Introduction

Global Superstore Dataset Analysis

The data contains information about sales transactions, including date, product, quantity, and revenue. This Jupyter Notebook explores the Global Superdf dataset with the aim of uncovering insights into sales patterns and product performance. The analysis is divided into two primary sections:

- **Data Exploration:** This phase focuses on preparing the data for analysis by handling missing values, inconsistencies, and outliers. Additionally, exploratory techniques will be employed to understand the dataset's characteristics and identify potential trends.
- **Analysis and Visualization:** In this section, in-depth analysis will be conducted to answer specific business questions. Visualizations will be created to effectively communicate findings and support data-driven decision-making.

1. Data Exploration

1.1 Import Necessary Libraries

- Import essential Python libraries for data manipulation, analysis, and visualization: Pandas, NumPy, Matplotlib, and Seaborn.

1.2 Load the Dataset

- Read the data from a CSV file format into a Pandas DataFrame.

1.3 Initial Data Overview

- Get a basic understanding of the dataset:
  - View the first few rows of data.
  - Get information about column names, data types, and missing values.
  - Calculate summary statistics for numerical columns.
  - Determine the overall size of the dataset.

1.4 Handle Missing Values

- Identify columns with missing data.
- Decide on a strategy to handle missing values.

1.5 Check for Duplicates

- Identify and remove any duplicate rows in the dataset.

1.6 Data Cleaning and Formatting

- Clean and prepare the data for analysis:
  - Rename columns for clarity.
  - Convert date columns to appropriate datetime format.
  - Create new columns if necessary.
  - Convert data types as needed.

In [ ]:
```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import ticker as mtick
import seaborn as sns
```

In [ ]:
```
# import datasets
df = pd.read_csv('/Users/harsh/Downloads/Global-Superstore.csv')
```

In [ ]:
```
# data head
df.head()
```

Out[ ]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment
0	32298	CA-2012-124891	7/31/2012	7/31/2012	Same Day	RH-19495	Rick Hansen	Consumer
1	26341	IN-2013-77878	2/5/2013	2/7/2013	Second Class	JR-16210	Justin Ritter	Corporate
2	25330	IN-2013-71249	10/17/2013	10/18/2013	First Class	CR-12730	Craig Reiter	Consumer
3	13524	ES-2013-1579342	1/28/2013	1/30/2013	First Class	KM-16375	Katherine Murray	Home Office
4	47221	SG-2013-4320	11/5/2013	11/6/2013	Same Day	RH-9495	Rick Hansen	Consumer

5 rows × 24 columns

In [ ]:

```
# data info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 24 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Row ID          51290 non-null  int64  
 1   Order ID        51290 non-null  object 
 2   Order Date      51290 non-null  object 
 3   Ship Date       51290 non-null  object 
 4   Ship Mode       51290 non-null  object 
 5   Customer ID     51290 non-null  object 
 6   Customer Name   51290 non-null  object 
 7   Segment         51290 non-null  object 
 8   City            51290 non-null  object 
 9   State           51290 non-null  object 
 10  Country         51290 non-null  object 
 11  Postal Code     9994 non-null   float64
 12  Market          51290 non-null  object 
 13  Region          51290 non-null  object 
 14  Product ID      51290 non-null  object 
 15  Category        51290 non-null  object 
 16  Sub-Category    51290 non-null  object 
 17  Product Name    51290 non-null  object 
 18  Sales           51290 non-null  float64
 19  Quantity        51290 non-null  int64  
 20  Discount        51290 non-null  float64
 21  Profit          51290 non-null  float64
 22  Shipping Cost   51290 non-null  float64
 23  Order Priority  51290 non-null  object 
dtypes: float64(5), int64(2), object(17)
memory usage: 9.4+ MB
```

In [ ]:

```
# data summary statistics
df.describe()
```

Out[ ]:

	Row ID	Postal Code	Sales	Quantity	Discount	
count	51290.00000	9994.000000	51290.000000	51290.000000	51290.000000	5
mean	25645.50000	55190.379428	246.490581	3.476545	0.142908	
std	14806.29199	32063.693350	487.565361	2.278766	0.212280	
min	1.00000	1040.000000	0.444000	1.000000	0.000000	-(
25%	12823.25000	23223.000000	30.758625	2.000000	0.000000	
50%	25645.50000	56430.500000	85.053000	3.000000	0.000000	
75%	38467.75000	90008.000000	251.053200	5.000000	0.200000	
max	51290.00000	99301.000000	22638.480000	14.000000	0.850000	:

In [ ]:

```
# dataframe shape
```

```
df.shape
```

Out[ ]: (51290, 24)

In [ ]:
```
# columns with missing data
df.isnull().sum()
```

Out[ ]:
```
Row ID                0
Order ID              0
Order Date            0
Ship Date             0
Ship Mode             0
Customer ID           0
Customer Name         0
Segment               0
City                  0
State                 0
Country               0
Postal Code       41296
Market                0
Region                0
Product ID            0
Category              0
Sub-Category          0
Product Name          0
Sales                 0
Quantity              0
Discount              0
Profit                0
Shipping Cost         0
Order Priority        0
dtype: int64
```

In [ ]:
```
# rows with duplicated data
df.duplicated().sum()
```

Out[ ]: 0

In [ ]:
```
df = df.copy()
# rename the column names to snake_case without spaces
df.columns = df.columns.str.replace(' ', '_').str.lower()
df.columns
```

Out[ ]:
```
Index(['row_id', 'order_id', 'order_date', 'ship_date', 'ship_mode',
       'customer_id', 'customer_name', 'segment', 'city', 'state', 'coun
try',
       'postal_code', 'market', 'region', 'product_id', 'category',
       'sub-category', 'product_name', 'sales', 'quantity', 'discount',
       'profit', 'shipping_cost', 'order_priority'],
      dtype='object')
```

In [ ]:
```
# strings to dates
df['order_date'] = pd.to_datetime(df['order_date'])
```

```
df['ship_date'] = pd.to_datetime(df['ship_date'])
```

In [ ]:
```
# confirm changes
df[['order_date', 'ship_date']].dtypes
```

Out[ ]:
```
order_date    datetime64[ns]
ship_date     datetime64[ns]
dtype: object
```

In [ ]:
```
# create a new column sales_year
df['sales_year'] = pd.DatetimeIndex(df['order_date']).year
```

In [ ]:
```
# convert categorical columns data type from object to category
cols = ['ship_mode', 'segment', 'state', 'country', 'region', 'market', '

df[cols] = df[cols].astype('category')
```

In [ ]:
```
# confirm changes
df.dtypes
```

Out[ ]:
```
row_id                     int64
order_id                  object
order_date        datetime64[ns]
ship_date         datetime64[ns]
ship_mode               category
customer_id               object
customer_name             object
segment                 category
city                      object
state                   category
country                 category
postal_code              float64
market                  category
region                  category
product_id                object
category                category
sub-category            category
product_name              object
sales                    float64
quantity                   int64
discount                 float64
profit                   float64
shipping_cost            float64
order_priority          category
sales_year                 int64
dtype: object
```

2. visualisation

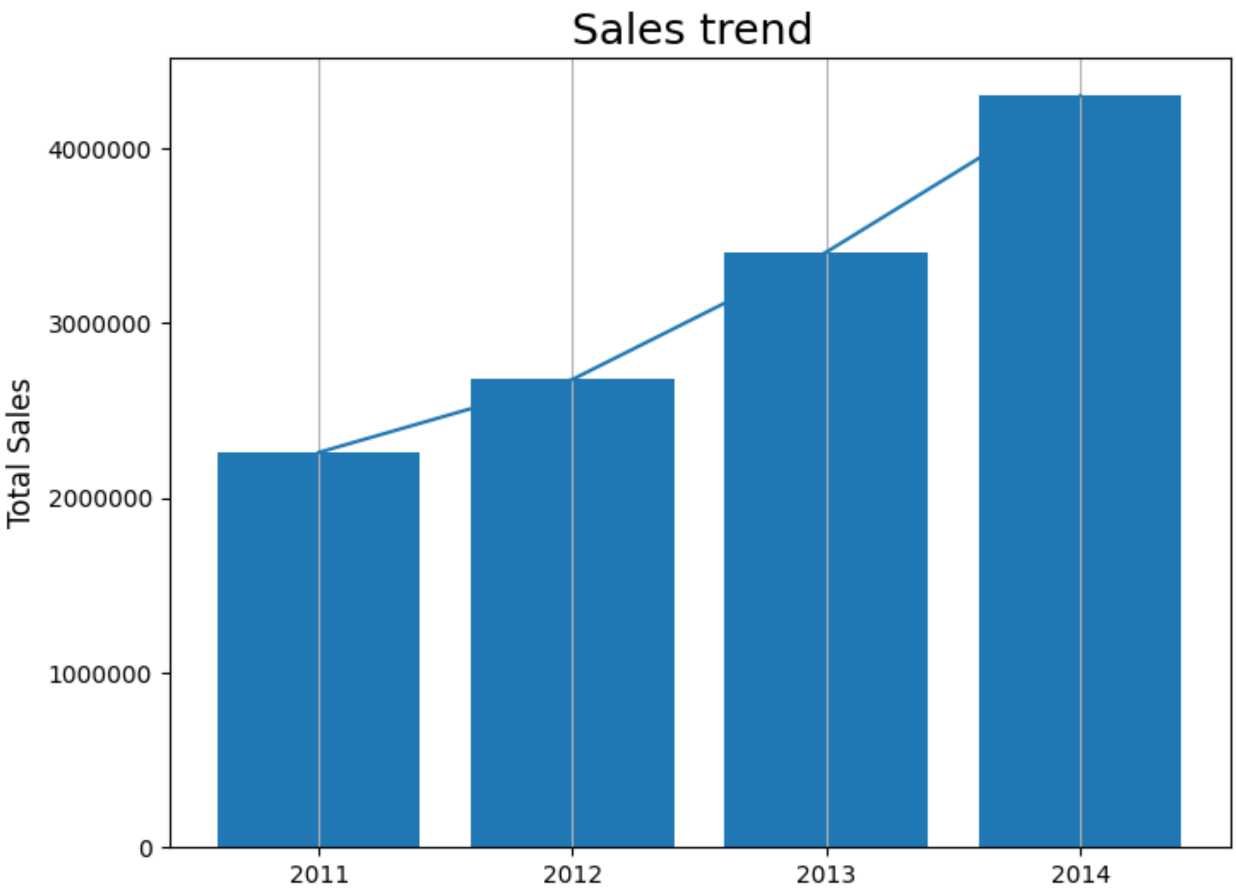
2.1 trend analysis

In [ ]:
```
# sales trend
sales_trend = df.groupby('sales_year', as_index=False)['sales'].sum()
sales_trend
```

Out[ ]:

	sales_year	sales
0	2011	2.259451e+06
1	2012	2.677439e+06
2	2013	3.405746e+06
3	2014	4.299866e+06

In [ ]:
```
# plot total profit trends
fig, ax = plt.subplots(figsize=(8,6))


ax.bar(x=sales_trend.sales_year, height=sales_trend.sales)
ax.plot(sales_trend.sales_year,sales_trend.sales)
ax.yaxis.get_major_formatter().set_scientific(False)
plt.title('Sales trend ', fontsize=18)
plt.ylabel('Total Sales', fontsize=12)
plt.xticks([2011,2012,2013,2014])
plt.grid(axis='x')
```



In [ ]:
```
# profit trend
```

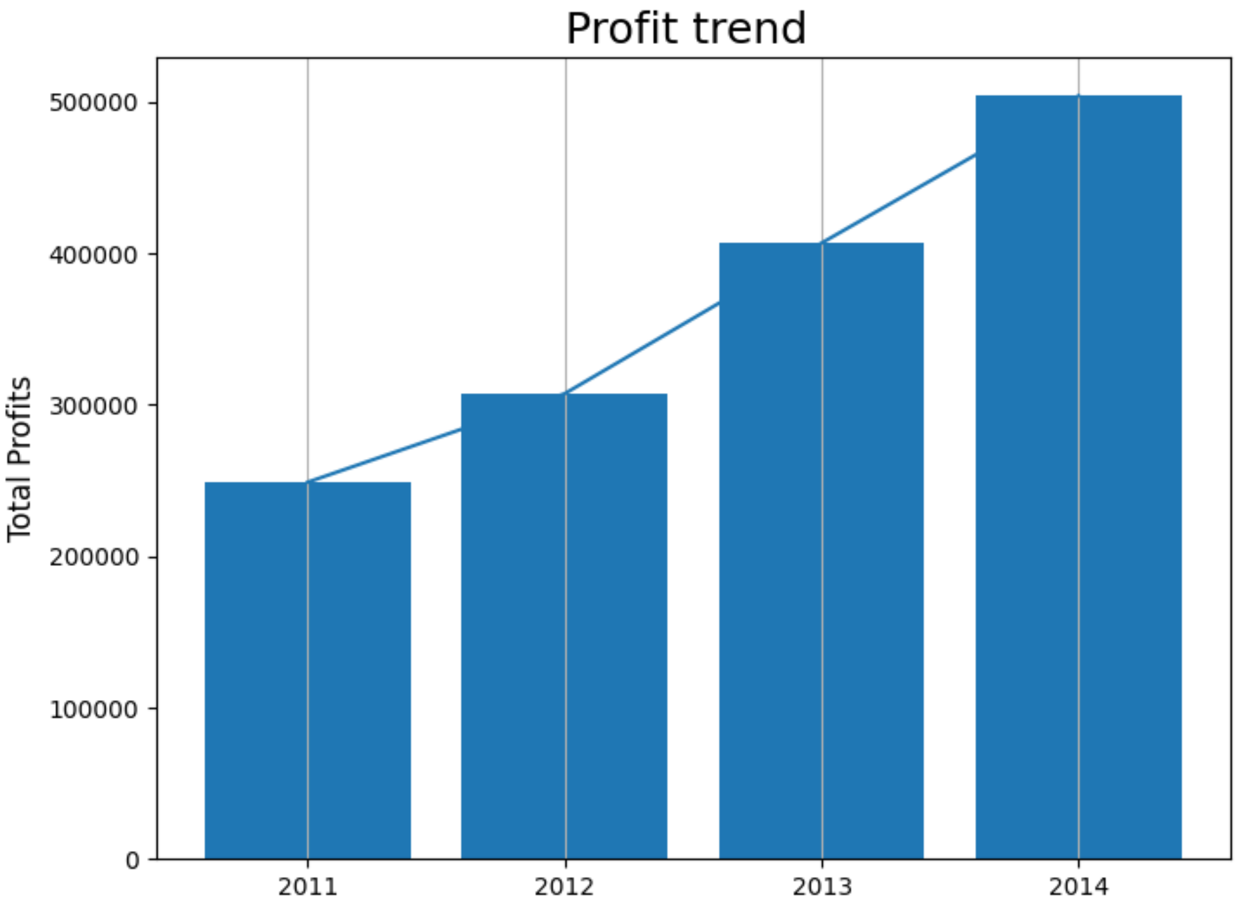
```
profit_trend = df.groupby('sales_year', as_index=False)['profit'].sum()
profit_trend['sales_year'] = profit_trend['sales_year'].astype('category'
profit_trend
```

Out[ ]:

	sales_year	profit
0	2011	248940.81154
1	2012	307415.27910
2	2013	406935.23018
3	2014	504165.97046

In [ ]:

```
# plot total profit trends
fig, ax = plt.subplots(figsize=(8,6))


plt.bar(x=profit_trend.sales_year, height=profit_trend.profit)
plt.plot(profit_trend.sales_year,profit_trend.profit)
plt.title('Profit trend ', fontsize=18)
plt.ylabel('Total Profits', fontsize=12)
plt.xticks([2011,2012,2013,2014])
plt.grid(axis='x')
```



2.2 geographic data pattern

In [ ]:
```
# group data by market
sales_by_market = df.groupby('market', as_index=False).sum().sort_values(

# calculate profit margins
sales_by_market['profit_margin'] = sales_by_market['profit'] / sales_by_m

sales_by_market
```

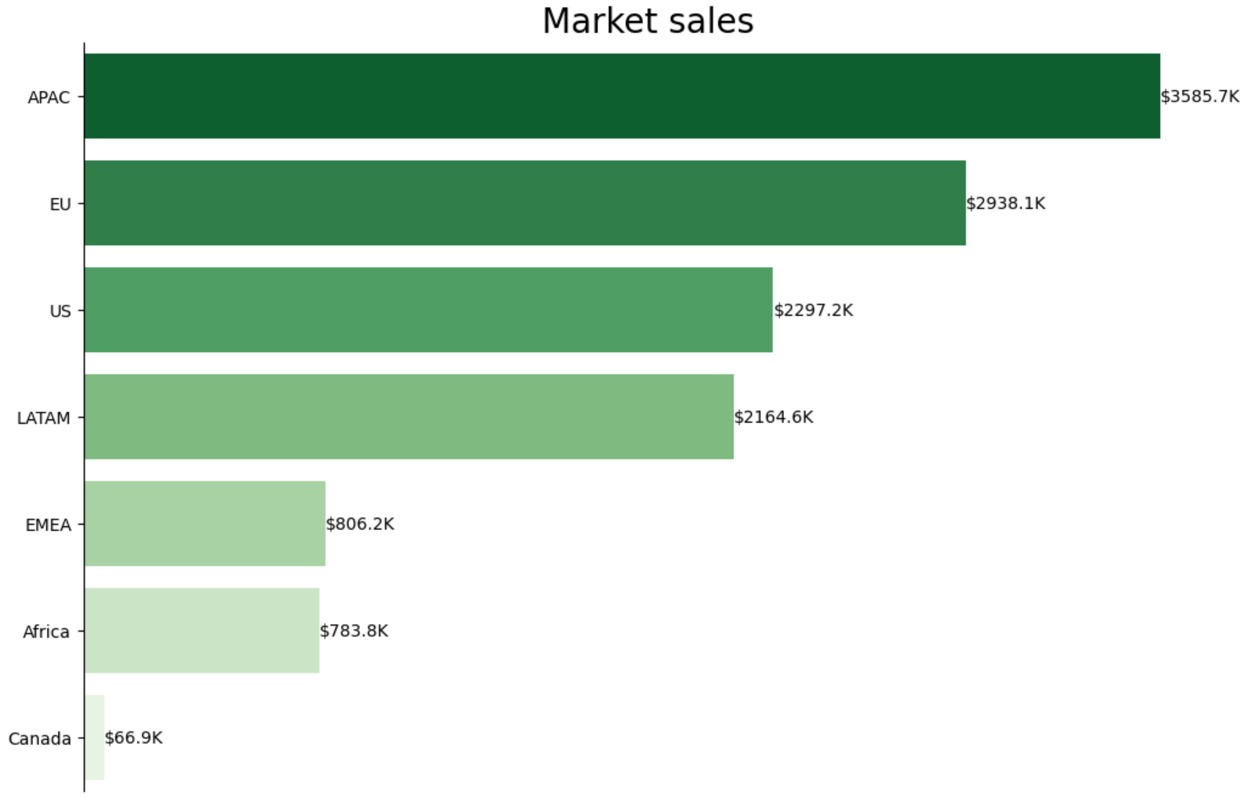
```
/var/folders/4m/c9tdbxjd0xx04psbvrw3q3ym0000gn/T/ipykernel_19593/20692967
8.py:2: FutureWarning: The default value of numeric_only in DataFrameGroup
By.sum is deprecated. In a future version, numeric_only will default to Fa
lse. Either specify numeric_only or select only columns which should be va
lid for the function.
  sales_by_market = df.groupby('market', as_index=False).sum().sort_value
s(by='sales', ascending=False)
```

Out[ ]:

	market	row_id	postal_code	sales	quantity	discount	pr
0	APAC	283802091	0.0	3.585744e+06	41226	1637.530	436000.04
4	EU	152945000	0.0	2.938089e+06	37773	1031.050	372829.74
6	US	362717239	551572652.0	2.297201e+06	37873	1561.090	286397.02
5	LATAM	52988365	0.0	2.164605e+06	38526	1395.158	221643.48
3	EMEA	233028207	0.0	8.061613e+05	11517	986.100	43897.97
1	Africa	212025742	0.0	7.837732e+05	10564	718.800	88871.63
2	Canada	17851051	0.0	6.692817e+04	833	0.000	17817.39

In [ ]:
```
# plot the graph
fig, ax = plt.subplots(figsize=(12,8))

# plot sales per market
sns.barplot(x=sales_by_market['sales'], y=sales_by_market['market'], pale
            order=sales_by_market.sort_values(by='sales', ascending=Fals
plt.title('Market sales', fontdict={'fontsize':20})

#Annotating the sales.
for p in ax.patches:
    _, y = p.get_xy()

    ax.annotate(f"${p.get_width() / 1000 :.1f}K", xy=(p.get_width(), y+0.

#Removing bar junk
ax.spines[["top", "right", "bottom"]].set_visible(False)
ax.set(ylabel=None, xlabel=None)
ax.tick_params(labelbottom=None, bottom=None)
```

```
/var/folders/4m/c9tdbxjd0xx04psbvrw3q3ym0000gn/T/ipykernel_19593/426254049
8.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for
the same effect.

  sns.barplot(x=sales_by_market['sales'], y=sales_by_market['market'], pal
ette='Greens_r',
```



In [ ]:

```
# profits in different Markets
df.groupby('market')['profit'].sum().sort_values(ascending=False)
```

Out[ ]:

```
market
APAC      436000.04900
EU        372829.74150
US        286397.02170
LATAM     221643.48708
Africa     88871.63100
EMEA       43897.97100
Canada     17817.39000
Name: profit, dtype: float64
```

In [ ]:

```
# profit in different countries
profit_by_market = df.groupby('market', as_index=False).sum().sort_values
```

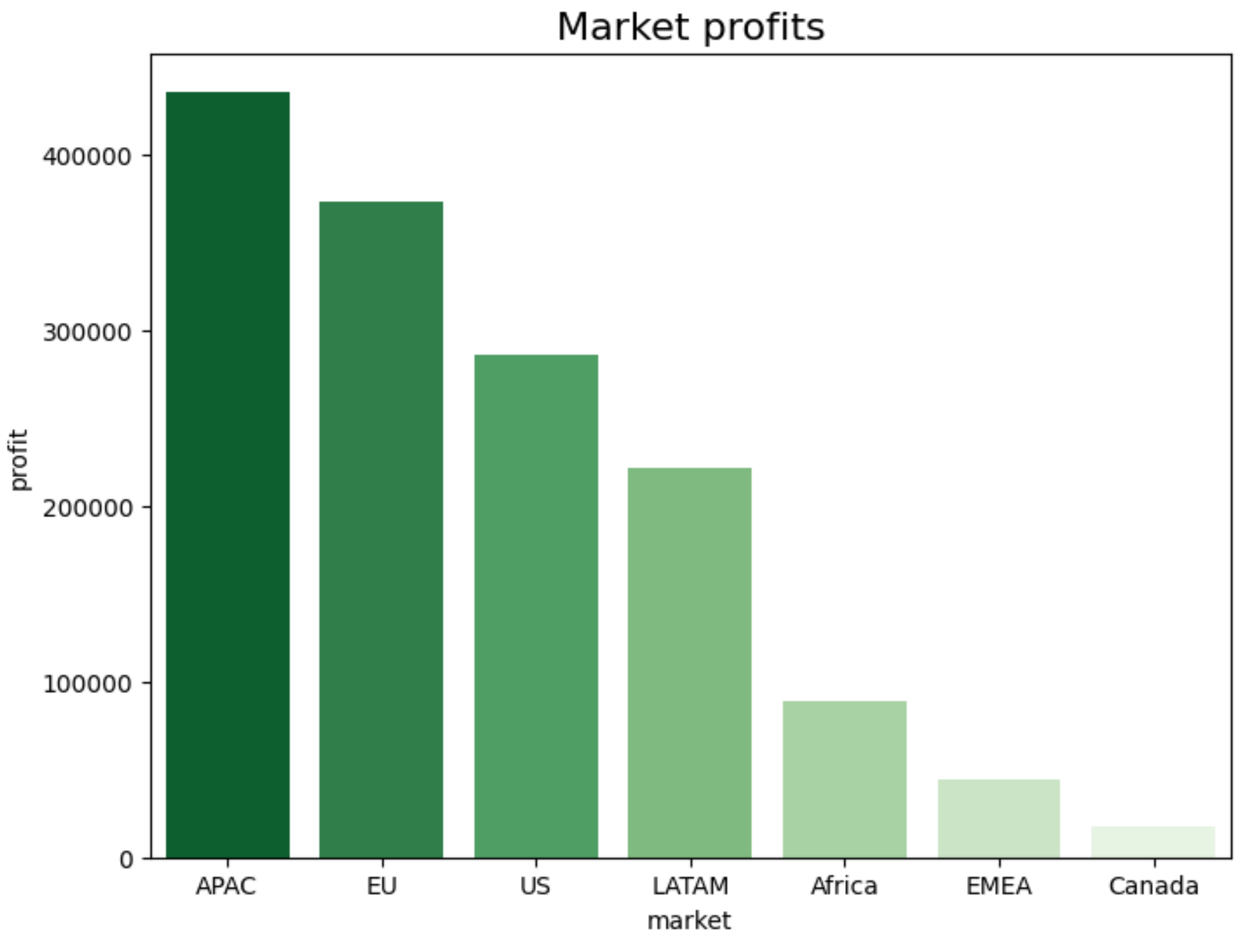
```
/var/folders/4m/c9tdbxjd0xx04psbvrw3q3ym0000gn/T/ipykernel_19593/273868504
2.py:2: FutureWarning: The default value of numeric_only in DataFrameGroup
By.sum is deprecated. In a future version, numeric_only will default to Fa
lse. Either specify numeric_only or select only columns which should be va
lid for the function.
  profit_by_market = df.groupby('market', as_index=False).sum().sort_value
s(by='profit', ascending=False)
```

In [ ]:

```
fig, ax =  plt.subplots(figsize=(8,6))

sns.barplot(x=profit_by_market['market'], y=profit_by_market['profit'], p
            order=profit_by_market.sort_values('profit', ascending=False)
plt.title('Market profits', fontdict={'fontsize':16})
```

```
/var/folders/4m/c9tdbxjd0xx04psbvrw3q3ym0000gn/T/ipykernel_19593/113218887
7.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remove
d in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for
the same effect.

  sns.barplot(x=profit_by_market['market'], y=profit_by_market['profit'],
palette='Greens_r',
```

Out[ ]: Text(0.5, 1.0, 'Market profits')



In [ ]:

```
# average shipping cost in different markets from lowest
```

```
df.groupby('market')['shipping_cost'].mean().sort_values()
```

Out[ ]:
```
market
EMEA      17.573221
Africa    19.215058
Canada    19.285495
LATAM     22.745153
US        23.831678
EU        30.942235
APAC      35.190430
Name: shipping_cost, dtype: float64
```

In [ ]:
```
# How long does it take to ship products
# add a new column shipment_days
df['shipment_days'] = df['ship_date'] - df['order_date']
```

In [ ]:
```
# shipment days on average in different markets
df.groupby('market')['shipment_days'].mean().sort_values()
```

Out[ ]:
```
market
Canada                 3 days 16:15:00
Africa   3 days 21:50:58.469587966
EMEA     3 days 22:24:04.581427719
US       3 days 23:00:46.828096858
APAC     3 days 23:15:29.940010907
LATAM    3 days 23:55:23.023120264
EU          4 days 00:11:57.120000
Name: shipment_days, dtype: timedelta64[ns]
```

2.3 category wise analysis

In [ ]:
```
# group total sales by category from the highest sale.
sales_category = df.groupby('category')['sales'].sum().sort_values(ascend
sales_category
```

Out[ ]:
```
category
Technology         4.744557e+06
Furniture          4.110874e+06
Office Supplies    3.787070e+06
Name: sales, dtype: float64
```

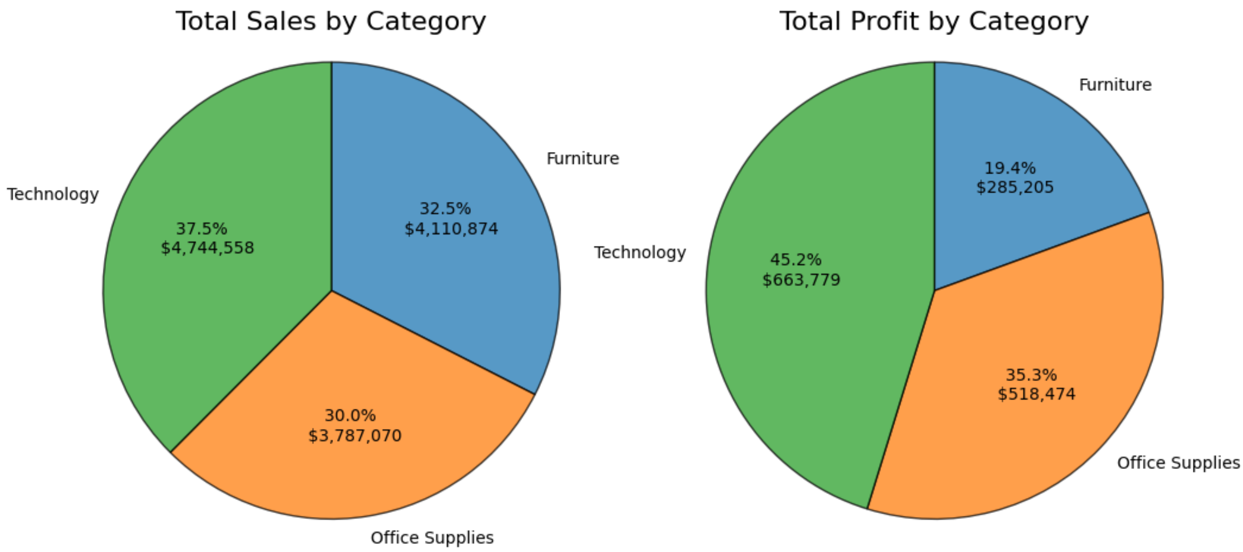
In [ ]:
```
# group total profits by category
profit_category = df.groupby('category')['profit'].sum().sort_values(asce
profit_category
```

Out[ ]:
```
category
Technology         663778.73318
Office Supplies    518473.83430
Furniture          285204.72380
Name: profit, dtype: float64
```

In [ ]:
```
# group total sales by category
```

```
sales_category = df.groupby('category')['sales'].sum()

# group total profits by category
profit_category = df.groupby('category')['profit'].sum()


# figure size
plt.figure(figsize=(12,8))

# left total sales pie chart
plt.subplot(1,2,1)  # 1 row, 2 columns, the 1st plot.
plt.pie(sales_category.values, labels=sales_category.index, startangle=90
        autopct=lambda p:f'{p:.1f}% \n ${p*np.sum(sales_category.values)/
        wedgeprops={'linewidth': 1, 'edgecolor':'black', 'alpha':0.75})
plt.axis('square')
plt.title('Total Sales by Category',  fontdict={'fontsize':16})

# right total profits pie chart
plt.subplot(1,2,2)  # 1 row, 2 columns, the 2nd plot
plt.pie(profit_category.values, labels=profit_category.index, startangle=
        autopct=lambda p:f'{p:.1f}% \n ${p*np.sum(profit_category.values)
        wedgeprops={'linewidth': 1, 'edgecolor':'black', 'alpha':0.75})
plt.axis('square')
plt.title('Total Profit by Category', fontdict={'fontsize':16})
```

Out[ ]: Text(0.5, 1.0, 'Total Profit by Category')



Interestingly, Furniture has a lower percentage of profits raked in compared to its share of percentage sale.

3 Findings

Categories and Subcategories

- The three main categories (Technology, Furniture, and Office Supplies) each

contribute significantly to sales, with Technology leading at 37.5%.
- Technology generates the highest total profits (45.2%), followed by Office Supplies (35.3%) and Furniture (19.4%).
- Furniture, despite high sales, has relatively low profits.

Geographical Markets

- APAC is the largest market for Global Superstore, followed by the US, EMEA, and Canada.
- Canada, despite low sales, has the highest profit margin.
- Profit margins generally range from 10% to 14% across markets.
- The EMEA market has lower profits and a lower profit margin compared to other markets.

Time Series

- Global Superstore has experienced steady growth in sales and profits from 2011 to 2014.
- This positive trend indicates potential for future growth.