In [1]:
```
import pandas as pd
pd.__version__
```

Out[1]: '1.1.3'

Data Cleaning

Data cleaning is a crucial step in the data analysis process, especially when dealing with real-world data, which often comes with inconsistencies, errors, and missing values. Given the superstore.csv dataset you're working with, here are generalized steps for data cleaning. These steps can be adjusted based on specific requirements and findings as you explore the data.

1. Load the Dataset Before any cleaning, you need to load the dataset into a pandas DataFrame to manipulate and analyze the data easily.

import pandas as pd

file_path = '---/superstore.csv' store = pd.read_csv(r 'file_path')

1. Initial Exploration Get a sense of the data's structure, types of data, and any obvious issues:

View the first few rows using store.head(). Get a summary of the DataFrame using store.info(), which shows column names, non-null counts, and data types. Examine basic statistics for numerical columns with store.describe().

1. Identify Missing Values Use store.isnull().sum() to identify columns with missing values. This helps you understand the extent of missing data.

2. Handle Missing Values Decide on a strategy for dealing with missing data, which might include:

Dropping: If a column has a significant number of missing values or if certain rows are missing data that is crucial for your analysis, you might decide to drop those columns or rows using store.dropna(). Imputing: For columns with numerical data, consider imputing missing values with a central tendency measure (mean, median, mode). For categorical data, you might impute with the most frequent category or use another imputation method. Pandas provides the fillna method for this purpose.

1. Correct Data Types Ensure each column is of the correct data type for your analysis needs:

Convert data types as necessary using astype() method. For example, converting a 'date' column from object to datetime.

1. Detect and Handle Outliers Outliers can skew your analysis:

Use statistical methods or visualizations (e.g., box plots) to detect outliers. Decide whether to keep, remove, or adjust these outliers.

1. Validate Data Consistency Check for and resolve inconsistencies in categorical data, such as different spellings of the same category: Use unique() to inspect unique values. Standardize

and clean strings as necessary, using methods like str.lower(), str.replace(), or manual mapping.

2. De-duplicate Data Remove any duplicate rows to prevent skewed analysis: Use store.duplicated().sum() to check for duplicates. Remove duplicates with store.drop_duplicates(inplace=True).

3. Feature Engineering (Optional) Based on your analysis goals, you might create new columns that are derived from existing data or combine data in meaningful ways to aid your analysis.

4. Final Inspection Perform a final review of the DataFrame to ensure all issues have been addressed and the data is clean: Re-examine the summary statistics and a sample of the data. Verify that all columns have the appropriate data type and no missing values remain.

5. Save Cleaned Dataset (Optional) Once the data is cleaned, consider saving the cleaned DataFrame to a new file for easy access in future analyses.

store.to_csv('---/superstore_cleaned.csv', index=False) These steps provide a comprehensive approach to cleaning the superstore.csv dataset. The exact process might vary based on the specific analyses you plan to perform and the issues you encounter in the data.

In [2]:
```
store = pd.read_csv(r'C:\DataScience\Panda\superstore.csv') # load csv file
```

In [3]:
```
store
```

Out[3]:

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code
0	Office Supplies	Houston	United States	Darren Powers	Message Book	03-01-2020	US-2020-103800	77095
1	Office Supplies	Naperville	United States	Phillina Ober	GBC	04-01-2020	US-2020-112326	60540
2	Office Supplies	Naperville	United States	Phillina Ober	Avery	04-01-2020	US-2020-112326	60540
3	Office Supplies	Naperville	United States	Phillina Ober	SAFCO	04-01-2020	US-2020-112326	60540
4	Office Supplies	Philadelphia	United States	Mick Brown	Avery	05-01-2020	US-2020-141817	19143
...	...	...	...	...	...	...	...	...

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code
10189	Office Supplies	New York City	United States	Patrick O'Donnell	Wilson Jones	30-12-2023	US-2023-143259	10009
10190	Office Supplies	Fairfield	United States	Erica Bern	GBC	30-12-2023	US-2023-115427	94533
10191	Office Supplies	Loveland	United States	Jill Matthias	Other	30-12-2023	US-2023-156720	80538
10192	Technology	New York City	United States	Patrick O'Donnell	Other	30-12-2023	US-2023-143259	10009
10193	Office Supplies	Charlottetown	Canada	Harry Olson	Wilson Jones	30-12-2023	CA-2023-143500	C0A

10194 rows × 19 columns

In [4]:
```
id(store) #address location
```

Out[4]:
```
2507902895296
```

In [5]:
```
len(store) #will return the total number of rows in the store DataFrame. In pandas,
```

Out[5]:
```
10194
```

In [6]:
```
store.columns #check the Store columns
```

Out[6]:
```
Index(['Category', 'City', 'Country/Region', 'Customer Name', 'Manufacturer',
       'Order Date', 'Order ID', 'Postal Code', 'Product Name', 'Region',
       'Segment', 'Ship Date', 'Ship Mode', 'State/Province', 'Sub-Category',
       'Discount', 'Profit', 'Quantity', 'Sales'],
      dtype='object')
```

In [7]:
```
len(store.columns) #number of the columns
```

Out[7]:
```
19
```

In [8]:
```
store.shape #dimension of the dataset (rows and columns)
```

Out[8]:
```
(10194, 19)
```

In [9]:
```
store.head
```

Out[9]:
```
<bound method NDFrame.head of              Category           City Country/Region
Customer Name  \
0      Office Supplies        Houston  United States      Darren Powers
1      Office Supplies     Naperville  United States      Phillina Ober
2      Office Supplies     Naperville  United States      Phillina Ober
3      Office Supplies     Naperville  United States      Phillina Ober
4      Office Supplies   Philadelphia  United States         Mick Brown
...                ...            ...            ...                ...
10189  Office Supplies  New York City  United States  Patrick O'Donnell
```

```
10190  Office Supplies      Fairfield  United States         Erica Bern
10191  Office Supplies       Loveland  United States      Jill Matthias
10192       Technology  New York City  United States  Patrick O'Donnell
10193  Office Supplies  Charlottetown         Canada        Harry Olson

       Manufacturer  Order Date        Order ID Postal Code  \
0      Message Book  03-01-2020  US-2020-103800       77095
1               GBC  04-01-2020  US-2020-112326       60540
2             Avery  04-01-2020  US-2020-112326       60540
3             SAFCO  04-01-2020  US-2020-112326       60540
4             Avery  05-01-2020  US-2020-141817       19143
...             ...         ...             ...         ...
10189  Wilson Jones  30-12-2023  US-2023-143259       10009
10190           GBC  30-12-2023  US-2023-115427       94533
10191         Other  30-12-2023  US-2023-156720       80538
10192         Other  30-12-2023  US-2023-143259       10009
10193  Wilson Jones  30-12-2023  CA-2023-143500         C0A

                                        Product Name   Region  \
0      Message Book, Wirebound, Four 5 1/2" X 4" Form...  Central
1             GBC Standard Plastic Binding Systems Combs  Central
2                                           Avery 508  Central
3                       SAFCO Boltless Steel Shelving  Central
4      Avery Hi-Liter EverBold Pen Style Fluorescent ...     East
...                                                  ...      ...
10189              Wilson Jones Legal Size Ring Binders     East
10190                                GBC Binding covers     West
10191                               Bagged Rubber Bands     West
10192                          Gear Head AU3700S Headset     East
10193                        Wilson Jones Impact Binders     East

           Segment   Ship Date       Ship Mode        State/Province  \
0         Consumer  07-01-2020  Standard Class                 Texas
1      Home Office  08-01-2020  Standard Class              Illinois
2      Home Office  08-01-2020  Standard Class              Illinois
3      Home Office  08-01-2020  Standard Class              Illinois
4         Consumer  12-01-2020  Standard Class          Pennsylvania
...            ...         ...             ...                   ...
10189     Consumer  03-01-2024  Standard Class              New York
10190    Corporate  03-01-2024  Standard Class            California
10191     Consumer  03-01-2024  Standard Class              Colorado
10192     Consumer  03-01-2024  Standard Class              New York
10193     Consumer  03-01-2024  Standard Class  Prince Edward Island

      Sub-Category  Discount   Profit  Quantity    Sales
0            Paper       0.2   5.5512         2   16.448
1          Binders       0.8  -5.4870         2    3.540
2           Labels       0.2   4.2717         3   11.784
3          Storage       0.2 -64.7748         3  272.736
4              Art       0.2   4.8840         3   19.536
...            ...       ...      ...       ...      ...
10189      Binders       0.2  19.7910         3   52.776
10190      Binders       0.2   6.4750         2   20.720
10191    Fasteners       0.2  -0.6048         3    3.024
10192       Phones       0.0   2.7279         7   90.930
10193      Binders       0.2  -0.6048         3    3.024

[10194 rows x 19 columns]>
```

I have encountered an unexpected result when attempting to display the first few rows of your DataFrame using store.head(). Instead of showing the rows, I got a representation of the bound method NDFrame.head of my DataFrame object. This typically happens when one accidentally treat store.head as a property or attribute rather than calling it as a method. To resolve this and correctly display the first few rows, ensure you include parentheses () after head, even if you are not passing any arguments to specify the number of rows you want to see.

In [10]:
```
print(store.head())
```

```
          Category          City Country/Region  Customer Name  Manufacturer  \
0  Office Supplies       Houston  United States  Darren Powers  Message Book
1  Office Supplies    Naperville  United States  Phillina Ober           GBC
2  Office Supplies    Naperville  United States  Phillina Ober         Avery
3  Office Supplies    Naperville  United States  Phillina Ober         SAFCO
4  Office Supplies  Philadelphia  United States     Mick Brown         Avery

   Order Date        Order ID Postal Code  \
0  03-01-2020  US-2020-103800       77095
1  04-01-2020  US-2020-112326       60540
2  04-01-2020  US-2020-112326       60540
3  04-01-2020  US-2020-112326       60540
4  05-01-2020  US-2020-141817       19143

                                      Product Name   Region      Segment  \
0  Message Book, Wirebound, Four 5 1/2" X 4" Form...  Central     Consumer
1         GBC Standard Plastic Binding Systems Combs  Central  Home Office
2                                       Avery 508  Central  Home Office
3                   SAFCO Boltless Steel Shelving  Central  Home Office
4  Avery Hi-Liter EverBold Pen Style Fluorescent ...     East     Consumer

    Ship Date       Ship Mode State/Province Sub-Category  Discount   Profit  \
0  07-01-2020  Standard Class          Texas        Paper       0.2   5.5512
1  08-01-2020  Standard Class       Illinois      Binders       0.8  -5.4870
2  08-01-2020  Standard Class       Illinois       Labels       0.2   4.2717
3  08-01-2020  Standard Class       Illinois      Storage       0.2 -64.7748
4  12-01-2020  Standard Class   Pennsylvania          Art       0.2   4.8840

   Quantity    Sales
0         2   16.448
1         2    3.540
2         3   11.784
3         3  272.736
4         3   19.536
```

This output gives one an initial glimpse into the dataset, showing various details about orders from a superstore, including categories of items sold, cities and countries where sales occurred, customer names, order dates, and financials like sales and profit.

From this snippet, you can observe the following columns in your dataset:

Category: The category of the product sold (e.g., Office Supplies). City: The city where the product was sold. Country/Region: The country or region of the sale. Customer Name: The name of the customer who made the purchase. Manufacturer: The brand or manufacturer of the product. Order Date: The date when the order was placed. Order ID: A unique identifier for the order. Postal Code: The postal code of the order's destination. Product Name: The name of the product sold. Region: The broader region where the city is located. Segment: The segment of the customer (e.g., Consumer, Home Office). Ship Date: The date when the product was shipped. Ship Mode: The shipping mode (e.g., Standard Class). State/Province: The state or province of the order's destination. Sub-Category: More detailed categorization of the product. Discount: The discount applied to the order. Profit: The profit made from the order. Quantity: The quantity of the product sold. Sales: The total sales amount of the order. With this data, you can perform various analyses, such as:

Sales and profit analysis per category, sub-category, or region. Customer behavior analysis based on segments and regions. Trends over time, considering order and ship dates. Effect of discounts on sales and profit. This dataset appears rich in information, allowing for multiple angles of

analysis to gain insights into business performance, customer preferences, and operational efficiency.

In [11]:
```
store.dtypes #return a Series with the data types of each column in the DataFrame
```

Out[11]:
```
Category           object
City               object
Country/Region     object
Customer Name      object
Manufacturer       object
Order Date         object
Order ID           object
Postal Code        object
Product Name       object
Region             object
Segment            object
Ship Date          object
Ship Mode          object
State/Province     object
Sub-Category       object
Discount          float64
Profit            float64
Quantity            int64
Sales             float64
dtype: object
```

In [12]:
```
store.info() #Print a concise summary of a dataframe.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10194 entries, 0 to 10193
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Category        10194 non-null  object 
 1   City            10194 non-null  object 
 2   Country/Region  10194 non-null  object 
 3   Customer Name   10194 non-null  object 
 4   Manufacturer    10194 non-null  object 
 5   Order Date      10194 non-null  object 
 6   Order ID        10194 non-null  object 
 7   Postal Code     10194 non-null  object 
 8   Product Name    10194 non-null  object 
 9   Region          10194 non-null  object 
 10  Segment         10194 non-null  object 
 11  Ship Date       10194 non-null  object 
 12  Ship Mode       10194 non-null  object 
 13  State/Province  10194 non-null  object 
 14  Sub-Category    10194 non-null  object 
 15  Discount        10194 non-null  float64
 16  Profit          10194 non-null  float64
 17  Quantity        10194 non-null  int64  
 18  Sales           10194 non-null  float64
dtypes: float64(3), int64(1), object(15)
memory usage: 1.5+ MB
```

The store.info() method in pandas is used to print a concise summary of a DataFrame. When you call this method on your DataFrame, named store in this case, it provides important information about the DataFrame, including:

The number of entries (rows) and columns in the DataFrame. The column names and their data types. The number of non-null values in each column, which can be useful for identifying columns with missing values. The amount of memory usage by the DataFrame, which can be important for managing resources, especially with large datasets. Getting a quick overview of

your data, understanding the structure of your DataFrame, and planning data preprocessing steps.

In [13]:
```
type(store)
```

Out[13]:
```
pandas.core.frame.DataFrame
```

The expression type(store) in Python is used to determine the type of the object named store. If store is a DataFrame object created using pandas, then type(store) will return <class 'pandas.core.frame.DataFrame'>. This indicates that store is indeed a pandas DataFrame, which is a 2-dimensional labeled data structure with columns that can be of different types.

In [14]:
```
store.describe()
```

Out[14]:

	Discount	Profit	Quantity	Sales
count	10194.000000	10194.000000	10194.000000	10194.000000
mean	0.155385	28.673417	3.791838	228.225854
std	0.206249	232.465115	2.228317	619.906839
min	0.000000	-6599.978000	1.000000	0.444000
25%	0.000000	1.760800	2.000000	17.220000
50%	0.200000	8.690000	3.000000	53.910000
75%	0.200000	29.297925	5.000000	209.500000
max	0.800000	8399.976000	14.000000	22638.480000

The store.describe() method in pandas generates descriptive statistics that summarize the central tendency, dispersion, and shape of a dataset's distribution, excluding NaN values. It's primarily used for numerical columns but can also be applied to categorical data by specifying the include parameter. By default, describe() includes only numerical columns in its output.

For each numerical column, describe() typically provides the following statistics:

count: The number of non-null (i.e., not missing) entries. mean: The mean of the values in the column. std: The standard deviation of the values, a measure of the dispersion or variability. min: The minimum value in the column. 25% (first quartile): The value below which 25% of the data fall. 50% (median): The middle value, dividing the data into two halves. 75% (third quartile): The value below which 75% of the data fall. max: The maximum value in the column. If your store DataFrame contains both numerical and categorical data and you want to include statistics for the categorical data as well, you can call describe(include='all'), which will include additional statistics for non-numeric columns:

unique: The number of unique values. top: The most common value. freq: The most common value frequency

In [15]:
```
print(store.describe()) # For numerical columns only
```

```
           Discount        Profit      Quantity         Sales
count  10194.000000  10194.000000  10194.000000  10194.000000
mean       0.155385     28.673417      3.791838    228.225854
std        0.206249    232.465115      2.228317    619.906839
min        0.000000  -6599.978000      1.000000      0.444000
25%        0.000000      1.760800      2.000000     17.220000
```

```
50%        0.200000      8.690000      3.000000     53.910000
75%        0.200000     29.297925      5.000000    209.500000
max        0.800000   8399.976000     14.000000  22638.480000
```

In [16]:
```
print(store.describe(include='all')) # For both numerical and categorical columns
```

```
                Category           City Country/Region  Customer Name  \
count              10194          10194          10194          10194
unique                 3            542              2            800
top     Office Supplies  New York City  United States  William Brown
freq                6128            915           9994             41
mean                 NaN            NaN            NaN            NaN
std                  NaN            NaN            NaN            NaN
min                  NaN            NaN            NaN            NaN
25%                  NaN            NaN            NaN            NaN
50%                  NaN            NaN            NaN            NaN
75%                  NaN            NaN            NaN            NaN
max                  NaN            NaN            NaN            NaN

       Manufacturer  Order Date        Order ID Postal Code Product Name  \
count         10194       10194           10194       10194        10194
unique          183        1242            5111         654         1849
top           Other  05-09-2022  US-2023-100111       10035      Staples
freq           1940          38              14         263           50
mean            NaN         NaN             NaN         NaN          NaN
std             NaN         NaN             NaN         NaN          NaN
min             NaN         NaN             NaN         NaN          NaN
25%             NaN         NaN             NaN         NaN          NaN
50%             NaN         NaN             NaN         NaN          NaN
75%             NaN         NaN             NaN         NaN          NaN
max             NaN         NaN             NaN         NaN          NaN

       Region   Segment   Ship Date       Ship Mode State/Province  \
count   10194     10194       10194           10194          10194
unique      4         3        1338               4             59
top      West  Consumer  16-12-2021  Standard Class     California
freq     3253      5281          38            6120           2001
mean      NaN       NaN         NaN             NaN            NaN
std       NaN       NaN         NaN             NaN            NaN
min       NaN       NaN         NaN             NaN            NaN
25%       NaN       NaN         NaN             NaN            NaN
50%       NaN       NaN         NaN             NaN            NaN
75%       NaN       NaN         NaN             NaN            NaN
max       NaN       NaN         NaN             NaN            NaN

       Sub-Category      Discount        Profit      Quantity         Sales
count         10194  10194.000000  10194.000000  10194.000000  10194.000000
unique           17           NaN           NaN           NaN           NaN
top         Binders           NaN           NaN           NaN           NaN
freq           1548           NaN           NaN           NaN           NaN
mean            NaN      0.155385     28.673417      3.791838    228.225854
std             NaN      0.206249    232.465115      2.228317    619.906839
min             NaN      0.000000  -6599.978000      1.000000      0.444000
25%             NaN      0.000000      1.760800      2.000000     17.220000
50%             NaN      0.200000      8.690000      3.000000     53.910000
75%             NaN      0.200000     29.297925      5.000000    209.500000
max             NaN      0.800000   8399.976000     14.000000  22638.480000
```

In [17]:
```
store.isnull()#check missing values
```

Out[17]:

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code	Product Name	R
0	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code	Product Name	R
3	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	
10189	False	False	False	False	False	False	False	False	False	
10190	False	False	False	False	False	False	False	False	False	
10191	False	False	False	False	False	False	False	False	False	
10192	False	False	False	False	False	False	False	False	False	
10193	False	False	False	False	False	False	False	False	False	

10194 rows × 19 columns

The store.isnull() method in pandas is used to check for missing values in the DataFrame. When you call store.isnull() on your DataFrame, it returns a new DataFrame of the same shape as store, where each element is a boolean indicating whether the corresponding element in store is a NaN (Not a Number) or not.

True: Indicates that the original value in the store DataFrame is missing or is a NaN. False: Indicates that the original value is present and not a NaN. Why Use store.isnull()? Checking for missing values is an essential step in data preprocessing because missing data can significantly impact data analysis and modeling. Identifying where these missing values occur helps in deciding how to handle them, for example, by imputing missing values with some strategy (mean, median, mode, etc.) or by dropping rows/columns with missing values.

In [18]:
```
store.isnull().sum() #Return numer of null values
```

Out[18]:
```
Category          0
City              0
Country/Region    0
Customer Name     0
Manufacturer      0
Order Date        0
Order ID          0
Postal Code       0
Product Name      0
Region            0
Segment           0
Ship Date         0
Ship Mode         0
State/Province    0
Sub-Category      0
Discount          0
Profit            0
Quantity          0
Sales             0
dtype: int64
```

The store.isnull().sum() method chain in pandas is a two-step process that helps identify the number of missing (null) values in each column of the DataFrame named store. Here's how it works:

store.isnull(): This part returns a new DataFrame of the same shape as store, where each element is a boolean indicating whether the corresponding element in store is null (True) or not null (False).

.sum(): This part sums up the boolean values column-wise. In pandas, True is treated as 1 and False is treated as 0. Therefore, the sum operation counts the number of True values in each column, effectively giving you the number of missing values in each column.

Why is it useful? Identifying Missing Data: It's crucial to know where you have missing data in your dataset, as it can significantly impact the results of your data analysis or the performance of machine learning models. Data Cleaning and Preprocessing: Once you know which columns have missing values and how many there are, you can make informed decisions about how to handle these missing values (e.g., imputing missing values, dropping rows or columns).

In [19]:
```
store.columns
```

Out[19]:
```
Index(['Category', 'City', 'Country/Region', 'Customer Name', 'Manufacturer',
       'Order Date', 'Order ID', 'Postal Code', 'Product Name', 'Region',
       'Segment', 'Ship Date', 'Ship Mode', 'State/Province', 'Sub-Category',
       'Discount', 'Profit', 'Quantity', 'Sales'],
      dtype='object')
```

In [20]:
```
store['Category']
```

Out[20]:
```
0        Office Supplies
1        Office Supplies
2        Office Supplies
3        Office Supplies
4        Office Supplies
             ...
10189    Office Supplies
10190    Office Supplies
10191    Office Supplies
10192         Technology
10193    Office Supplies
Name: Category, Length: 10194, dtype: object
```

In [21]:
```
store[['Customer Name', 'City']]
```

Out[21]:

	Customer Name	City
0	Darren Powers	Houston
1	Phillina Ober	Naperville
2	Phillina Ober	Naperville
3	Phillina Ober	Naperville
4	Mick Brown	Philadelphia
...	...	...
10189	Patrick O'Donnell	New York City
10190	Erica Bern	Fairfield
10191	Jill Matthias	Loveland
10192	Patrick O'Donnell	New York City
10193	Harry Olson	Charlottetown

10194 rows × 2 columns

The expression store[['Customer Name', 'City']] is used to select and retrieve only the columns Customer Name and City from the pandas DataFrame named store. This operation results in a new DataFrame that contains just these two columns, preserving all rows that exist in the original store DataFrame.

In [22]:
```
store_categorical = store[['Category', 'City', 'Country/Region', 'Customer Name', 'M
       'Order Date', 'Order ID', 'Postal Code', 'Product Name', 'Region',
       'Segment', 'Ship Date', 'Ship Mode', 'State/Province', 'Sub-Category']]
```

It creates a new pandas DataFrame named store_categorical from the original store DataFrame. This new DataFrame store_categorical includes only the specified columns, which seem to be primarily categorical in nature, along with some other types like dates and potentially numeric or mixed types (e.g., 'Postal Code') depending on the specific data content and structure.

This operation does not modify the original store DataFrame; instead, it creates a subset of store that contains only the columns listed. This can be particularly useful for several reasons:

Focus on Categorical Data: It allows you to focus on the categorical data for analysis or data manipulation tasks that are specific to categorical variables, such as data visualization, encoding for machine learning models, or exploring relationships between categorical features.

Data Cleaning and Preparation: By isolating the categorical (and other selected) columns, you can more easily apply cleaning and preparation steps relevant to these types of data, such as filling missing values in categorical data, converting data types (e.g., converting 'Order Date' and 'Ship Date' to datetime format), or handling categorical data with too many unique values.

Efficiency: Working with a smaller subset of the data can improve performance and efficiency, especially if the original dataset is very large and you're only interested in a specific aspect of the data.

With store_categorical, one might consider performing operations such as:

Checking for Missing Values: Use store_categorical.isnull().sum() to identify missing values in these columns. Exploratory Data Analysis (EDA): Explore the distribution of categorical variables, relationships between them, or how they relate to numerical variables in the dataset. Preparing Data for Machine Learning: This could involve encoding categorical variables using methods like one-hot encoding or ordinal encoding, depending on the nature of the categorical data and the requirements of your analysis or models. Remember, when dealing with dates (such as 'Order Date' and 'Ship Date'), you might need to convert them from string/object type to datetime type using pd.to_datetime() for more efficient manipulation and analysis:

store_categorical['Order Date'] = pd.to_datetime(store_categorical['Order Date'])
store_categorical['Ship Date'] = pd.to_datetime(store_categorical['Ship Date'])

This approach of isolating certain types of data can make your data analysis workflow more organized and efficient.

In [23]:
```
store_categorical #return only categorical data
```

Out[23]:

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code
0	Office Supplies	Houston	United States	Darren Powers	Message Book	03-01-2020	US-2020-103800	77095
1	Office Supplies	Naperville	United States	Phillina Ober	GBC	04-01-2020	US-2020-112326	60540
2	Office Supplies	Naperville	United States	Phillina Ober	Avery	04-01-2020	US-2020-112326	60540
3	Office Supplies	Naperville	United States	Phillina Ober	SAFCO	04-01-2020	US-2020-112326	60540
4	Office Supplies	Philadelphia	United States	Mick Brown	Avery	05-01-2020	US-2020-141817	19143
...	...	...	...	...	...	...	...	...
10189	Office Supplies	New York City	United States	Patrick O'Donnell	Wilson Jones	30-12-2023	US-2023-143259	10009
10190	Office Supplies	Fairfield	United States	Erica Bern	GBC	30-12-2023	US-2023-115427	94533
10191	Office Supplies	Loveland	United States	Jill Matthias	Other	30-12-2023	US-2023-156720	80538
10192	Technology	New York City	United States	Patrick O'Donnell	Other	30-12-2023	US-2023-143259	10009
10193	Office Supplies	Charlottetown	Canada	Harry Olson	Wilson Jones	30-12-2023	CA-2023-143500	C0A

10194 rows × 15 columns

In [24]:
```
store_categorical.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10194 entries, 0 to 10193
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
```

```
 0   Category        10194 non-null  object
 1   City            10194 non-null  object
 2   Country/Region  10194 non-null  object
 3   Customer Name   10194 non-null  object
 4   Manufacturer    10194 non-null  object
 5   Order Date      10194 non-null  object
 6   Order ID        10194 non-null  object
 7   Postal Code     10194 non-null  object
 8   Product Name    10194 non-null  object
 9   Region          10194 non-null  object
 10  Segment         10194 non-null  object
 11  Ship Date       10194 non-null  object
 12  Ship Mode       10194 non-null  object
 13  State/Province  10194 non-null  object
 14  Sub-Category    10194 non-null  object
dtypes: object(15)
memory usage: 1.2+ MB
```

In [25]:
```
store_numerical = store [['Discount', 'Profit', 'Quantity', 'Sales']]
```

In [26]:
```
store_numerical #return only numerical
```

Out[26]:

	Discount	Profit	Quantity	Sales
0	0.2	5.5512	2	16.448
1	0.8	-5.4870	2	3.540
2	0.2	4.2717	3	11.784
3	0.2	-64.7748	3	272.736
4	0.2	4.8840	3	19.536
...	...	...	...	...
10189	0.2	19.7910	3	52.776
10190	0.2	6.4750	2	20.720
10191	0.2	-0.6048	3	3.024
10192	0.0	2.7279	7	90.930
10193	0.2	-0.6048	3	3.024

10194 rows × 4 columns

In [27]:
```
store.head() #return top 5 rows
```

Out[27]:

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code	Produ Nar
0	Office Supplies	Houston	United States	Darren Powers	Message Book	03-01-2020	US-2020-103800	77095	Messa Boo Wirebour Four 5 1/ X 4" Forn
1	Office Supplies	Naperville	United States	Phillina Ober	GBC	04-01-2020	US-2020-112326	60540	G Standa Plas Bindi Syste Com

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code	Produ Nar
2	Office Supplies	Naperville	United States	Phillina Ober	Avery	04-01-2020	US-2020-112326	60540	Avery 5
3	Office Supplies	Naperville	United States	Phillina Ober	SAFCO	04-01-2020	US-2020-112326	60540	SAF(Boltle Ste Shelvi
4	Office Supplies	Philadelphia	United States	Mick Brown	Avery	05-01-2020	US-2020-141817	19143	Avery H Li EverBc Pen Sty Fluoresce

In [28]:
```
store.tail() #bottom 5 records
```

Out[28]:

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code
10189	Office Supplies	New York City	United States	Patrick O'Donnell	Wilson Jones	30-12-2023	US-2023-143259	10009
10190	Office Supplies	Fairfield	United States	Erica Bern	GBC	30-12-2023	US-2023-115427	94533
10191	Office Supplies	Loveland	United States	Jill Matthias	Other	30-12-2023	US-2023-156720	80538
10192	Technology	New York City	United States	Patrick O'Donnell	Other	30-12-2023	US-2023-143259	10009
10193	Office Supplies	Charlottetown	Canada	Harry Olson	Wilson Jones	30-12-2023	CA-2023-143500	C0A

In [29]:
```
store[:]
```

Out[29]:

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code
0	Office Supplies	Houston	United States	Darren Powers	Message Book	03-01-2020	US-2020-103800	77095

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code
1	Office Supplies	Naperville	United States	Phillina Ober	GBC	04-01-2020	US-2020-112326	60540
2	Office Supplies	Naperville	United States	Phillina Ober	Avery	04-01-2020	US-2020-112326	60540
3	Office Supplies	Naperville	United States	Phillina Ober	SAFCO	04-01-2020	US-2020-112326	60540
4	Office Supplies	Philadelphia	United States	Mick Brown	Avery	05-01-2020	US-2020-141817	19143
...	...	...	...	...	...	...	...	...
10189	Office Supplies	New York City	United States	Patrick O'Donnell	Wilson Jones	30-12-2023	US-2023-143259	10009
10190	Office Supplies	Fairfield	United States	Erica Bern	GBC	30-12-2023	US-2023-115427	94533
10191	Office Supplies	Loveland	United States	Jill Matthias	Other	30-12-2023	US-2023-156720	80538
10192	Technology	New York City	United States	Patrick O'Donnell	Other	30-12-2023	US-2023-143259	10009
10193	Office Supplies	Charlottetown	Canada	Harry Olson	Wilson Jones	30-12-2023	CA-2023-143500	C0A

10194 rows × 19 columns

In [30]:
```
store[0:10] #top 9records
```

Out[30]:

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code	Produ Nar
0	Office Supplies	Houston	United States	Darren Powers	Message Book	03-01-2020	US-2020-103800	77095	Messa Boo Wirebour Four 5 1/ X 4" Forn
1	Office Supplies	Naperville	United States	Phillina Ober	GBC	04-01-	US-2020-	60540	G Standa

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code	Produ Nar
						2020	112326		Plas Bindi Syste Com
2	Office Supplies	Naperville	United States	Phillina Ober	Avery	04-01-2020	US-2020-112326	60540	Avery 5
3	Office Supplies	Naperville	United States	Phillina Ober	SAFCO	04-01-2020	US-2020-112326	60540	SAF(Boltle Ste Shelvi
4	Office Supplies	Philadelphia	United States	Mick Brown	Avery	05-01-2020	US-2020-141817	19143	Avery H Li EverBc Pen Sty Fluoresce
5	Furniture	Henderson	United States	Maria Etezadi	Global	06-01-2020	US-2020-167199	42420	Glob Delu High-Ba Manage Ch
6	Office Supplies	Henderson	United States	Maria Etezadi	Rogers	06-01-2020	US-2020-167199	42420	Roge Handhe Bar Pen Sharper
7	Office Supplies	Athens	United States	Jack O'Briant	Dixon	06-01-2020	US-2020-106054	30605	Dix Pra Waterco Pencils, 1 Color S v
8	Office Supplies	Henderson	United States	Maria Etezadi	Ibico	06-01-2020	US-2020-167199	42420	Ibico H Te Manu Bindi Syste
9	Office Supplies	Henderson	United States	Maria Etezadi	Alliance	06-01-2020	US-2020-167199	42420	Allian Super-Si Banc Assort Siz

In [31]:
```
store[::-1]
```

Out[31]:

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code
10193	Office Supplies	Charlottetown	Canada	Harry Olson	Wilson Jones	30-12-2023	CA-2023-143500	C0A

	Category	City	Country/Region	Customer Name	Manufacturer	Order Date	Order ID	Postal Code
10192	Technology	New York City	United States	Patrick O'Donnell	Other	30-12-2023	US-2023-143259	10009
10191	Office Supplies	Loveland	United States	Jill Matthias	Other	30-12-2023	US-2023-156720	80538
10190	Office Supplies	Fairfield	United States	Erica Bern	GBC	30-12-2023	US-2023-115427	94533
10189	Office Supplies	New York City	United States	Patrick O'Donnell	Wilson Jones	30-12-2023	US-2023-143259	10009
...	...	...	...	...	...	...	...	...
4	Office Supplies	Philadelphia	United States	Mick Brown	Avery	05-01-2020	US-2020-141817	19143
3	Office Supplies	Naperville	United States	Phillina Ober	SAFCO	04-01-2020	US-2020-112326	60540
2	Office Supplies	Naperville	United States	Phillina Ober	Avery	04-01-2020	US-2020-112326	60540
1	Office Supplies	Naperville	United States	Phillina Ober	GBC	04-01-2020	US-2020-112326	60540
0	Office Supplies	Houston	United States	Darren Powers	Message Book	03-01-2020	US-2020-103800	77095

10194 rows × 19 columns

In [32]:
```
store.duplicated().sum() #returned number of duplicates
```

Out[32]: 2

In [33]:
```
duplicate_rows = store.duplicated(keep='first') #Find duplicate rows, keeping the fi
```

In [34]:
```
store_duplicates_only = store[duplicate_rows] # Use boolean indexing to filter and d
```

In [35]:
```
print(store_duplicates_only) # Display the duplicate rows
```

```
      Category        City Country/Region   Customer Name Manufacturer  \
391   Furniture    Columbus  United States  Laurel Beltran       Global
1699  Furniture  St. John's         Canada  James Peterman      Nu-Dell

      Order Date        Order ID Postal Code  \
391   23-04-2020  US-2020-150119       43229
1699  24-11-2020  CA-2020-153623         A0A

                                          Product Name Region      Segment  \
391   Global Leather Highback Executive Chair with P...   East  Home Office
1699                          Nu-Dell Executive Frame   East    Corporate

       Ship Date       Ship Mode             State/Province Sub-Category  \
391   27-04-2020  Standard Class                       Ohio       Chairs
1699  05-12-2020  Standard Class  Newfoundland and Labrador  Furnishings

      Discount   Profit  Quantity    Sales
391        0.3 -12.0588         2  281.372
1699       0.0  35.4144         8   99.120
```

store.duplicated(keep='first') returns a boolean Series, where True indicates a row is a duplicate of an earlier row in the DataFrame, according to the criteria you've specified with keep='first'. This means the first occurrence is not considered a duplicate. store[duplicate_rows] uses boolean indexing to select rows from store that are marked as True in duplicate_rows, effectively filtering the DataFrame to include only duplicate rows. store_duplicates_only now contains only the rows that are duplicates (excluding the first occurrence of those duplicates), and printing it will show you these rows. If you want to include the first occurrence of the duplicates in your display (i.e., show all instances of data that have duplicates), you can use keep=False:

In [36]:
```
all_duplicates = store.duplicated(keep=False) # Mark all duplicates as True, includi
```

Data and Time Conversion

Convert columns containing dates and times into datetime objects to facilitate time series analysis or extract specific date-related features like the day of the week, month, or year.

To perform date and time conversion in pandas, particularly for the dataset you're working with, you'd typically want to convert string representations of dates into datetime objects. This conversion allows you to easily perform time-based operations, such as filtering by date, extracting components like month or year, and resampling for time series analysis.

Given your dataset contains columns like 'Order Date' and 'Ship Date' which likely represent dates, you can convert these columns using pd.to_datetime() function from pandas. Here's how you can do it:

In [37]:
```
store['Order Date'] = pd.to_datetime(store['Order Date']) # Convert 'Order Date' and

store['Ship Date'] = pd.to_datetime(store['Ship Date'])
```

In [38]:
```
print(store.dtypes) # Verify the conversion by checking the data types again
```

```
Category                  object
City                      object
Country/Region            object
Customer Name             object
Manufacturer              object
Order Date        datetime64[ns]
Order ID                  object
```

```
Postal Code              object
Product Name             object
Region                   object
Segment                  object
Ship Date         datetime64[ns]
Ship Mode                object
State/Province           object
Sub-Category             object
Discount                float64
Profit                  float64
Quantity                  int64
Sales                   float64
dtype: object
```

What Does This Do? pd.to_datetime(): This function converts a string (or series of strings) into a pandas datetime object. When applied to a DataFrame column, each string in that column is converted into a datetime object, which represents a point in time. store['Order Date'] = pd.to_datetime(store['Order Date']): This line converts the 'Order Date' column from its current format (likely a string) into datetime. store['Ship Date'] = pd.to_datetime(store['Ship Date']): Similarly, this line converts the 'Ship Date' column. After Conversion Once the dates are converted to datetime objects, you can perform a variety of date-related operations, such as:

In [39]:
```
start_date = pd.to_datetime('2020-01-01')
end_date = pd.to_datetime('2020-12-31')
filtered_store = store[(store['Order Date'] >= start_date) & (store['Order Date'] <=
```

In [40]:
```
store['Order Year'] = store['Order Date'].dt.year
store['Order Month'] = store['Order Date'].dt.month
#Extracting Date Components
```

In [41]:
```
store['Shipping Time'] = store['Ship Date'] - store['Order Date']
#Calculating Durations
```

In [42]:
```
store['Shipping Time']
```

Out[42]:
```
0       122 days
1       122 days
2       122 days
3       122 days
4       214 days
          ...
10189    62 days
10190    62 days
10191    62 days
10192    62 days
10193    62 days
Name: Shipping Time, Length: 10194, dtype: timedelta64[ns]
```

Data Visualization

In [43]:
```
pip install pandas matplotlib seaborn
```

```
Requirement already satisfied: pandas in c:\users\me\anaconda3\lib\site-packages (1.
1.3)
Requirement already satisfied: matplotlib in c:\users\me\anaconda3\lib\site-packages
(3.3.2)
Requirement already satisfied: seaborn in c:\users\me\anaconda3\lib\site-packages (0.
11.0)
Requirement already satisfied: numpy>=1.15.4 in c:\users\me\anaconda3\lib\site-packag
es (from pandas) (1.19.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\me\anaconda3\lib\si
```

```
te-packages (from pandas) (2.8.1)Note: you may need to restart the kernel to use upda
ted packages.

Requirement already satisfied: pytz>=2017.2 in c:\users\me\anaconda3\lib\site-package
s (from pandas) (2020.1)
Requirement already satisfied: cycler>=0.10 in c:\users\me\anaconda3\lib\site-package
s (from matplotlib) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\me\anaconda3\lib\site-packag
es (from matplotlib) (8.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\me\anaconda3\lib\site-pa
ckages (from matplotlib) (1.3.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\m
e\anaconda3\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: certifi>=2020.06.20 in c:\users\me\anaconda3\lib\site-
packages (from matplotlib) (2020.6.20)
Requirement already satisfied: scipy>=1.0 in c:\users\me\anaconda3\lib\site-packages
(from seaborn) (1.5.2)
Requirement already satisfied: six>=1.5 in c:\users\me\anaconda3\lib\site-packages (f
rom python-dateutil>=2.7.3->pandas) (1.15.0)
```

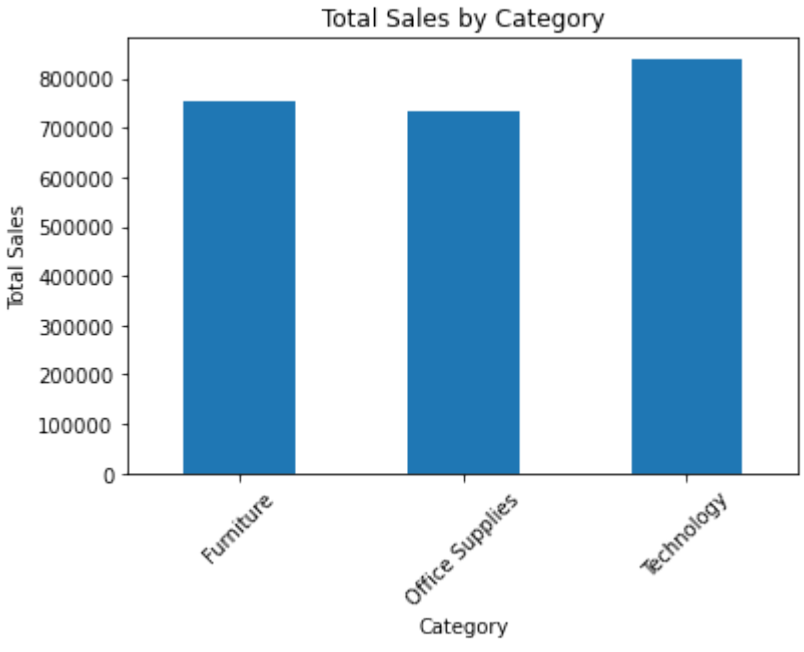
Importing Libraries

In [44]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Sales Distribution by Category

In [45]:
```
# Calculate total sales by category
sales_by_category = store.groupby('Category')['Sales'].sum()

# Plot
sales_by_category.plot(kind='bar')
plt.title('Total Sales by Category')
plt.xlabel('Category')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)  # Rotate category names for better readability
plt.show()
```



The bar chart titled "Total Sales by Category" displays the total sales for three product categories: Furniture, Office Supplies, and Technology. Each bar represents the aggregate sales for each category over a specified time period.

Furniture: The sales for Furniture are shown by the first bar. The category has generated a total of approximately $700,000 in sales, indicating a strong performance, although not the highest among the three categories.

Office Supplies: The second bar represents the Office Supplies category. It appears to have the lowest total sales among the three categories, with sales around $600,000. While this is the lowest, its still a substantial contribution to the overall sales.

Technology: The final bar, representing the Technology category, shows the highest total sales, which are close to $800,000. This suggests that products in the Technology category are a major driver of revenue and may represent a key area for the business.

Sales Performance by Product Category

The sales performance analysis for our three primary product categories over the recent period. The enclosed bar chart, "Total Sales by Category," articulates the sales volume for each category, providing us with insightful data to inform our strategic decisions moving forward.

Technology as a Revenue Leader: It is clear from the visual data that our Technology category stands out as the revenue leader with the highest total sales, nearly reaching the $800,000 mark. This dominance in sales suggests that our Technology offerings resonate strongly with our customer base, possibly due to cutting-edge product features or alignment with current market trends.

Market Dynamics and Sales Drivers: The Furniture category shows healthy sales performance with total sales around $700,000.

While this is commendable, understanding market dynamics, such as seasonal demands or consumer preferences that have influenced these figures, could offer us opportunities to optimize sales further.

The Office Supplies category, while slightly trailing with sales approximately at the $600,000 level, still represents a significant portion of our business and will benefit from targeted strategies to bolster its market position.

Strategic Recommendations: Given the robust sales in the Technology category, I suggest a strategy that continues to capitalize on this momentum. This may involve increasing inventory for high-demand tech products, exploring expansion into emerging tech markets, or intensifying marketing efforts to further enhance this category visibility and reach.

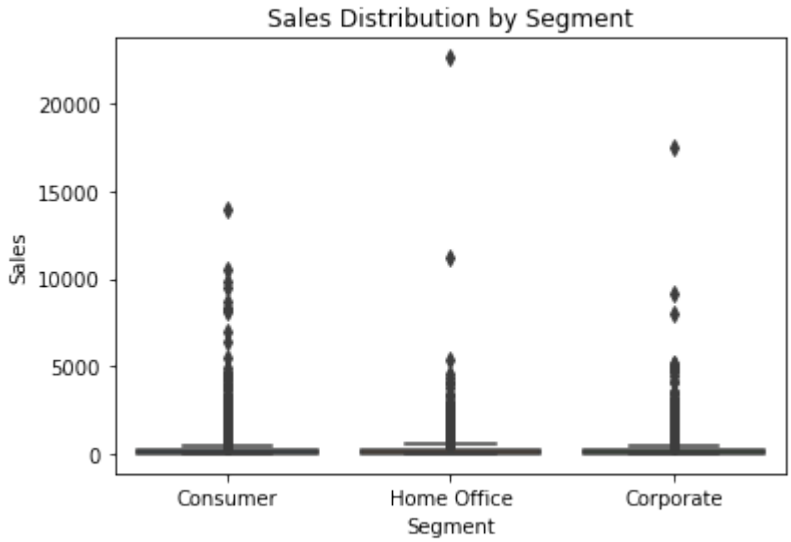
Further Analytical Opportunities: To refine our sales strategy, I propose a deeper dive into the sales data within each category. By analyzing sales at the individual product level, I can identify star performers and underperformers, thereby tailoring our inventory and marketing investments more effectively. Additionally, a review of the profit margins associated with these sales figures will ensure that our high-performing sales categories are not just generating revenue, but are also profitable.

Attached is the bar chart illustrating these insights. I look forward to discussing these findings in more detail and developing targeted strategies to enhance sales performance across all categories.

Box Plot of Sales by Segment

A box plot can help you understand the distribution of sales across different customer segments and spot any outliers.

In [46]:
```
# Box plot
sns.boxplot(x='Segment', y='Sales', data=store)
plt.title('Sales Distribution by Segment')
plt.xlabel('Segment')
plt.ylabel('Sales')
plt.show()
```



The box plot titled "Sales Distribution by Segment" visually represents the spread and central tendencies of sales data across three different customer segments: Consumer, Home Office, and Corporate. Here's a detailed explanation of the components of the box plot and what they reveal about the sales data:

Median (The Middle Line of the Box): The line within the box represents the median sales in each segment, which is the midpoint of the data where half of the sales numbers are above and half are below.

Interquartile Range (IQR - The Box Length): The box itself shows the middle 50% of data, known as the interquartile range. The bottom and top of the box represent the first quartile (25th percentile) and third quartile (75th percentile) of the sales data, respectively. A larger box indicates greater variability in sales within that segment.

Whiskers (Lines Extending from the Box): The "whiskers" extend from the top and bottom of the box to the highest and lowest sales within 1.5 times the interquartile range from the box. Points beyond the whiskers are considered outliers.

Outliers (Individual Points): Outliers are the points that lie beyond the whiskers. They represent sales that are unusually high or low compared to the rest of the data in that segment. In this plot, each segment has outliers, indicating individual sales that stand out from the norm.

Interpretation: Consumer Segment: Shows a relatively large range of sales, with the median sales lower than the highest outlier, which suggests there is a mix of small and a few very large sales transactions.

Home Office Segment: This segment seems to have a narrower IQR, which implies that the sales figures are more consistent. The median is lower compared to the Consumer segment, and there are fewer outliers, indicating that extreme sales values are less common in this segment.

Corporate Segment: The Corporate segment sales distribution looks somewhat similar to the Consumer segment but with the median closer to the first quartile, indicating that more sales fall on the lower end. However, like the Consumer segment, there are high-value outliers, showing that some large sales transactions occur in this segment as well.

This box plot is valuable for identifying which segments have the most consistent sales, where there might be opportunities to increase sales, and where to focus customer relationship efforts to capitalize on high-value sales opportunities.

Understanding Sales Variability Across Customer Segments

The "Sales Distribution by Segment" box plot, which offers a visual representation of the sales data across the three key customer segments. This graph is instrumental in understanding the spread and central tendency of sales in each segment, helping to inform targeted sales strategies.

Consumer Segment: This segment exhibits a wide range of sales, with a relatively even distribution around the median, suggesting that our consumer base has diverse purchasing patterns.

While the bulk of sales are concentrated under the $5,000 mark, there are notable instances of high value sales reaching up to 20000.

Home Office Segment: Sales in this segment tend to be more centrally clustered, with less variability in the interquartile range. The median sales value appears to be lower than the Consumer segment, with fewer high-value outliers. This may indicate more consistent but smaller transactions within the Home Office segment.

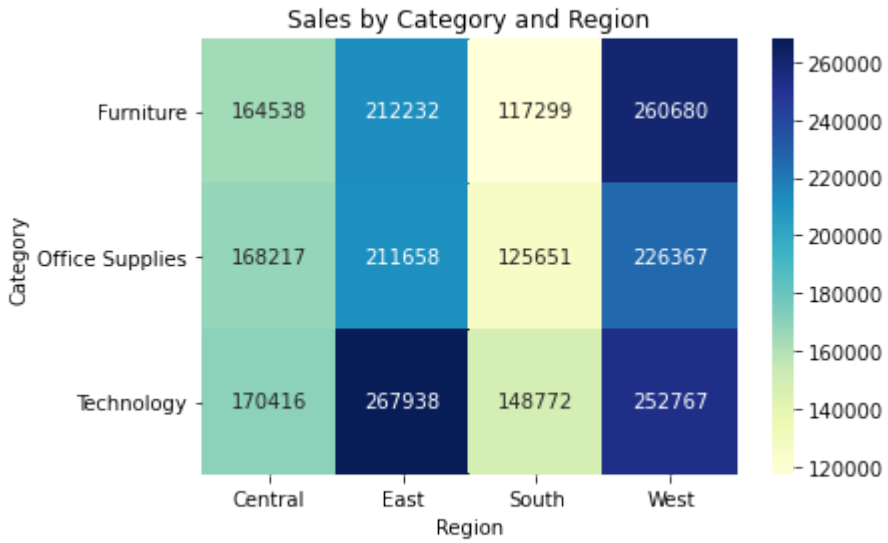
Corporate Segment: The Corporate segment shows a distribution similar to the Consumer segment but with a slightly higher concentration of sales in the lower quartile. The presence of high-value outliers is also noticeable, suggesting that while many transactions are of lower value, there are significant opportunities in high-value corporate sales.

Heatmap of Sales by Category and Region

A heatmap can visualize the concentration of sales across categories and regions.

In [47]:
```
# Pivot table for heatmap
sales_pivot = store.pivot_table(values='Sales', index='Category', columns='Region',

# Heatmap
sns.heatmap(sales_pivot, annot=True, fmt=".0f", cmap='YlGnBu')
plt.title('Sales by Category and Region')
plt.xlabel('Region')
plt.ylabel('Category')
plt.show()
```



Report/Analysis:

Regional Sales Performance Analysis

The Sales by Category and Region heatmap, offers a strategic overview of our sales distribution across various regions and product categories. This visual tool is pivotal in identifying regional market strengths and opportunities for targeted growth.

Overview of Findings:

Furniture:

Shows strong sales in the West region, suggesting a robust market presence or favorable customer preference. The South region reflects the lowest sales for Furniture, which could indicate untapped market potential or the need for improved marketing strategies. Office Supplies:

The sales performance in the West region is the most notable, again highlighting the West as a key market for our offerings. Comparatively, sales in the South region are lower, similar to the trend observed in the Furniture category. Technology:

The East region leads with the highest sales, pointing to a successful market penetration or a high demand for technology products. Interestingly, Technology sales in the South are more on par with the Central region, differing from the trends seen in the other categories. Key Insights:

The West region is a strong performer for Furniture and Office Supplies, which could be leveraged for further market expansion and strategic promotions.

The East region's exceptional performance in Technology sales implies a significant market opportunity that we should continue to nurture and possibly use as a benchmark for other regions.

The South region shows consistently lower sales across all categories, signaling a need for a comprehensive review of our regional strategy, including marketing initiatives, distribution channels, or a localized consumer behavior analysis.

Across all regions, Office Supplies show a balanced distribution, suggesting a stable demand that we can depend on for steady revenue.

Strategic Recommendations:

Capitalize on Strengths: Continue to invest in the West region for Furniture and Office Supplies, and in the East for Technology, potentially exploring upselling strategies or expanding product lines.

Address Regional Gaps: Develop tailored approaches for the South region to understand and overcome the barriers to sales performance.

Leverage Consistency: Utilize the consistent performance of Office Supplies to ensure a stable revenue base, while exploring ways to innovate and capture additional market share.

In [ ]:

In [ ]: