	5820745	1487580006317032337	NaN	NaN	2.2
4264751	2020-01-31 23:59:58 UTC	remove_from_cart	5820745	1487580006317032337	NaN	NaN	2.2

4264752 rows × 9 columns

In [3]:

```
#Data Exploration: Check the shape of dataset number of rows and columns.

data.shape
```

Out[3]:
```
(4264752, 9)
```

In [4]:
```
#Data Exploration: check the top-5 rows and bottom 5-rows to geta quick overview of th

data.head()
```

Out[4]:

	event_time	event_type	product_id	category_id	category_code	brand	price	user_i
0	2020-01-01 00:00:00 UTC	view	5809910	1602943681873052386	NaN	grattol	5.24	59541462
1	2020-01-01 00:00:09 UTC	view	5812943	1487580012121948301	NaN	kinetics	3.97	59541464
2	2020-01-01 00:00:19 UTC	view	5798924	1783999068867920626	NaN	zinger	3.97	59541261
3	2020-01-01 00:00:24 UTC	view	5793052	1487580005754995573	NaN	NaN	4.92	42065286
4	2020-01-01 00:00:25 UTC	view	5899926	2115334439910245200	NaN	NaN	3.92	48407120

In [21]:
```
data.tail()
```

Out[21]:

	event_time	event_type	product_id	category_id	brand	price	user_id
4264747	2020-01-31 23:59:52 UTC	remove_from_cart	5886774	1487580006317032337	Unknown	1.59	607092857
4264748	2020-01-31 23:59:52 UTC	remove_from_cart	5886774	1487580006317032337	Unknown	1.59	607092857
4264749	2020-01-31 23:59:53 UTC	view	5875432	2084144451428549153	Unknown	2.05	423651741
4264750	2020-01-31 23:59:57 UTC	remove_from_cart	5820745	1487580006317032337	Unknown	2.22	607092857
4264751	2020-01-31 23:59:58 UTC	remove_from_cart	5820745	1487580006317032337	Unknown	2.22	607092857

In [4]:

```python
#Check the insights structure of the DataFrame, including the data types, memory usage

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4264752 entries, 0 to 4264751
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: 292.8+ MB
```

In [14]:

```python
#count the number of missing values in each column.

data.isnull().sum()
```

Out[14]:

```
event_time             0
event_type             0
product_id             0
category_id            0
category_code    4190033
brand            1775630
price                  0
user_id                0
user_session        1314
dtype: int64
```

In [81]:

```
# Finding the percentage of missing data in every column
#/data.shape[0] converts the count of missing values into a fraction of missing values
data_percentage=round(data.isnull().sum()/data.shape[0],2)*100
data_percentage
```

Out[81]:

```
event_time      0.0
event_type      0.0
product_id      0.0
category_id     0.0
brand           0.0
price           0.0
user_id         0.0
user_session    0.0
date            0.0
dtype: float64
```

In [15]:

```
#Drop the "category_code" column since 98% of the data is missing and it won't be usef

data.drop('category_code', axis=1, inplace=True)
```

In [16]:

```
#umber of columns
data.columns
```

Out[16]:

```
Index(['event_time', 'event_type', 'product_id', 'category_id', 'brand',
       'price', 'user_id', 'user_session'],
      dtype='object')
```

In [32]:

```
#Count the number of unique values in a column
data.nunique()
```

Out[32]:

```
event_time      1811522
event_type            4
product_id        45484
category_id         482
brand               257
price              2097
user_id          410018
user_session     965351
dtype: int64
```

In [30]:

```
# Get unique values from the 'event_type' column
unique_event_types = data['event_type'].unique()
unique_event_types
```

Out[30]:

```
array(['view', 'cart', 'remove_from_cart', 'purchase'], dtype=object)
```

In [17]:

```
unique_value_counts=data[data['event_type']=='view'].nunique()
unique_value_counts
```

Out[17]:

```
event_time      1313514
event_type            1
product_id        44280
category_id         481
brand               256
price              2086
user_id          397775
user_session     912885
dtype: int64
```

In [18]:

```
unique_value_counts=data[data['event_type']=='purchase'].nunique()
unique_value_counts
```

Out[18]:

```
event_time      32647
event_type          1
product_id      27376
category_id       380
brand             227
price            1401
user_id         28220
user_session    32385
dtype: int64
```

In [19]:

```
data.isnull().sum()
```

Out[19]:

```
event_time            0
event_type            0
product_id            0
category_id           0
brand           1775630
price                 0
user_id               0
user_session       1314
dtype: int64
```

In [20]:

```
# Handling Missing Values

# Fill missing values in 'brand' column with 'Unknown'
data['brand'].fillna('Unknown', inplace=True)

# Drop rows with missing values in 'user_session' column
data.dropna(subset=['user_session'], inplace=True)

# Final Check for Missing Values
print(data.isnull().sum())
```

```
event_time      0
event_type      0
product_id      0
category_id     0
brand           0
price           0
user_id         0
user_session    0
dtype: int64
```

In [55]:

```
sales_funnel = data['event_type'].value_counts()[['view', 'cart','remove_from_cart','p
print("Sales Funnel:")
print(sales_funnel)
```

```
Sales Funnel:
view                2037600
cart                1147259
remove_from_cart     814782
purchase             263797
Name: event_type, dtype: int64
```

In [70]:

```
data
```

Out[70]:

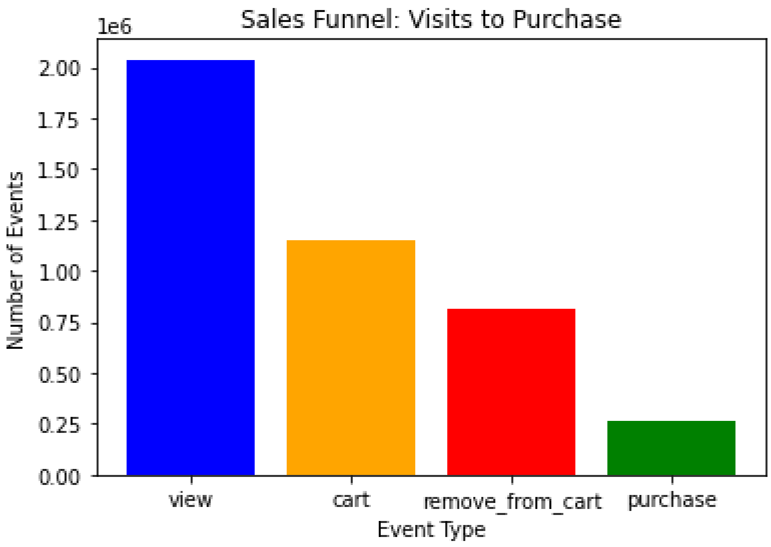
	event_time	event_type	product_id	category_id	brand	price	user
0	2020-01-01 00:00:00+00:00	view	5809910	1602943681873052386	grattol	5.24	5954146
1	2020-01-01 00:00:09+00:00	view	5812943	1487580012121948301	kinetics	3.97	5954146
2	2020-01-01 00:00:19+00:00	view	5798924	1783999068867920626	zinger	3.97	5954126
3	2020-01-01 00:00:24+00:00	view	5793052	1487580005754995573	Unknown	4.92	4206528
4	2020-01-01 00:00:25+00:00	view	5899926	2115334439910245200	Unknown	3.92	4840712
...	...	...	...	...	...	...	
4264747	2020-01-31 23:59:52+00:00	remove_from_cart	5886774	1487580006317032337	Unknown	1.59	6070928
4264748	2020-01-31 23:59:52+00:00	remove_from_cart	5886774	1487580006317032337	Unknown	1.59	6070928
4264749	2020-01-31 23:59:53+00:00	view	5875432	2084144451428549153	Unknown	2.05	4236517
4264750	2020-01-31 23:59:57+00:00	remove_from_cart	5820745	1487580006317032337	Unknown	2.22	6070928
4264751	2020-01-31 23:59:58+00:00	remove_from_cart	5820745	1487580006317032337	Unknown	2.22	6070928

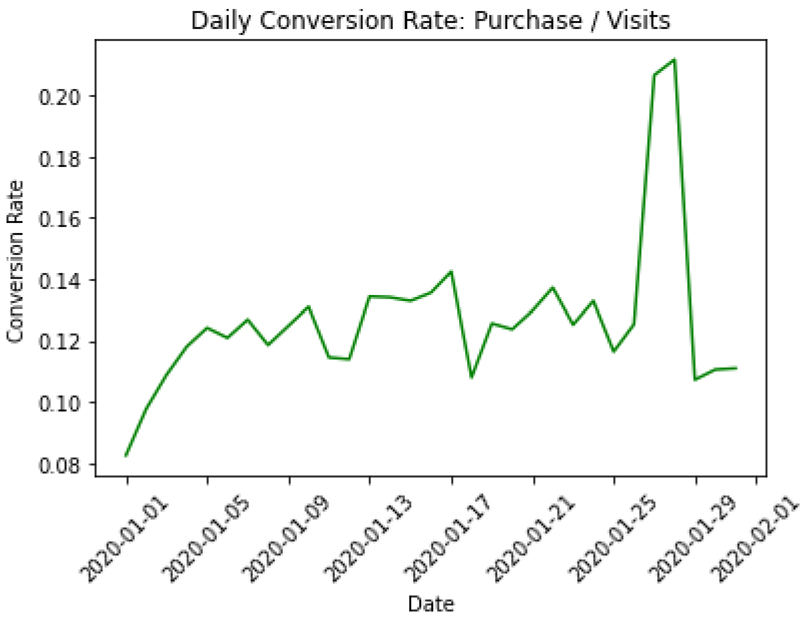
4263438 rows × 8 columns

In [56]:

```
# Convert 'event_time' column to datetime shows accurate time-based analysis.
data['event_time'] = pd.to_datetime(data['event_time'])
```

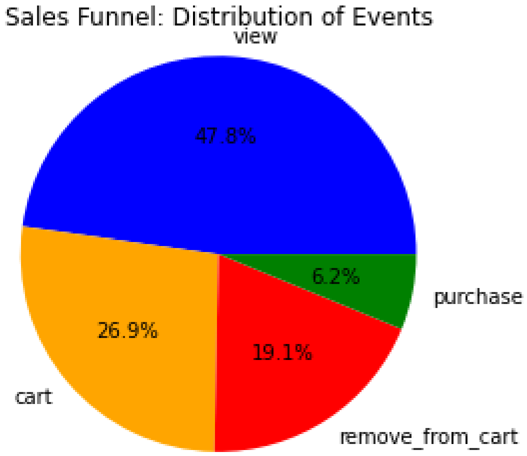
```
# Task 1: Sales Funnel (Visits to Purchase)
'''Count the occurrences of each event type and reindex to make sure all event types a
reindex method to ensure that all three event types are present .We fill any missing v

sales_funnel = data['event_type'].value_counts().reindex(['view', 'cart', 'remove_from

# Create a bar chart for the sales funnel
plt.bar(sales_funnel.index, sales_funnel.values, color=['blue', 'orange', 'red','green
plt.title('Sales Funnel: Visits to Purchase')
plt.xlabel('Event Type')
plt.ylabel('Number of Events')
plt.xticks(rotation=0)
plt.show()




# Task 2: Track Daily Conversion Rate (Purchase / Visits)

'''Resample the data to daily frequency and count the occurrences of each event type
resample method to aggregate the data to a daily frequency ('D')
count the occurrences of each event type on each day.
The result is a DataFrame where each column represents an event type.
We use unstack() to pivot the data, converting the event types from columns to rows, a

daily_events = data.set_index('event_time').resample('D')['event_type'].value_counts()

# Calculate the daily conversion rate (purchase / view)
daily_conversion_rate = daily_events['purchase'] / daily_events['view']

# Create a line chart for the daily conversion rate
plt.plot(daily_conversion_rate.index, daily_conversion_rate.values, color='green')
plt.title('Daily Conversion Rate: Purchase / Visits')
plt.xlabel('Date')
plt.ylabel('Conversion Rate')
plt.xticks(rotation=45)
plt.show()
```

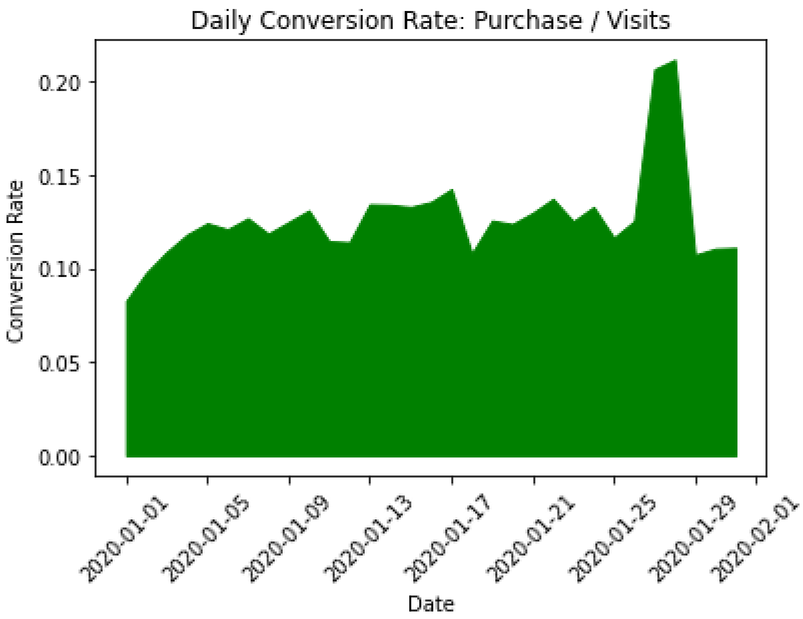




In [72]:

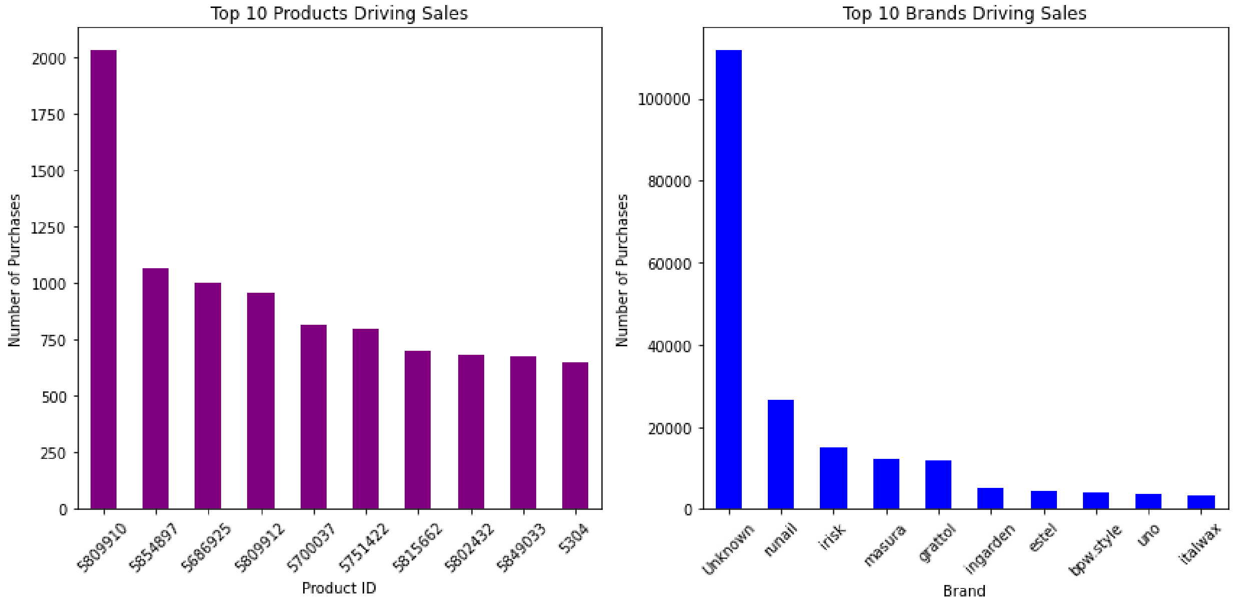
```
# Task 1: Sales Funnel (Visits to Purchase)
'''Count the occurrences of each event type and reindex to make sure all event types a
reindex method to ensure that all four event types are present .We fill any missing va
sales_funnel = data['event_type'].value_counts().reindex(['view', 'cart', 'remove_fron

# Create a pie chart for the sales funnel
plt.pie(sales_funnel, labels=sales_funnel.index, autopct='%1.1f%%', colors=['blue', 'c
plt.title('Sales Funnel: Distribution of Events')
plt.axis('equal')  # Equal aspect ratio ensures the pie chart is circular.
plt.show()



# Task 2: Track Daily Conversion Rate (Purchase / Visits)
# Create an area chart for the daily conversion rate
plt.fill_between(daily_conversion_rate.index, daily_conversion_rate.values, color='gre
plt.title('Daily Conversion Rate: Purchase / Visits')
plt.xlabel('Date')
plt.ylabel('Conversion Rate')
plt.xticks(rotation=45)
plt.show()
```





In [73]:

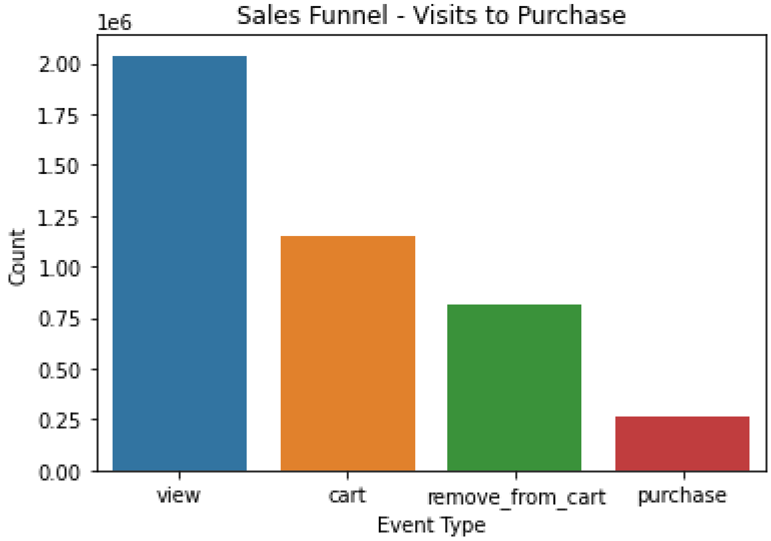
```
# Task 3:Understand products/brands which are driving the sales


# Filter data for 'purchase' events
purchase_data = data[data['event_type'] == 'purchase']

# Count the occurrences of each product and brand for purchases
top_products = purchase_data['product_id'].value_counts().head(10)
top_brands = purchase_data['brand'].value_counts().head(10)

# Create a side-by-side bar chart for top products and brands
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
top_products.plot(kind='bar', color='purple')
plt.title('Top 10 Products Driving Sales')
plt.xlabel('Product ID')
plt.ylabel('Number of Purchases')
plt.xticks(rotation=45)

plt.subplot(1, 2, 2)
top_brands.plot(kind='bar', color='blue')
plt.title('Top 10 Brands Driving Sales')
plt.xlabel('Brand')
plt.ylabel('Number of Purchases')
plt.xticks(rotation=45)

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

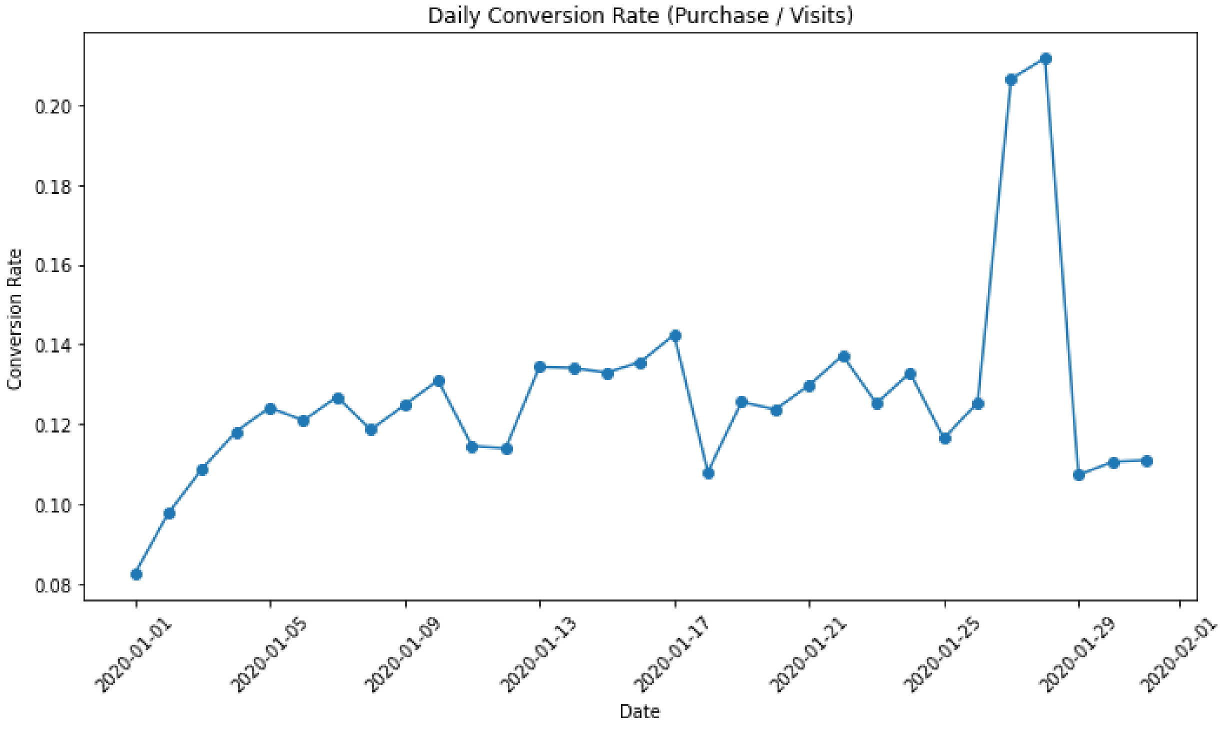


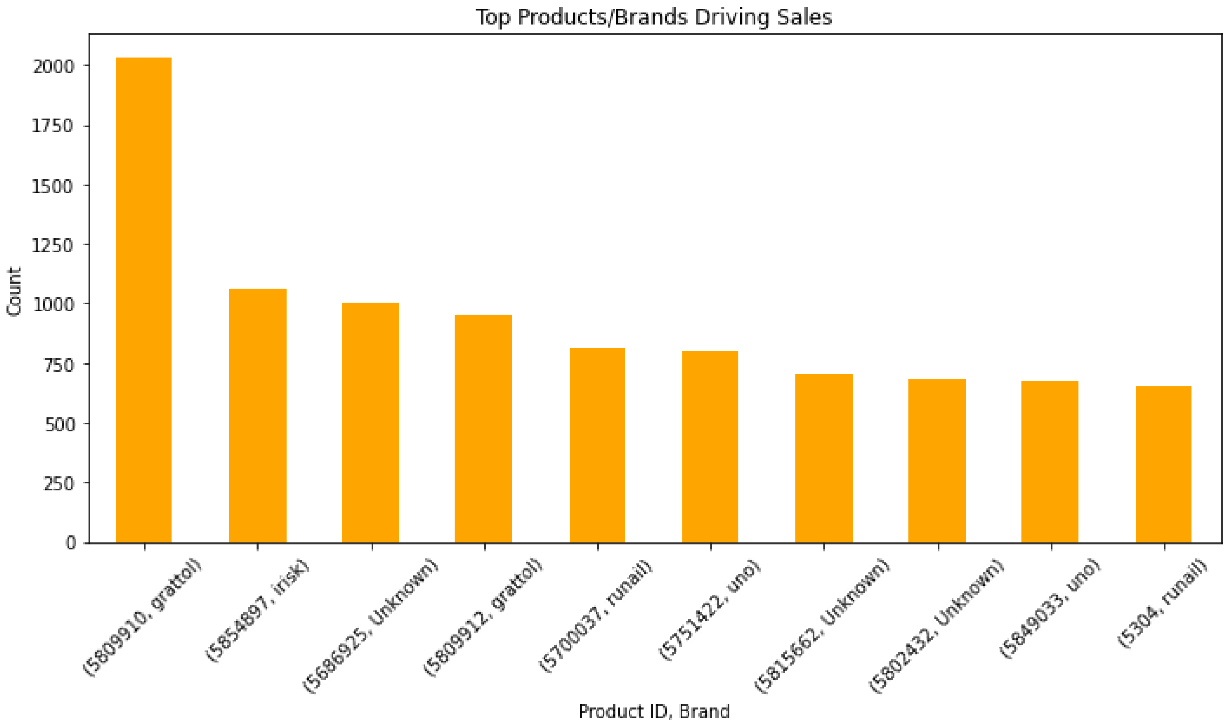
In [79]:

```
# Task 3: Top Products/Brands Driving Sales
top_products = data[data['event_type'] == 'purchase'].groupby(['product_id', 'brand',
print("\nTop Products/Brands Driving Sales:")
print(top_products)

# Visualization: Sales Funnel
plt.figure(figsize=(6, 4))
sns.countplot(data=data, x='event_type', order=data['event_type'].value_counts().index
plt.title('Sales Funnel - Visits to Purchase')
plt.xlabel('Event Type')
plt.ylabel('Count')
plt.show()

# Visualization: Daily Conversion Rate
plt.figure(figsize=(10, 6))
conversion_rate.plot(marker='o')
plt.title('Daily Conversion Rate (Purchase / Visits)')
plt.xlabel('Date')
plt.ylabel('Conversion Rate')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Visualization: Top Products/Brands Driving Sales
plt.figure(figsize=(10, 6))
top_products.plot(kind='bar', color='orange')
plt.title('Top Products/Brands Driving Sales')
plt.xlabel('Product ID, Brand')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Top Products/Brands Driving Sales:
product_id  brand
5809910     grattol    2029
5854897     irisk      1065
5686925     Unknown    1001
5809912     grattol     953
5700037     runail      813
5751422     uno         798
5815662     Unknown     702
5802432     Unknown     683
5849033     uno         676
5304        runail      652
dtype: int64
```






Top Products/Brands Driving Sales
Count
2000
1750
1500
1250
1000
750
500
250
0
(5809910, grattol)
(5854897, irisk)
(5686925, Unknown)
(5809912, grattol)
(5700037, runail)
(5751422, uno)
(5815662, Unknown)
(5802432, Unknown)
(5849033, uno)
(5304, runail)
Product ID, Brand