Importing the required libraries for data manipulation and visualization

In [39]:

```
import numpy as np
import pandas as pd

import os
import datetime as dt
from dateutil import parser

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
```

Importing Dataset

In [2]:

```
df = pd.read_csv(os.path.join(os.path.expanduser("~"), "Downloads/Portfolio Datasets/Python/Messy_E-commerce_Orders_Dataset.csv"))
```

Taking a look at data

In [3]:

```
df.shape
```

Out[3]:

```
(200, 8)
```

In [4]:

```
df
```

Out[4]:

	Order ID	Order Date	Customer Name	City	Product Info	Price	Quantity	Shipping Date
0	f6722f25-3aac-48b4-ad00-565568085bd0	09-01-2025	Brandon Greene	LA	Shirt - Clothing	NaN	NaN	NaN
1	4708dace-5f9f-44c9-a5de-33744071e8a9	2024-03-11	Debra Murphy	LA	Shoes - Apparel	1493.46	3.0	14-03-2024
2	2b4bb345-8c7d-4323-8fc9-6666d92c2812	May 12, 2024	Diane Knight	San Francisco	Shirt - Clothing	1858.36	3.0	May 18, 2024
3	96fc6afd-d9f8-4020-ae5c-9d14fbd190e7	2024-12-05	Alexandra Sharp	san francisco	TV - Electronics	1524.00	3.0	NaN
4	b2ed6622-0688-4a5f-ae9f-3cda8bd043fd	2024-01-20	Michael Simpson	san francisco	Shirt - Clothing	183.23	2.0	2024-01-29
...	...	...	...	...	...	...	...	...
195	0e7c81ad-adcf-48ea-9016-15df8e841ad3	Oct 14, 2024	Amy White	Los Angeles	TV - Electronics	NaN	3.0	15-10-2024
196	c343a87c-5fc4-4af6-b85d-e8eee84347d8	Aug 30, 2024	Lindsey Martin	Los Angeles	Laptop - Electronics	NaN	3.0	NaN
197	0f038ff6-bd4a-4e9d-b1e3-c0512a698c5d	2023-07-15	Katelyn Taylor	New York	Shoes - Apparel	1216.54	2.0	Jul 25, 2023
198	252bb286-97af-43ea-9253-0cd43cad8c70	2023-12-16	Jennifer Acosta	New York	Shirt - Clothing	NaN	1.0	NaN
199	56c969df-2e17-4f68-9fb8-03e340819612	Oct 29, 2023	Michael Martin	LA	Laptop - Electronics	1064.08	3.0	NaN

200 rows × 8 columns

Issues Visible in the First Glance

- The 'Order Date' Column seems to have data in multiple formats eg: 'Mar 20, 2024', '2024-10-13', '2023-09-17'
- The City column seems to have same data but in different cases eg: 'New York', 'new york'
- The Price, Quantity and Shipping Date seem to have null values as well

Inspecting Data for any null values

In [5]:

```
# df.info()
df.isnull().sum()
```

Out[5]:

```
Order ID           0
Order Date         0
Customer Name      0
City               0
Product Info       0
Price            129
Quantity          56
Shipping Date     66
dtype: int64
```

Cleaning The Data

Starting from the left cleaning all the data in the dataset

Order Date Column

Since Only three formats of datetime are seen in the Order Date Column, we are creatig a function that does the following:

- Step 1: Converts Data to str and Removes The Leading/Trailing whitespace
- Step 2: Checks if the data is in any of the three formats ("%Y-%m-%d", "%d-%m-%Y", "%b %d, %Y") and converts it to date,
- Step 3: If none of the formats match converts the data to NAT

In [6]:

```
# df.columns
#Creating a date time formatting function
def parse_date_fallback(date_str):

    if pd.isna(date_str):
        return pd.NaT


    date_str = str(date_str).strip()

    for fmt in ("%Y-%m-%d", "%d-%m-%Y", "%b %d, %Y"):
        try:
            return dt.datetime.strptime(date_str, fmt)
        except:
            continue
    return pd.NaT  # If all formats fail

#Applying the function to 'Order Date' Column
df['Order Date'] = df['Order Date'].apply(parse_date_fallback)
```

Checking the Order Date column for null

```
In [7]: df['Order Date'].isna().sum()
```

```
Out[7]: 0
```

- No Null Values shown
- Checking the Top 10 values

```
In [8]: df['Order Date'].head(10)
```

```
Out[8]: 0   2025-01-09
        1   2024-03-11
        2   2024-05-12
        3   2024-12-05
        4   2024-01-20
        5   2025-03-13
        6   2023-12-06
        7   2024-05-23
        8   2024-08-28
        9   2024-09-22
        Name: Order Date, dtype: datetime64[ns]
```

City Column

Checking the data available in the City Column

```
In [9]: df['City'].value_counts()
```

```
Out[9]: City
        LA               38
        San Francisco    35
        san francisco    35
        New York         34
        Los Angeles      32
        new york         26
        Name: count, dtype: int64
```

Since Same information(City Name) is in different Format, We are going to standarize the city names

```
In [10]: #Creating a city_standarize dictionary that maps city names
         city_standarize = {
             "LA": "Los Angeles",
             "new york": "New York",
             "san francisco": "San Francisco"
         }
         df['City'] = df['City'].replace(city_standarize).str.title()
```

```
In [11]: df['City'].value_counts()
```

```
Out[11]: City
         Los Angeles      70
         San Francisco    70
         New York         60
         Name: count, dtype: int64
```

City Names Now dont have multiple values for same city name

Product Info Column

```
In [12]: df['Product Info'].value_counts()
```

```
Out[12]: Product Info
         Laptop - Electronics    61
         Shoes - Apparel         53
         TV - Electronics        44
         Shirt - Clothing        42
         Name: count, dtype: int64
```

The 'Product Info' column seem to have Product Type and Category Merged into one using '-' delimeter but Since there are not much differet data in it we are not going to seperate the columns into two.

Price Column

In [13]:

```
df.head()
```

Out[13]:

	Order ID	Order Date	Customer Name	City	Product Info	Price	Quantity	Shipping Date
0	f6722f25-3aac-48b4-ad00-565568085bd0	2025-01-09	Brandon Greene	Los Angeles	Shirt - Clothing	NaN	NaN	NaN
1	4708dace-5f9f-44c9-a5de-33744071e8a9	2024-03-11	Debra Murphy	Los Angeles	Shoes - Apparel	1493.46	3.0	14-03-2024
2	2b4bb345-8c7d-4323-8fc9-6666d92c2812	2024-05-12	Diane Knight	San Francisco	Shirt - Clothing	1858.36	3.0	May 18, 2024
3	96fc6afd-d9f8-4020-ae5c-9d14fbd190e7	2024-12-05	Alexandra Sharp	San Francisco	TV - Electronics	1524.00	3.0	NaN
4	b2ed6622-0688-4a5f-ae9f-3cda8bd043fd	2024-01-20	Michael Simpson	San Francisco	Shirt - Clothing	183.23	2.0	2024-01-29

For the price column, filling the missing data is quite tricky so we apply the following steps

- Step 1: Create Unit Price column i.e Price / Quantity
- Step 2: Filter Out the rows where both Price and Quantity is empty(as it does not make sense to fill the quantity column with any kind of average value)
- Step 3: For the Columns with only null values in Price Column but data in Quantity Column, fill the missing unit price with the median value based on quarter and city. If the median value is null for specific quarter, fill it with the median value of previous quarter
- Step 4: Fill the Price column by multiplying the Unit Price Column with Quantity.

In [14]:

```
df['Unit Price'] = df['Price']/df['Quantity']
```

In [15]:

```
quantity_mask = (~ df['Quantity'].isna())
price_quantity_mask = (~df['Price'].isna() & ~df['Quantity'].isna())
```

Creating a new dataframe that does not contain Null value in Quantity Column

In [16]:

```
df2 = df[(quantity_mask)]
```

Checking the number of rows and columns in the new dataframe

In [17]:

```
df2.shape
```

Out[17]:

```
(144, 9)
```

In [18]:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 144 entries, 1 to 199
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   Order ID       144 non-null    object        
 1   Order Date     144 non-null    datetime64[ns]
 2   Customer Name  144 non-null    object        
 3   City           144 non-null    object        
 4   Product Info   144 non-null    object        
 5   Price          52 non-null     float64       
 6   Quantity       144 non-null    float64       
 7   Shipping Date  96 non-null     object        
 8   Unit Price     52 non-null     float64       
dtypes: datetime64[ns](1), float64(3), object(5)
memory usage: 11.2+ KB
```

In [19]:

```
# Step 1: Calculate the median Unit Price for each City + Product Info
unit_price_medians = df2.groupby(['City', 'Product Info'])['Unit Price'].media
n()


# Step 2: Mask for missing Unit Price
mask = df2['Unit Price'].isna()

# Step 3: Use .loc and .map to fill values
df2.loc[mask, 'Unit Price'] = (
    df2.loc[mask]
        .set_index(['City', 'Product Info'])
        .index.map(unit_price_medians)
)
```

In [20]:

```
# Creating a mask for rows where Price is NaN
price_mask = df2['Price'].isna()

# Filling missing Price using Unit Price * Quantity
df2.loc[price_mask, 'Price'] = df2.loc[price_mask, 'Unit Price'] * df2.loc[pri
ce_mask, 'Quantity']
```

In [21]:
```
df2
```

Out[21]:

	Order ID	Order Date	Customer Name	City	Product Info	Price	Quantity	Shipping Date	Unit P
1	4708dace-5f9f-44c9-a5de-33744071e8a9	2024-03-11	Debra Murphy	Los Angeles	Shoes - Apparel	1493.460	3.0	14-03-2024	497.820
2	2b4bb345-8c7d-4323-8fc9-6666d92c2812	2024-05-12	Diane Knight	San Francisco	Shirt - Clothing	1858.360	3.0	May 18, 2024	619.453
3	96fc6afd-d9f8-4020-ae5c-9d14fbd190e7	2024-12-05	Alexandra Sharp	San Francisco	TV - Electronics	1524.000	3.0	NaN	508.000
4	b2ed6622-0688-4a5f-ae9f-3cda8bd043fd	2024-01-20	Michael Simpson	San Francisco	Shirt - Clothing	183.230	2.0	2024-01-29	91.615
6	279ccb64-f197-42b1-93c6-bbfc1c70a755	2023-12-06	Brenda Jackson	New York	Laptop - Electronics	1932.255	3.0	NaN	644.085
...	...	...	...	...	...	...	...	...	
195	0e7c81ad-adcf-48ea-9016-15df8e841ad3	2024-10-14	Amy White	Los Angeles	TV - Electronics	734.540	3.0	15-10-2024	244.846
196	c343a87c-5fc4-4af6-b85d-e8eee84347d8	2024-08-30	Lindsey Martin	Los Angeles	Laptop - Electronics	1307.310	3.0	NaN	435.770
197	0f038ff6-bd4a-4e9d-b1e3-c0512a698c5d	2023-07-15	Katelyn Taylor	New York	Shoes - Apparel	1216.540	2.0	Jul 25, 2023	608.270
198	252bb286-97af-43ea-9253-0cd43cad8c70	2023-12-16	Jennifer Acosta	New York	Shirt - Clothing	756.060	1.0	NaN	756.060
199	56c969df-2e17-4f68-9fb8-03e340819612	2023-10-29	Michael Martin	Los Angeles	Laptop - Electronics	1064.080	3.0	NaN	354.693

144 rows × 9 columns

Shipping Date Column

Since the Shipping Date Column also has date in multiple date format, we will change the format to one single type just like Order Date column. We will use the same funtion that was previously created and used i.e parse_date_fallback

In [ ]:

```
df['Shipping Date'] = df['Shipping Date'].apply(parse_date_fallback)
df2['Shipping Date'] = df2['Shipping Date'].apply(parse_date_fallback)
```

In [23]:

```
df2.head()
```

Out[23]:

	Order ID	Order Date	Customer Name	City	Product Info	Price	Quantity	Shipping Date	Unit Pric
1	4708dace-5f9f-44c9-a5de-33744071e8a9	2024-03-11	Debra Murphy	Los Angeles	Shoes - Apparel	1493.460	3.0	2024-03-14	497.82000
2	2b4bb345-8c7d-4323-8fc9-6666d92c2812	2024-05-12	Diane Knight	San Francisco	Shirt - Clothing	1858.360	3.0	2024-05-18	619.45333
3	96fc6afd-d9f8-4020-ae5c-9d14fbd190e7	2024-12-05	Alexandra Sharp	San Francisco	TV - Electronics	1524.000	3.0	NaT	508.00000
4	b2ed6622-0688-4a5f-ae9f-3cda8bd043fd	2024-01-20	Michael Simpson	San Francisco	Shirt - Clothing	183.230	2.0	2024-01-29	91.61500
6	279ccb64-f197-42b1-93c6-bbfc1c70a755	2023-12-06	Brenda Jackson	New York	Laptop - Electronics	1932.255	3.0	NaT	644.08500

Since some of the data in Shipping Date column is Null, we will first calculate the average no of days between order date and shipping date

In [ ]:

```
df2['Days to Ship'] = (df2['Shipping Date'] - df2['Order Date']).dt.days
```

In [25]:
```python
df2.head()
```

Out[25]:

	Order ID	Order Date	Customer Name	City	Product Info	Price	Quantity	Shipping Date	Unit Pric
1	4708dace-5f9f-44c9-a5de-33744071e8a9	2024-03-11	Debra Murphy	Los Angeles	Shoes - Apparel	1493.460	3.0	2024-03-14	497.8200
2	2b4bb345-8c7d-4323-8fc9-6666d92c2812	2024-05-12	Diane Knight	San Francisco	Shirt - Clothing	1858.360	3.0	2024-05-18	619.4533
3	96fc6afd-d9f8-4020-ae5c-9d14fbd190e7	2024-12-05	Alexandra Sharp	San Francisco	TV - Electronics	1524.000	3.0	NaT	508.0000
4	b2ed6622-0688-4a5f-ae9f-3cda8bd043fd	2024-01-20	Michael Simpson	San Francisco	Shirt - Clothing	183.230	2.0	2024-01-29	91.6150
6	279ccb64-f197-42b1-93c6-bbfc1c70a755	2023-12-06	Brenda Jackson	New York	Laptop - Electronics	1932.255	3.0	NaT	644.0850

Now we are going to fill the Null Values with average days by city

In [26]:
```python
# Step 1: Compute average Days to Ship per City
city_avg_days = df2.groupby('City')['Days to Ship'].mean().round()

# Step 2: Create a mask for rows with NaN Days to Ship
days_mask = df2['Days to Ship'].isna()

# Step 3: Fill NaNs using .loc and mapping from city_avg_days
df2.loc[days_mask, 'Days to Ship'] = df2.loc[days_mask, 'City'].map(city_avg_d
ays)
```

In [27]:

```
df2.head()
```

Out[27]:

	Order ID	Order Date	Customer Name	City	Product Info	Price	Quantity	Shipping Date	Unit Pric
1	4708dace-5f9f-44c9-a5de-33744071e8a9	2024-03-11	Debra Murphy	Los Angeles	Shoes - Apparel	1493.460	3.0	2024-03-14	497.82000
2	2b4bb345-8c7d-4323-8fc9-6666d92c2812	2024-05-12	Diane Knight	San Francisco	Shirt - Clothing	1858.360	3.0	2024-05-18	619.45333
3	96fc6afd-d9f8-4020-ae5c-9d14fbd190e7	2024-12-05	Alexandra Sharp	San Francisco	TV - Electronics	1524.000	3.0	NaT	508.00000
4	b2ed6622-0688-4a5f-ae9f-3cda8bd043fd	2024-01-20	Michael Simpson	San Francisco	Shirt - Clothing	183.230	2.0	2024-01-29	91.61500
6	279ccb64-f197-42b1-93c6-bbfc1c70a755	2023-12-06	Brenda Jackson	New York	Laptop - Electronics	1932.255	3.0	NaT	644.08500

Finally filling the Shipping Date column by adding Order Date and Days to Ship

In [28]:

```
shipping_mask = df2['Shipping Date'].isna()

# Step 2: Fill missing Shipping Date = Order Date + Days to Ship
df2.loc[shipping_mask, 'Shipping Date'] = pd.to_datetime(df2.loc[shipping_mask, 'Order Date']) + pd.to_timedelta(df2.loc[shipping_mask, 'Days to Ship'], unit='D')
```

In [29]:

```
df2.head()
```

Out[29]:

	Order ID	Order Date	Customer Name	City	Product Info	Price	Quantity	Shipping Date	Unit Pric
1	4708dace-5f9f-44c9-a5de-33744071e8a9	2024-03-11	Debra Murphy	Los Angeles	Shoes - Apparel	1493.460	3.0	2024-03-14	497.8200(
2	2b4bb345-8c7d-4323-8fc9-6666d92c2812	2024-05-12	Diane Knight	San Francisco	Shirt - Clothing	1858.360	3.0	2024-05-18	619.4533:
3	96fc6afd-d9f8-4020-ae5c-9d14fbd190e7	2024-12-05	Alexandra Sharp	San Francisco	TV - Electronics	1524.000	3.0	2024-12-11	508.0000(
4	b2ed6622-0688-4a5f-ae9f-3cda8bd043fd	2024-01-20	Michael Simpson	San Francisco	Shirt - Clothing	183.230	2.0	2024-01-29	91.6150(
6	279ccb64-f197-42b1-93c6-bbfc1c70a755	2023-12-06	Brenda Jackson	New York	Laptop - Electronics	1932.255	3.0	2023-12-12	644.0850(

Finally Checking if there is any null values in any of the columns

In [30]:

```
df2.isna().sum()
```

Out[30]:

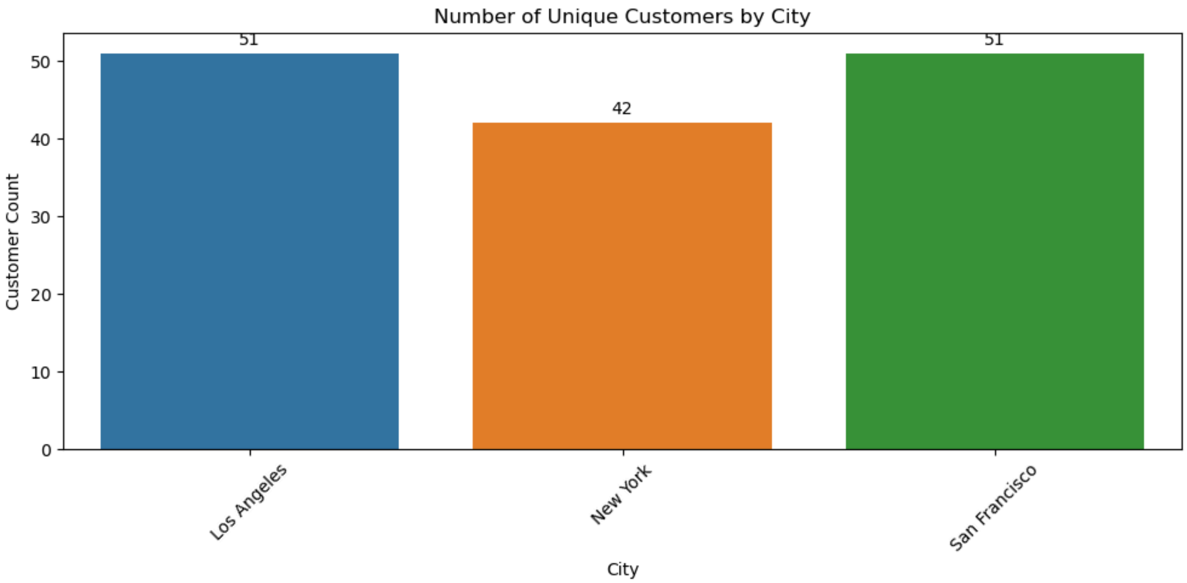
```
Order ID         0
Order Date       0
Customer Name    0
City             0
Product Info     0
Price            0
Quantity         0
Shipping Date    0
Unit Price       0
Days to Ship     0
dtype: int64
```

Visualizing the Data

No of customers by city

In [40]:

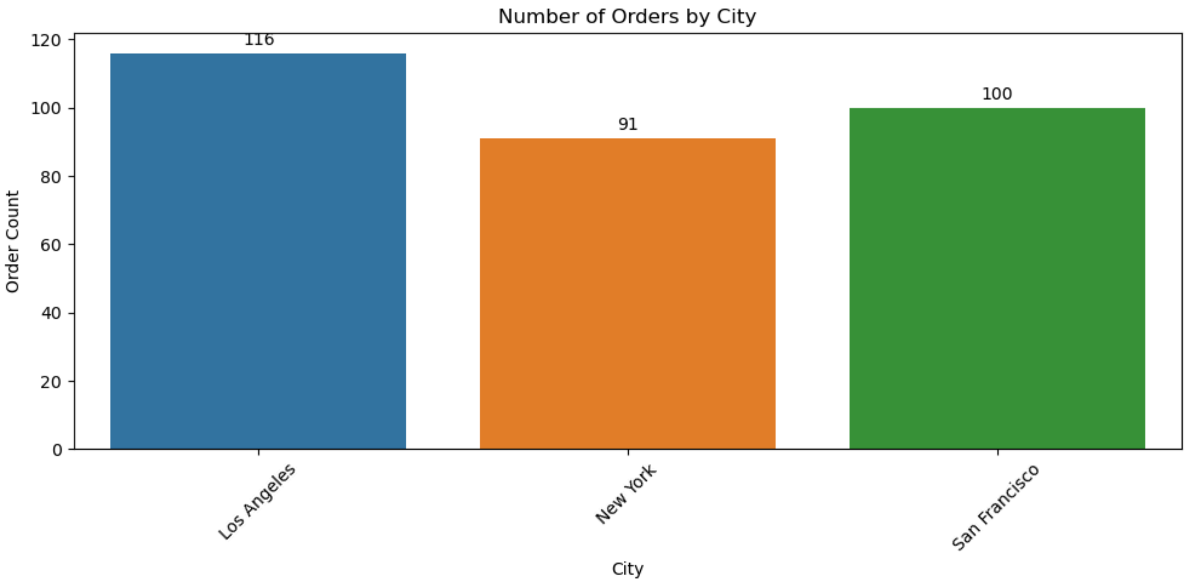
```
customers_per_city = df2.groupby('City')['Customer Name'].nunique().reset_inde
x()
customers_per_city.columns = ['City', 'Customer Count']

# Step 2: Plot using Seaborn
plt.figure(figsize=(10, 5))
ax = sns.barplot(data=customers_per_city, x='City', y='Customer Count')

# Step 3: Add Seaborn-styled labels (still using matplotlib for annotation)
for bar in ax.patches:
    ax.annotate(
        f'{int(bar.get_height())}',
        (bar.get_x() + bar.get_width() / 2, bar.get_height()),
        ha='center', va='bottom',
        fontsize=10, color='black',
        xytext=(0, 3),
        textcoords='offset points'
    )

plt.title("Number of Unique Customers by City")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



No of Orders by City

In [41]:

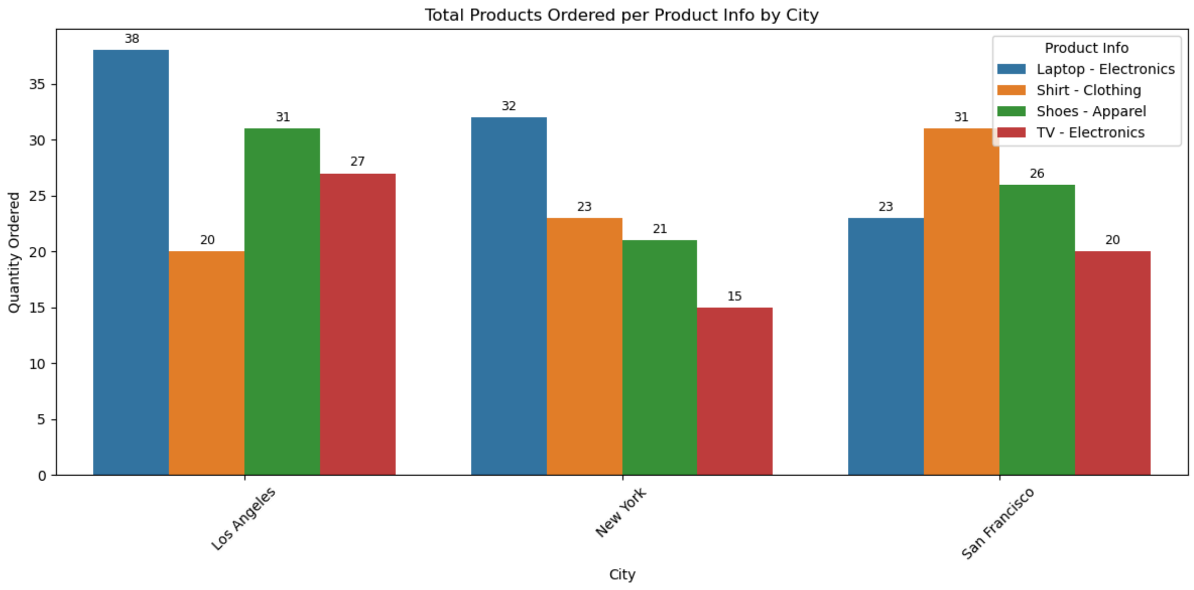
```
orders_per_city = df2.groupby('City')['Quantity'].sum().reset_index()
orders_per_city.columns = ['City', 'Order Count']

# Step 2: Seaborn bar plot
plt.figure(figsize=(10, 5))
ax = sns.barplot(data=orders_per_city, x='City', y='Order Count')

# Step 3: Annotate bar heights
for bar in ax.patches:
    ax.annotate(
        f'{int(bar.get_height())}',
        (bar.get_x() + bar.get_width() / 2, bar.get_height()),
        ha='center', va='bottom',
        fontsize=10, color='black',
        xytext=(0, 3),
        textcoords='offset points'
    )

plt.title("Number of Orders by City")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



No of Product ordered by city

In [42]:

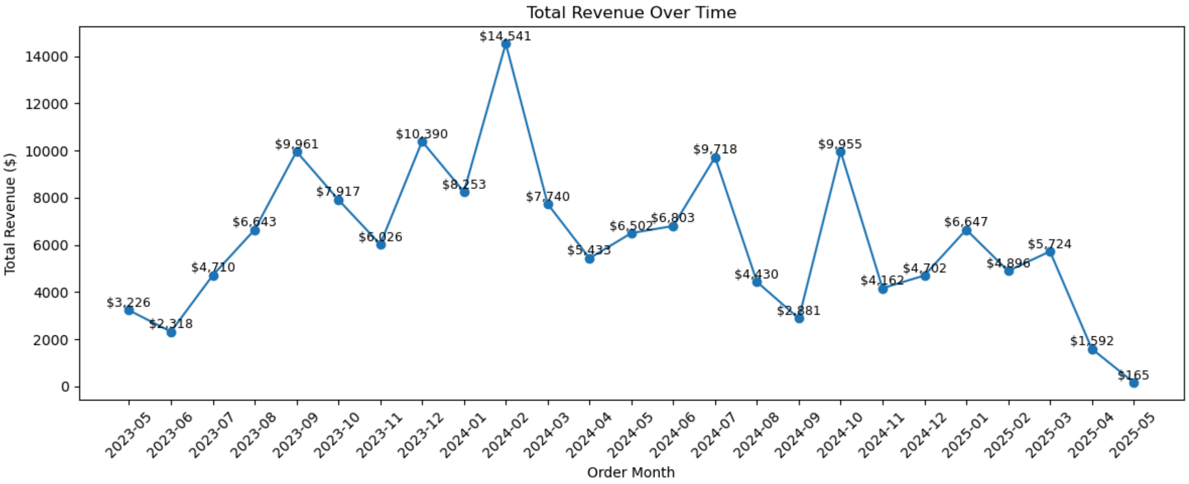
```python
# Step 1: Group and sum Quantity
product_orders = df2.groupby(['City', 'Product Info'])['Quantity'].sum().reset_index()

# Step 2: Plot with Seaborn
plt.figure(figsize=(12, 6))
ax = sns.barplot(data=product_orders, x='City', y='Quantity', hue='Product Info')

# Step 3: Add labels on top of bars
for bar in ax.patches:
    height = bar.get_height()
    if not pd.isna(height) and height > 0:
        ax.annotate(
            f'{int(height)}',
            (bar.get_x() + bar.get_width() / 2, height),
            ha='center', va='bottom',
            fontsize=9,
            xytext=(0, 3),
            textcoords='offset points'
        )

plt.title("Total Products Ordered per Product Info by City")
plt.ylabel("Quantity Ordered")
plt.xticks(rotation=45)
plt.legend(title='Product Info')
plt.tight_layout()
plt.show()
```



Total Revenue Over Time(Year-Month)

In [43]:

```
# Ensure datetime
df2['Order Date'] = pd.to_datetime(df2['Order Date'], errors='coerce')

# Drop missing dates or prices
df_temp = df2.dropna(subset=['Order Date', 'Price']).copy()

# Ensure price is numeric
df_temp['Price'] = pd.to_numeric(df_temp['Price'], errors='coerce')

# Create month column
df_temp['Order Month'] = df_temp['Order Date'].dt.to_period('M').astype(str)

# Group and sort
revenue_over_time = (
    df_temp.groupby('Order Month')['Price']
    .sum()
    .reset_index()
    .sort_values('Order Month')
)

# Convert to NumPy for plotting
x_vals = revenue_over_time['Order Month'].to_numpy()
y_vals = revenue_over_time['Price'].to_numpy()

# Plot
plt.figure(figsize=(12, 5))
plt.plot(x_vals, y_vals, marker='o')

# Add value labels on each point
for i in range(len(x_vals)):
    plt.text(x_vals[i], y_vals[i] + max(y_vals) * 0.01, f"${y_vals[i]:,.0f}",
             ha='center', fontsize=9)

plt.title("Total Revenue Over Time")
plt.xlabel("Order Month")
plt.ylabel("Total Revenue ($)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

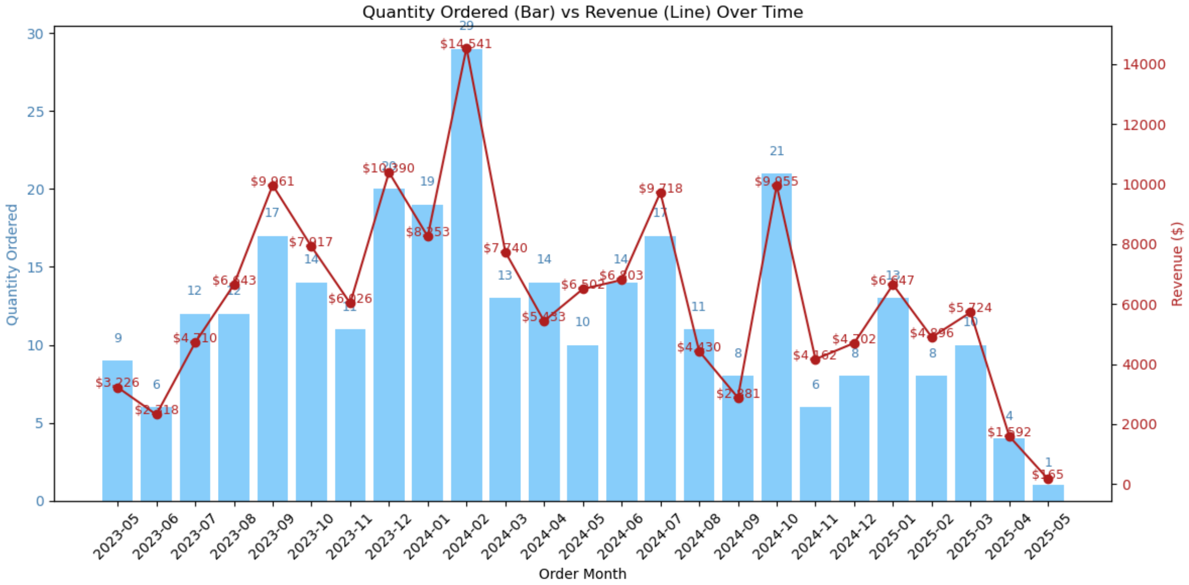


Total Revenue Over Time

Total Quantity Ordered along with revenue

In [58]:

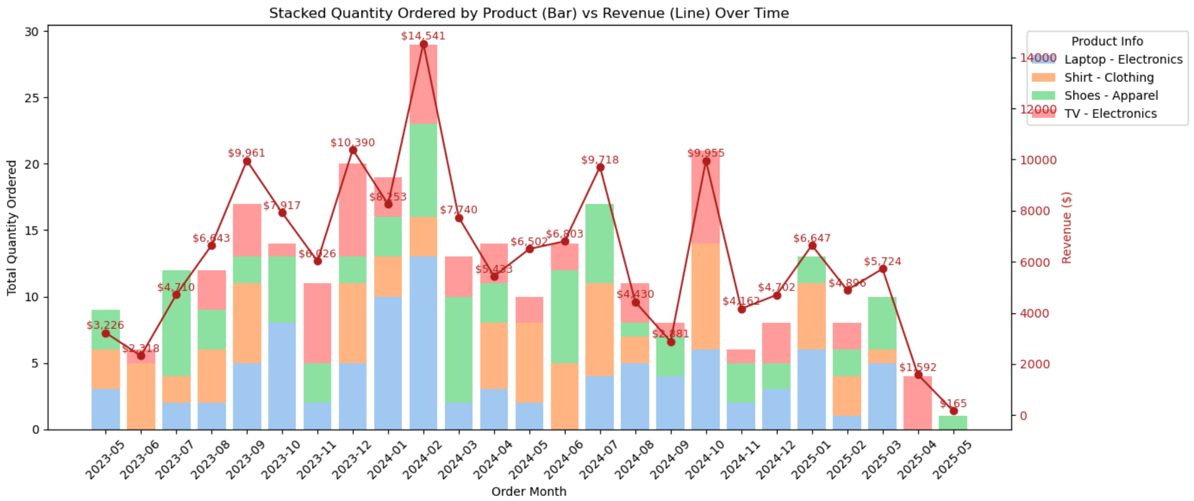
```python
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Clean and prepare data
df2['Order Date'] = pd.to_datetime(df2['Order Date'], errors='coerce')
df2['Price'] = pd.to_numeric(df2['Price'], errors='coerce')
df2 = df2.dropna(subset=['Order Date', 'Price', 'Quantity']).copy()
df2['Order Month'] = df2['Order Date'].dt.to_period('M').astype(str)

# Step 2: Group and sort
summary = df2.groupby('Order Month').agg({
    'Quantity': 'sum',
    'Price': 'sum'
}).reset_index().sort_values('Order Month')

# Convert to NumPy arrays to avoid multi-indexing issues
x_vals = summary['Order Month'].to_numpy()
quantity_vals = summary['Quantity'].to_numpy()
revenue_vals = summary['Price'].to_numpy()

# Step 3: Plot
fig, ax1 = plt.subplots(figsize=(12, 6))

# Bar chart for Quantity Ordered
bars = ax1.bar(x_vals, quantity_vals, color='lightskyblue', label='Quantity Ordered')
ax1.set_ylabel("Quantity Ordered", color='steelblue')
ax1.set_xlabel("Order Month")
ax1.tick_params(axis='y', labelcolor='steelblue')
ax1.tick_params(axis='x', rotation=45)

# Add bar labels
for bar in bars:
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width() / 2, height + 1, f"{int(height)}",
             ha='center', va='bottom', fontsize=9, color='steelblue')

# Line chart for Revenue
ax2 = ax1.twinx()
ax2.plot(x_vals, revenue_vals, color='firebrick', marker='o', label='Revenue')
ax2.set_ylabel("Revenue ($)", color='firebrick')
ax2.tick_params(axis='y', labelcolor='firebrick')

# Add revenue point labels
for i in range(len(x_vals)):
    ax2.text(x_vals[i], revenue_vals[i] + 1, f"${revenue_vals[i]:,.0f}",
             ha='center', fontsize=9, color='firebrick')

# Final formatting
plt.title("Quantity Ordered (Bar) vs Revenue (Line) Over Time")
plt.tight_layout()
plt.show()
```



Total Quantity Ordered Per Product Info and Revenue

In [56]:

```python
# Step 1: Clean data
df2['Order Date'] = pd.to_datetime(df2['Order Date'], errors='coerce')
df2['Price'] = pd.to_numeric(df2['Price'], errors='coerce')
df2 = df2.dropna(subset=['Order Date', 'Price', 'Quantity']).copy()
df2['Order Month'] = df2['Order Date'].dt.to_period('M').astype(str)

# Step 2: Group quantity for stacked bars
quantity_grouped = df2.groupby(['Order Month', 'Product Info'])['Quantity'].sum().unstack(fill_value=0)
order_months = quantity_grouped.index.tolist()

# Step 3: Group revenue for line chart
revenue_grouped = df2.groupby('Order Month')['Price'].sum().reindex(order_months).fillna(0).to_numpy()

# Step 4: Plot
fig, ax1 = plt.subplots(figsize=(14, 6))

# Stacked bar chart
bottom_vals = [0] * len(order_months)
colors = sns.color_palette("pastel", len(quantity_grouped.columns))

for i, product in enumerate(quantity_grouped.columns):
    values = quantity_grouped[product].to_numpy()
    ax1.bar(order_months, values, label=product, bottom=bottom_vals, color=colors[i])
    bottom_vals = [i + j for i, j in zip(bottom_vals, values)]

ax1.set_ylabel("Total Quantity Ordered")
ax1.set_xlabel("Order Month")
ax1.tick_params(axis='x', rotation=45)
ax1.legend(title="Product Info", bbox_to_anchor=(1.01, 1), loc='upper left')

# Twin axis for revenue line
ax2 = ax1.twinx()
ax2.plot(order_months, revenue_grouped, color='firebrick', marker='o', label='Revenue')
ax2.set_ylabel("Revenue ($)", color='firebrick')
ax2.tick_params(axis='y', labelcolor='firebrick')

# Add revenue labels
for x, y in zip(order_months, revenue_grouped):
    ax2.text(x, y + max(revenue_grouped) * 0.01, f"${y:,.0f}", ha='center', fontsize=9, color='firebrick')

plt.title("Stacked Quantity Ordered by Product (Bar) vs Revenue (Line) Over Time")
plt.tight_layout()
plt.show()
```



In [48]:
```
df2.groupby(['City']).agg({
    'Quantity':'sum',
    'Price':'sum'
})
```

Out[48]:

	Quantity	Price
City		
Los Angeles	116.0	49937.170
New York	91.0	53656.015
San Francisco	100.0	51742.395

In [51]:
```
df2.groupby(['City']).agg({
    'Days to Ship':'median'
})
```

Out[51]:

	Days to Ship
City	
Los Angeles	5.0
New York	6.0
San Francisco	6.0

In [54]:

```
df2.groupby(['Product Info']).agg({
    'Unit Price':'median'
}).sort_values(by='Unit Price',ascending=False)
```

Out[54]:

	Unit Price
Product Info	
Laptop - Electronics	644.085000
TV - Electronics	536.770000
Shirt - Clothing	491.703333
Shoes - Apparel	462.370000

Business Insights

- Cities like **New York** and **San Francisco** drive the most volume and revenue
- **Laptops** and **TVs** have the highest average unit price
- Some cities show longer Days to Ship; these could be operational inefficiencies

In [ ]:

```

```