Exploration Data Analysis (EDA) - Diagnostic Analysis

Product Analysis

Which product had the highest sales, and what factors do you believe contributed to its success?

1. Overview

In today's data-driven marketplace, identifying top-performing products is vital for shaping profitable business strategies. This project dives into Exploratory Data Analysis (EDA) with a focus on Product Analysis, aimed at uncovering which products are driving the most sales and the underlying factors contributing to their success. Leveraging purchase data, we aggregated and visualized product performance to provide actionable insights for inventory optimization, marketing alignment, and revenue growth.

2. Goal

- To analyze product-level sales data and identify the best-selling products.
- To visualize product performance for clearer insights into customer preferences.
- To explore possible factors contributing to the success of high-performing products.
- To support data-driven decisions in product management and marketing strategies.
- To uncover patterns and anomalies that may inform future forecasting and planning.

3. Business Challenge

- Difficulty in identifying which products are driving the highest customer demand.
- Lack of clarity on whether sales are concentrated among a few products or spread evenly.
- Uncertainty about the effectiveness of current marketing and inventory strategies.
- The need for visual and data-backed evidence to support product-related business decisions.

4. Methodology

- Perform Exploratory Data Analysis (EDA) on sales data, focusing on product-level performance.

- Aggregate units purchased by product using groupby to rank products by total sales volume.
- Use bar charts to visualize and compare product sales in a clear and digestible format.
- Investigate potential contributing factors such as pricing, availability, marketing, and customer preference.
- Present insights in a business-friendly format to guide product planning and strategic decisions.

Import necessary libraries

In [9]:
```
import pandas as pd
import os
import glob
```

Combine the sales data from all months into a single consolidated CSV file

In [11]:
```
folder_path = r"C:\Monthly_Sales"

# Retrieve all CSV files from the folder using glob
all_files = glob.glob(os.path.join(folder_path, "*.csv"))

# All CSV files combined as one DataFrame
all_data = pd.concat([pd.read_csv(file) for file in all_files], ignore_index=True)

# Merged DataFrame saved into a new CSV
output_file = os.path.join(folder_path, "all_data.csv")
all_data.to_csv(output_file, index=False)

print("All files integrated into:", output_file)
```

```
All files integrated into: C:\Monthly_Sales\all_data.csv
```

Load the updated DataFrame

In [13]:
```
# Skip Blank Rows if present in the dataset

df = pd.read_csv(r'C:\Monthly_Sales\all_data.csv', skip_blank_lines=True)
df.head()
```

Out[13]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address
0	175667	iPhone	1	700.0	04/24/24 19:12	135 Meadow St, Boston, MA 02215
1	175668	AA Batteries (4-pack)	1	5.84	04/20/24 13:45	592 4th St, San Francisco, CA 94016
2	175669	AA Batteries (4-pack)	1	5.84	04/28/24 09:17	632 Park St, Dallas, TX 75001
3	175670	AA Batteries (4-pack)	2	5.84	04/23/24 14:06	131 Pine St, San Francisco, CA 94016
4	175671	Samsung Odyssey Monitor	1	409.99	04/23/24 12:13	836 Forest St, Boston, MA 02215

In [14]:
```
df.shape
```

Out[14]:
```
(8799083, 6)
```

Data Cleaning Process

Thoroughly clean and standardize the data to eliminate errors, ensure consistency, and build a solid foundation for meaningful insights.

Find and remove rows with NaN values

In [17]:
```
df.isna().sum()
```

Out[17]:
```
Order ID            22848
Product Name        22848
Units Purchased     22850
Unit Price          22850
Order Date          22851
Delivery Address    22852
dtype: int64
```

In [18]:
```
# If Nan value is present in Order ID and Unit Purchased, it will be impossible to
# Therefore, drop Nan values in Order ID and Units Purchased.

df.dropna(subset=['Order ID', 'Units Purchased'], inplace=True)
```

In [19]:
```
# Check if Nan value is present

df.isna().sum()
```

Out[19]:
```
Order ID            0
Product Name        0
Units Purchased     0
Unit Price          0
Order Date          1
Delivery Address    2
dtype: int64
```

In [20]:
```
# Further check if any NaN values or blank rows are present

blank_rows_na = df[df.isnull().any(axis=1)]
blank_rows_na
```

Out[20]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address
2195228	Charging Cable	1	14.95	05/24/24 07:04	852 Hickory St, San Francisco, CA 94016	NaN
3001506	150766	iPhone	1	7	NaN	NaN

Find and remove rows with duplicate values

In [22]:
```
# Find duplicate values

df.duplicated()
```

Out[22]:
```
0          False
1          False
2          False
3          False
4          False
           ...
8799078     True
8799079     True
8799080     True
8799081     True
8799082     True
Length: 8776233, dtype: bool
```

In [23]:
```
# Remove duplicated values

df.drop_duplicates(inplace = True)
```

In [24]:
```
# Check again for duplicated values

df.duplicated()
```

Out[24]:
```
0          False
1          False
2          False
3          False
4          False
           ...
172530     False
2195228    False
3001506    False
6370083    False
6403571    False
Length: 171546, dtype: bool
```

Verify and fix incorrect data types in the dataset

In [26]:
```
# check for data types

df.dtypes
```

Out[26]:
```
Order ID            object
Product Name        object
Units Purchased     object
Unit Price          object
Order Date          object
Delivery Address    object
dtype: object
```

Fix incorrect data types

In [28]:
```
df['Order Date'] = pd.to_datetime(df['Order Date'], format='%m/%d/%y %H:%M', errors

df['Units Purchased'] = pd. to_numeric(df['Units Purchased'], errors='coerce')

df['Unit Price'] = pd. to_numeric(df['Unit Price'], errors='coerce')
```

In [29]:
```
# Verify the presence of NaN values remaining in the columns as a result of using e

df.isna().sum()
```

Out[29]:
```
Order ID            0
Product Name        0
Units Purchased     1
Unit Price          2
Order Date          3
Delivery Address    2
dtype: int64
```

In [30]:
```
df = df.dropna()
```

Change the data type to optimize memory usage (Optional)

In [32]:
```
df['Order ID'] = pd.to_numeric(df['Order ID'], downcast='integer')
df['Product Name'] = df['Product Name'].astype('category')
```

```
df['Units Purchased'] = df['Units Purchased']. astype('int8')
df['Unit Price'] = pd.to_numeric(df['Unit Price'], downcast='float')
df['Delivery Address'] = df['Delivery Address'].astype('category')
```

Expand the dataset with supplementary columns

Add month column

In [35]:
```
df['Month'] = df['Order Date'].dt.month
df
```

Out[35]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month
0	175667	iPhone	1	700.00000	2024-04-24 19:12:00	135 Meadow St, Boston, MA 02215	4
1	175668	AA Batteries (4-pack)	1	5.84000	2024-04-20 13:45:00	592 4th St, San Francisco, CA 94016	4
2	175669	AA Batteries (4-pack)	1	5.84000	2024-04-28 09:17:00	632 Park St, Dallas, TX 75001	4
3	175670	AA Batteries (4-pack)	2	5.84000	2024-04-23 14:06:00	131 Pine St, San Francisco, CA 94016	4
4	175671	Samsung Odyssey Monitor	1	409.98999	2024-04-23 12:13:00	836 Forest St, Boston, MA 02215	4
...	...	...	...	...	...	...	...
172528	248378	Google Phone	1	600.00000	2024-09-02 08:53:00	668 Wilson St, Boston, MA 02215	9
172529	248379	Alienware Monitor	1	400.98999	2024-09-04 22:58:00	466 2nd St, Boston, MA 02215	9
172530	248380	AAA Batteries (4-pack)	1	4.99000	2024-09-04 13:09:00	133 Walnut St, Seattle, WA 98101	9
6370083	252436	Apple Airpods Headphones	1	150.00000	2024-10-14 16:44:00	740 Dogwood St, Boston, \rA 02215	10
6403571	233092	USB-C Charging Cable	1	11.95000	2024-08-28 12:39:00	740 Dogwood St, Boston, \rA 02215	8

171543 rows × 7 columns

In [36]:
```
df['Month Name'] = df['Order Date'].dt.strftime('%B')
df
```

Out[36]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	M N
0	175667	iPhone	1	700.00000	2024-04-24 19:12:00	135 Meadow St, Boston, MA 02215	4	
1	175668	AA Batteries (4-pack)	1	5.84000	2024-04-20 13:45:00	592 4th St, San Francisco, CA 94016	4	
2	175669	AA Batteries (4-pack)	1	5.84000	2024-04-28 09:17:00	632 Park St, Dallas, TX 75001	4	
3	175670	AA Batteries (4-pack)	2	5.84000	2024-04-23 14:06:00	131 Pine St, San Francisco, CA 94016	4	
4	175671	Samsung Odyssey Monitor	1	409.98999	2024-04-23 12:13:00	836 Forest St, Boston, MA 02215	4	
...	...	...	...	...	...	...	...	
172528	248378	Google Phone	1	600.00000	2024-09-02 08:53:00	668 Wilson St, Boston, MA 02215	9	Septer
172529	248379	Alienware Monitor	1	400.98999	2024-09-04 22:58:00	466 2nd St, Boston, MA 02215	9	Septer
172530	248380	AAA Batteries (4-pack)	1	4.99000	2024-09-04 13:09:00	133 Walnut St, Seattle, WA 98101	9	Septer
6370083	252436	Apple Airpods Headphones	1	150.00000	2024-10-14 16:44:00	740 Dogwood St, Boston, \rA 02215	10	Oct

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	M N
6403571	233092	USB-C Charging Cable	1	11.95000	2024-08-28 12:39:00	740 Dogwood St, Boston, \rA 02215	8	Au

171543 rows × 8 columns

Add week day column

In [38]:
```
df['Day of Week'] = df['Order Date'].dt.strftime('%a')
df
```

Out[38]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	M N
0	175667	iPhone	1	700.00000	2024-04-24 19:12:00	135 Meadow St, Boston, MA 02215	4	
1	175668	AA Batteries (4-pack)	1	5.84000	2024-04-20 13:45:00	592 4th St, San Francisco, CA 94016	4	
2	175669	AA Batteries (4-pack)	1	5.84000	2024-04-28 09:17:00	632 Park St, Dallas, TX 75001	4	
3	175670	AA Batteries (4-pack)	2	5.84000	2024-04-23 14:06:00	131 Pine St, San Francisco, CA 94016	4	
4	175671	Samsung Odyssey Monitor	1	409.98999	2024-04-23 12:13:00	836 Forest St, Boston, MA 02215	4	
...	...	...	...	...	...	...	...	
172528	248378	Google Phone	1	600.00000	2024-09-02 08:53:00	668 Wilson St, Boston, MA 02215	9	Septer
172529	248379	Alienware Monitor	1	400.98999	2024-09-04 22:58:00	466 2nd St, Boston, MA 02215	9	Septer
172530	248380	AAA Batteries (4-pack)	1	4.99000	2024-09-04 13:09:00	133 Walnut St, Seattle, WA 98101	9	Septer
6370083	252436	Apple Airpods Headphones	1	150.00000	2024-10-14 16:44:00	740 Dogwood St, Boston, \rA 02215	10	Oct

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	M N
6403571	233092	USB-C Charging Cable	1	11.95000	2024-08-28 12:39:00	740 Dogwood St, Boston, \rA 02215	8	Au

171543 rows × 9 columns

Add hour column

In [40]:
```
df['Hour'] = df['Order Date'].dt.hour
df
```

Out[40]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	M N
0	175667	iPhone	1	700.00000	2024-04-24 19:12:00	135 Meadow St, Boston, MA 02215	4	
1	175668	AA Batteries (4-pack)	1	5.84000	2024-04-20 13:45:00	592 4th St, San Francisco, CA 94016	4	
2	175669	AA Batteries (4-pack)	1	5.84000	2024-04-28 09:17:00	632 Park St, Dallas, TX 75001	4	
3	175670	AA Batteries (4-pack)	2	5.84000	2024-04-23 14:06:00	131 Pine St, San Francisco, CA 94016	4	
4	175671	Samsung Odyssey Monitor	1	409.98999	2024-04-23 12:13:00	836 Forest St, Boston, MA 02215	4	
...	...	...	...	...	...	...	...	
172528	248378	Google Phone	1	600.00000	2024-09-02 08:53:00	668 Wilson St, Boston, MA 02215	9	Septer
172529	248379	Alienware Monitor	1	400.98999	2024-09-04 22:58:00	466 2nd St, Boston, MA 02215	9	Septer
172530	248380	AAA Batteries (4-pack)	1	4.99000	2024-09-04 13:09:00	133 Walnut St, Seattle, WA 98101	9	Septer
6370083	252436	Apple Airpods Headphones	1	150.00000	2024-10-14 16:44:00	740 Dogwood St, Boston, \rA 02215	10	Oct

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	M N
6403571	233092	USB-C Charging Cable	1	11.95000	2024-08-28 12:39:00	740 Dogwood St, Boston, \rA 02215	8	Au

171543 rows × 10 columns

Add city column

In [42]:
```
def city(address):
    return address.split(",")[1].strip(" ")

def state_abbrev(address):
    return address.split(",")[2].split(" ")[1]

df['City'] = df['Delivery Address'].apply(lambda x: f"{city(x)}  ({state_abbrev(x)}
df.head()
```

Out[42]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Name	Day of Week
0	175667	iPhone	1	700.00000	2024-04-24 19:12:00	135 Meadow St, Boston, MA 02215	4	April	Wed
1	175668	AA Batteries (4-pack)	1	5.84000	2024-04-20 13:45:00	592 4th St, San Francisco, CA 94016	4	April	Sat
2	175669	AA Batteries (4-pack)	1	5.84000	2024-04-28 09:17:00	632 Park St, Dallas, TX 75001	4	April	Sun
3	175670	AA Batteries (4-pack)	2	5.84000	2024-04-23 14:06:00	131 Pine St, San Francisco, CA 94016	4	April	Tue
4	175671	Samsung Odyssey Monitor	1	409.98999	2024-04-23 12:13:00	836 Forest St, Boston, MA 02215	4	April	Tue

Organize Data by Order Date Chronologically and Reindex

In [44]:
```
df = df.sort_values(by = 'Order Date')
df
```

Out[44]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Name
78282	160155	Alienware Monitor	1	400.989990	2024-01-01 05:04:00	765 Ridge St, Portland, OR 97035	1	January
68761	151041	AAA Batteries (4-pack)	1	4.990000	2024-01-01 05:04:00	964 Lakeview St, Atlanta, GA 30301	1	January
64303	146765	AAA Batteries (4-pack)	1	4.990000	2024-01-01 05:20:00	546 10th St, San Francisco, CA 94016	1	January
63092	145617	Amana Washing Machine	1	600.000000	2024-01-01 05:24:00	961 Meadow St, Portland, OR 97035	1	January
74502	156535	iPhone	1	700.000000	2024-01-01 05:45:00	451 Elm St, Los Angeles, CA 90001	1	January
...	...	...	...	...	...	...	...	...
44457	297748	iPhone	1	700.000000	2025-01-01 02:37:00	258 Forest St, Los Angeles, CA 90001	1	January
30663	284606	Bose SoundSport Headphones	1	99.989998	2025-01-01 02:50:00	211 Johnson St, Boston, MA 02215	1	January
49246	302330	AA Batteries (4-pack)	1	5.840000	2025-01-01 03:03:00	665 6th St, San Francisco, CA 94016	1	January
30770	284711	AA Batteries (4-pack)	1	5.840000	2025-01-01 03:19:00	250 8th St, San Francisco, CA 94016	1	January

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Name
50619	303626	USB-C Charging Cable	3	11.950000	2025-01-01 04:43:00	651 Lakeview St, Dallas, TX 75001	1	January

171543 rows × 11 columns

In [45]:
```
df = df.reset_index(drop=True)
df
```

Out[45]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Nam
0	160155	Alienware Monitor	1	400.989990	2024-01-01 05:04:00	765 Ridge St, Portland, OR 97035	1	Januar
1	151041	AAA Batteries (4-pack)	1	4.990000	2024-01-01 05:04:00	964 Lakeview St, Atlanta, GA 30301	1	Januar
2	146765	AAA Batteries (4-pack)	1	4.990000	2024-01-01 05:20:00	546 10th St, San Francisco, CA 94016	1	Januar
3	145617	Amana Washing Machine	1	600.000000	2024-01-01 05:24:00	961 Meadow St, Portland, OR 97035	1	Januar
4	156535	iPhone	1	700.000000	2024-01-01 05:45:00	451 Elm St, Los Angeles, CA 90001	1	Januar
...	...	...	...	...	...	...	...	.
171538	297748	iPhone	1	700.000000	2025-01-01 02:37:00	258 Forest St, Los Angeles, CA 90001	1	Januar
171539	284606	Bose SoundSport Headphones	1	99.989998	2025-01-01 02:50:00	211 Johnson St, Boston, MA 02215	1	Januar
171540	302330	AA Batteries (4-pack)	1	5.840000	2025-01-01 03:03:00	665 6th St, San Francisco, CA 94016	1	Januar
171541	284711	AA Batteries (4-pack)	1	5.840000	2025-01-01 03:19:00	250 8th St, San Francisco, CA 94016	1	Januar

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Mont Nam
171542	303626	USB-C Charging Cable	3	11.950000	2025-01-01 04:43:00	651 Lakeview St, Dallas, TX 75001	1	Januar

171543 rows × 11 columns

Add Total Sales column

In [47]:
```
df['Total Sales'] = df['Units Purchased'] * df['Unit Price']
df.head()
```

Out[47]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Name	Day of Week
0	160155	Alienware Monitor	1	400.98999	2024-01-01 05:04:00	765 Ridge St, Portland, OR 97035	1	January	Mon
1	151041	AAA Batteries (4-pack)	1	4.99000	2024-01-01 05:04:00	964 Lakeview St, Atlanta, GA 30301	1	January	Mon
2	146765	AAA Batteries (4-pack)	1	4.99000	2024-01-01 05:20:00	546 10th St, San Francisco, CA 94016	1	January	Mon
3	145617	Amana Washing Machine	1	600.00000	2024-01-01 05:24:00	961 Meadow St, Portland, OR 97035	1	January	Mon
4	156535	iPhone	1	700.00000	2024-01-01 05:45:00	451 Elm St, Los Angeles, CA 90001	1	January	Mon

Format Unit Price and Total Sales to 2 decimal places

In [49]:
```
df['Unit Price'] = df['Unit Price'].apply(lambda x: "%.2f" % x)
```

In [50]:
```
df['Total Sales'] = df['Total Sales'].apply(lambda x: "%.2f" % x)
df.head()
```

Out[50]:

	Order ID	Product Name	Units Purchased	Unit Price	Order Date	Delivery Address	Month	Month Name	Day of Week	Hc
0	160155	Alienware Monitor	1	400.99	2024-01-01 05:04:00	765 Ridge St, Portland, OR 97035	1	January	Mon	
1	151041	AAA Batteries (4-pack)	1	4.99	2024-01-01 05:04:00	964 Lakeview St, Atlanta, GA 30301	1	January	Mon	
2	146765	AAA Batteries (4-pack)	1	4.99	2024-01-01 05:20:00	546 10th St, San Francisco, CA 94016	1	January	Mon	
3	145617	Amana Washing Machine	1	600.00	2024-01-01 05:24:00	961 Meadow St, Portland, OR 97035	1	January	Mon	
4	156535	iPhone	1	700.00	2024-01-01 05:45:00	451 Elm St, Los Angeles, CA 90001	1	January	Mon	

Format Unit Price and Total Sales to numeric

In [52]:
```
df['Unit Price'] = pd.to_numeric(df['Unit Price'])
df['Total Sales'] = pd.to_numeric(df['Total Sales'])
```

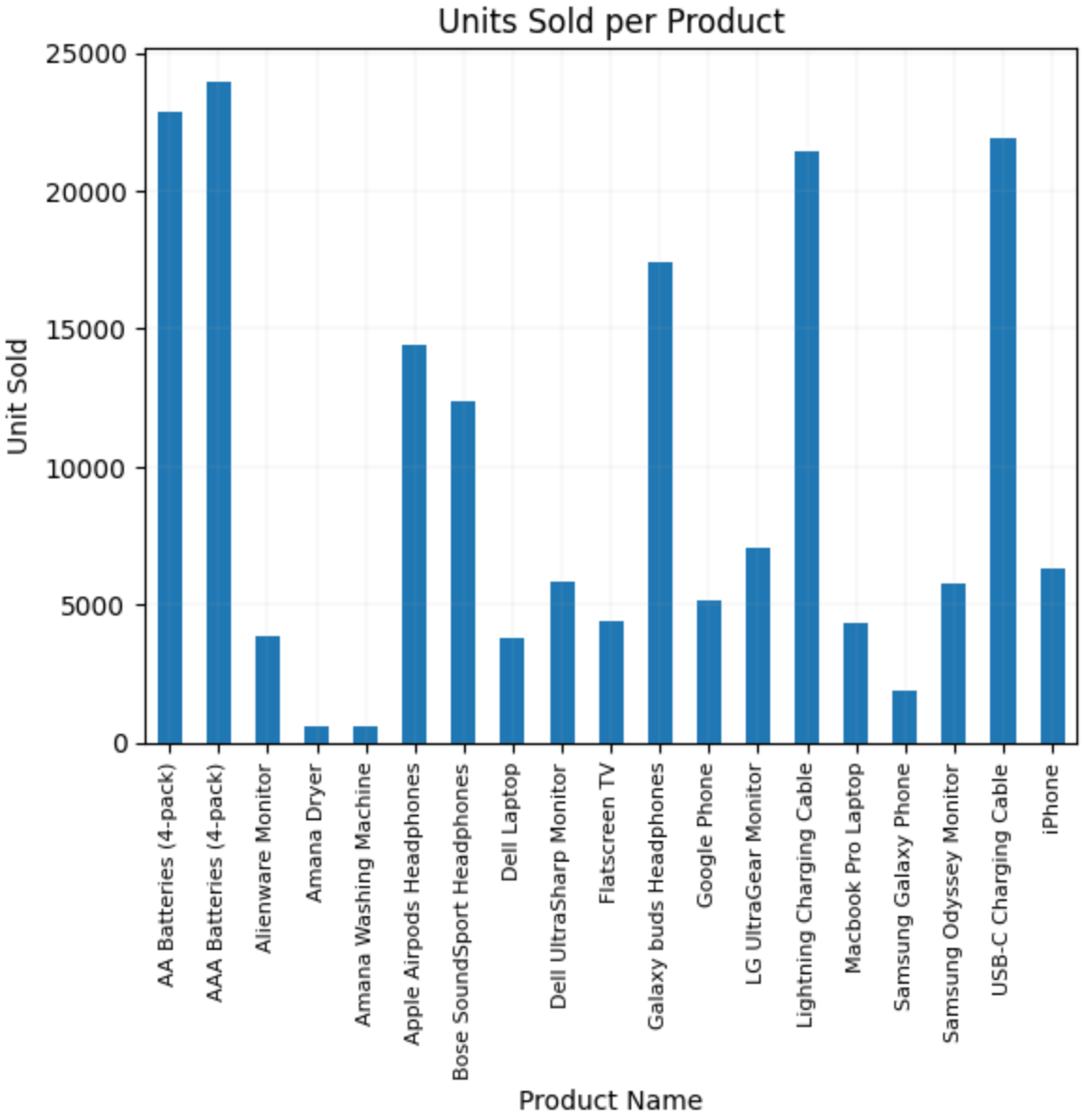
Sum of Units Purchased per Product

In [54]:
```
product_df = df.groupby('Product Name', observed=False)['Units Purchased'].sum()

sorted_product_df = product_df.sort_values(ascending=False)
sorted_product_df
```

Out[54]:
```
Product Name
AAA Batteries (4-pack)        23970
AA Batteries (4-pack)         22830
USB-C Charging Cable          21927
Lightning Charging Cable      21442
Galaxy buds Headphones        17384
Apple Airpods Headphones      14445
Bose SoundSport Headphones    12386
LG UltraGear Monitor           7096
iPhone                         6294
Dell UltraSharp Monitor        5820
Samsung Odyssey Monitor        5741
Google Phone                   5174
Flatscreen TV                  4422
Macbook Pro Laptop             4344
Alienware Monitor              3861
Dell Laptop                    3798
Samsung Galaxy Phone           1859
Amana Dryer                     622
Amana Washing Machine           611
Name: Units Purchased, dtype: int64
```

Plot Units Sold per Product

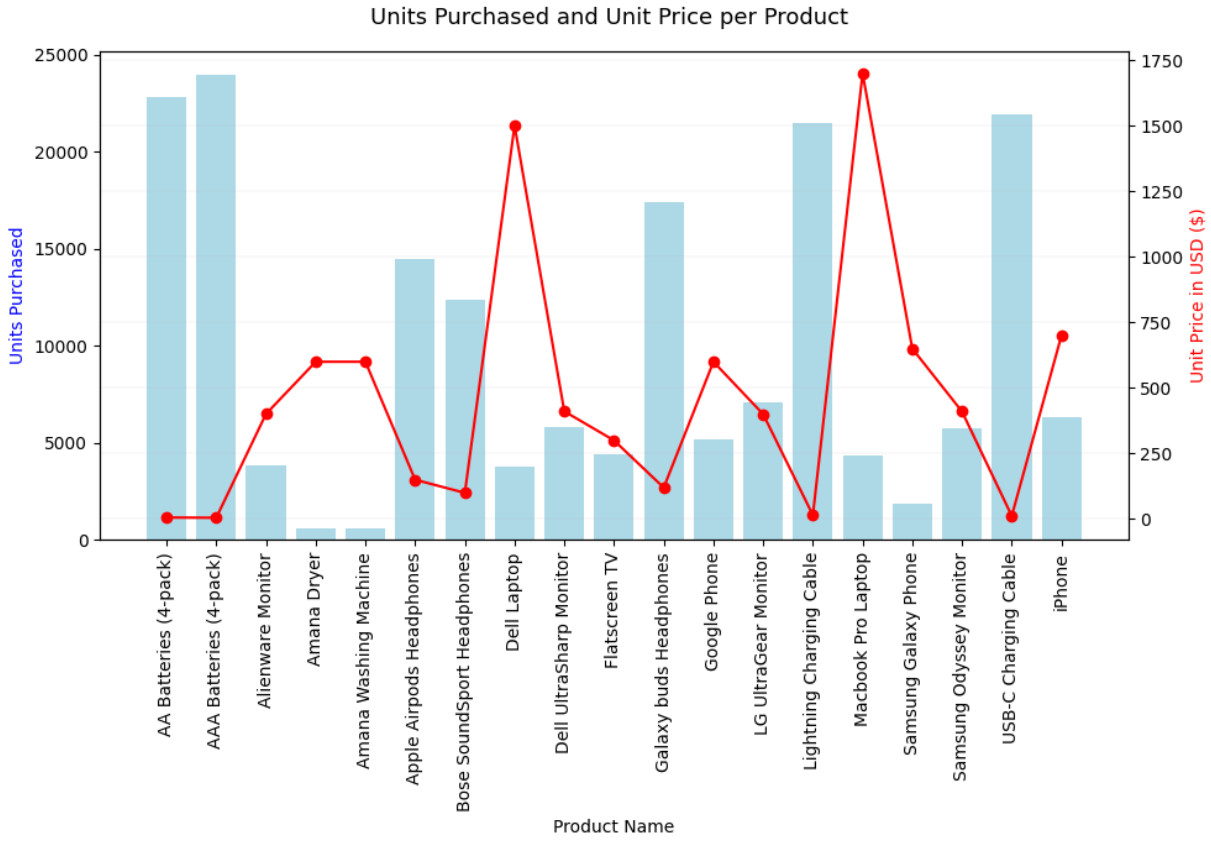
In [56]:
```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

product_df = df.groupby('Product Name', observed=False)['Units Purchased'].sum()

product_df.plot(kind='bar', title="Units Sold per Product")

plt.xlabel('Product Name')
plt.ylabel('Unit Sold')
plt.xticks(rotation='vertical', size=8)
plt.grid(linewidth=0.1)

plt.savefig(r"C:/Users/DELL/OneDrive - COVENANT UNIVERSITY/Desktop/1/Product Analys

plt.show()
```



Dual Axis Plot of Units Purchased and Unit Price per Product

In [58]:
```
product_df = df.groupby('Product Name', observed=False)['Units Purchased'].sum()
price_df = df.groupby('Product Name', observed=False)['Unit Price'].mean()

# first axis and figure size
fig, ax1 = plt.subplots(figsize=(10, 7))

# Bar chart and first y-axis for Units Purchased
ax1.bar(product_df.index, product_df, color='lightblue', label='Units Purchased')
ax1.set_xlabel('Product Name')
ax1.set_ylabel('Units Purchased', color='b')
ax1.tick_params(axis='x', rotation=90)  # Rotate x-axis labels for better visibilit

# Line chart and the second y-axis for Unit Price
ax2 = ax1.twinx()
```

```
ax2.plot(price_df.index, price_df, color='r', label='Unit Price', marker='o')
ax2.set_ylabel('Unit Price in USD ($)', color='r')

plt.grid(linewidth=0.1)
fig.suptitle('Units Purchased and Unit Price per Product', fontsize=13)

plt.tight_layout()
plt.savefig(r"C:/Users/DELL/OneDrive - COVENANT UNIVERSITY/Desktop/1/Product Analys

plt.show()
```



Key Insights

1. Top-Selling Product:

AAA Batteries (4-pack) lead the pack with 23,970 units sold, closely followed by AA Batteries (4-pack) and USB-C Charging Cables.

2. Accessory Dominance:

The top 5 products are all low-cost, high-demand accessories (batteries, cables, headphones) rather than high-ticket items like phones or laptops.

3. Price vs. Volume Trend:

Low-cost items significantly outperform premium electronics in volume sold. For example, AAA Batteries outsold the iPhone nearly 4x and the MacBook Pro 5.5x.

4. Brand Appeal in Audio:

In the headphone category, Galaxy Buds outperform Apple Airpods and Bose SoundSport, suggesting strong market preference or price-value advantage.

5. Low Sales in High-Value Appliances:

Items like Amana Dryers and Washing Machines had the lowest sales, indicating either limited demand, higher price sensitivity, or fewer purchase occasions.

Strategic Recommendations

1. Double Down on Fast-Moving Accessories: Increase marketing and shelf space for top-performing accessories like batteries and charging cables. Consider bundling them with other products for upselling.
2. Leverage Volume Leaders for Cross-Selling: Use high-demand products (e.g., AAA/AA batteries) as entry points to recommend related items (e.g., remotes, toys, small electronics) during checkout.
3. Promote Mid-Tier Headphones More Aggressively: Galaxy Buds and Apple Airpods are clear leaders in the audio space. Offer discounts, bundles, or financing to push premium brands like Bose or increase sales further.
4. Review Pricing & Promotion of High-Ticket Items: Laptops, phones, and monitors show relatively lower sales. Reassess marketing efforts, financing options, or promotional campaigns to boost their visibility and affordability.
5. Investigate Appliance Sales Channels: The extremely low sales for dryers and washing machines may indicate channel mismatch (e.g., people buying these elsewhere), poor visibility, or low stock levels, worth further diagnostic analysis.