Dataset EDA

product_sales DataFrame as `df`

```
-- Explore the data in the table (SQL)
SELECT *
FROM 'product_sales.csv';
```

	week	sales_method	customer_id	nb_sold	revenue	years_as_customer	nb_site_visits	state
80	5	Email	eb9cfaf4-7f4e-4b76-8028-bec21f5bb3f9	10	104.46	14	18	Maryland
81	2	Call	7f2244c1-bf0b-4afc-8b43-206a087633d9	10	48.98	1	27	California
82	6	Email + Call	9a39d6f6-ef55-4812-ad9d-0016d6285ff7	15	229.36	3	29	Georgia
83	2	Email + Call	adc06513-5b96-4464-800c-e4accef0f171	10	153.87	5	32	Oregon
84	4	Call	0d7104e0-a930-4100-9b51-3746a3449403	11	52.54	3	23	Texas
85	4	Email	60a2b518-0f48-4643-b039-14a813269609	11	113.84	1	24	California
86	4	Call	c11f3e76-4f1a-4184-994c-4a43efd8f76f	10	51.35	17	27	California
87	5	Email	f0078065-6827-4be2-ac96-827312e5d7f8	11	105.44	4	22	Illinois
88	5	Call	51b46c10-bf32-4e16-bdb2-19546d28ef70	10	52.44	10	29	Arkansas
89	1	Call	a1651d36-f88a-4535-882b-40f132bd72a0	7	33.63	3	16	Connecticut
90	1	Email	f1a7d98f-10ee-4850-b75c-79ed68eca614	8	82.94	14	19	North Carolina
91	1	Email	44c29720-c815-4bff-bcf6-8182912c3c74	8	83.71	4	29	Washington
92	2	Email	6ffdca0f-8054-4d1f-99cb-5b510a1e57cf	10	100.38	2	28	New Jersey
93	5	Email + Call	cf11385c-a004-45b2-ba8f-3003c32184d5	12	186.27	3	22	Michigan
94	4	Email	dbc09910-d723-4e02-9a63-945c137c4080	11	107.1	3	22	Tennessee
95	1	Email	172fcd47-62ac-4976-bac7-6934464a8c5c	8	81.98	0	23	Arizona

12,500 rows truncated from 15,000 rows

```
# Display basic information about the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   week               15000 non-null  int64 
 1   sales_method       15000 non-null  object
 2   customer_id        15000 non-null  object
 3   nb_sold            15000 non-null  int64 
 4   revenue            15000 non-null  object
 5   years_as_customer  15000 non-null  int64 
 6   nb_site_visits     15000 non-null  int64 
 7   state              15000 non-null  object
dtypes: int64(4), object(4)
memory usage: 937.6+ KB
```

```
# Display the number of rows in the dataframe
df.shape[0]
```

```
15000
```

```
# Fill NA values in the 'revenue' column with 0 and display the first few rows
# df['revenue'] = df['revenue'].fillna(0)
df['revenue'].head()
```

	revenue
0	NA
1	225.47
2	52.55
3	NA
4	90.49

5 rows

```
import pandas as pd

# Check for missing values
print(df.isna().sum())
```

```
week                0
sales_method        0
customer_id         0
nb_sold             0
revenue             0
years_as_customer   0
nb_site_visits      0
state               0
dtype: int64
```

```
df['revenue'] = pd.to_numeric(df['revenue'], errors='coerce')
df['revenue'].head()
```

	revenue
0	null
1	225.47
2	52.55
3	null
4	90.49

5 rows

```
na_count = df[df['revenue'] == 'NA'].shape[0]
print(na_count)
```

```
0
```

```
# Calculate the total revenue
total_revenue = df['revenue'].sum()
total_revenue
```

```
1308138.01
```

```
df['revenue'].head()
```

	revenue
0	null
1	225.47
2	52.55
3	null
4	90.49

5 rows

```
# Display all distinct categories in the 'sales_method' column
unique_sales_methods = df['sales_method'].unique()
print(unique_sales_methods)
```

```
['Email' 'Email + Call' 'Call' 'em + call' 'email']
```

```
sales_method_mapping = {
    'Email': 'Email',
    'email': 'Email',          # Normalize different cases
    'Email + Call': 'Email + Call',
    'Call': 'Call',
```

```
    'em + call': 'Email + Call' # Merge 'em + call' into 'Email + Call'
}

# Replace categories in the 'sales_method' column based on the mapping
df['sales_method'] = df['sales_method'].replace(sales_method_mapping)

# Display all distinct categories in the 'sales_method' column
unique_sales_methods = df['sales_method'].unique()
print(unique_sales_methods)
```

```
['Email' 'Email + Call' 'Call']
```

```
df['sales_method'].count()
```

```
15000
```

```
# Count occurrences of each category
sales_method_counts = df['sales_method'].value_counts()

# Display the counts
print(sales_method_counts)
```

```
sales_method
Email           7466
Call            4962
Email + Call    2572
Name: count, dtype: int64
```

```
unique_weeks = df['week'].unique()
print(unique_weeks)
```

```
[2 6 5 4 3 1]
```

```
weeks_count = df['week'].value_counts()
print(weeks_count)
```

```
week
1    3721
4    2575
5    2574
2    2491
3    2411
6    1228
Name: count, dtype: int64
```

```
# Calculate the total weeks counts
total_count = weeks_count.sum()
total_count
```

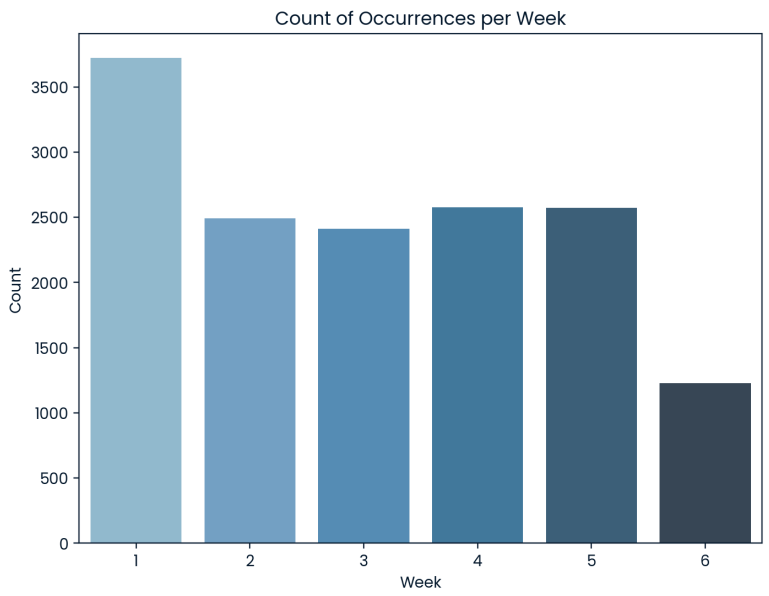
```
15000
```

```
# Calculate the percentage distribution
percentage_distribution = (weeks_count / total_count) * 100

# Print the original counts and the percentage distribution
print("Weeks Count:")
print(weeks_count)

print("\nPercentage Distribution:")
print(percentage_distribution)
```

```
Weeks Count:
week
1    3721
4    2575
5    2574
2    2491
3    2411
6    1228
Name: count, dtype: int64

Percentage Distribution:
week
1    24.806667
4    17.166667
5    17.160000
2    16.606667
3    16.073333
6     8.186667
Name: count, dtype: float64
```

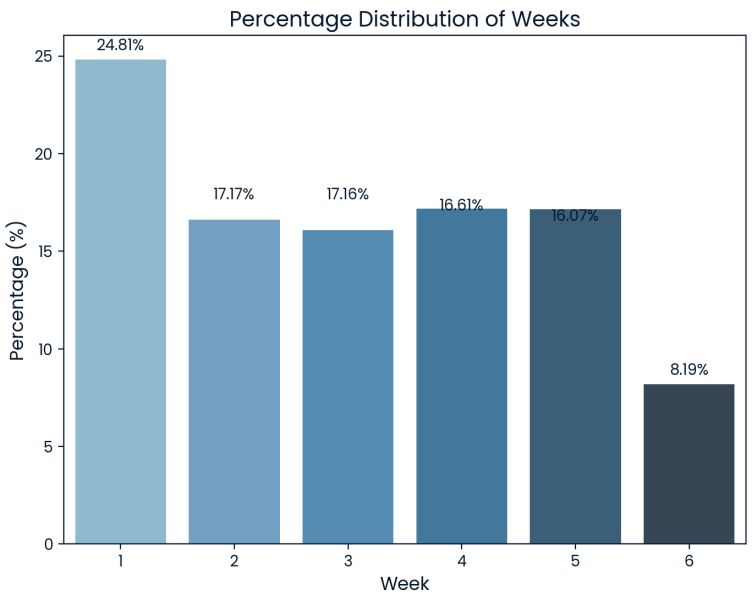
```
import matplotlib.pyplot as plt
import seaborn as sns

# Data (manually input based on your provided weeks_count)
weeks_count = {
    1: 3721,
    2: 2491,
    3: 2411,
    4: 2575,
    5: 2574,
    6: 1228
}

# Convert dictionary to two lists: weeks and counts
weeks = list(weeks_count.keys())
counts = list(weeks_count.values())

# Create the bar plot
plt.figure(figsize=(8, 6))  # Set figure size
sns.barplot(x=weeks, y=counts, palette='Blues_d')  # Create the barplot

# Add titles and labels
plt.title('Count of Occurrences per Week')
plt.xlabel('Week')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



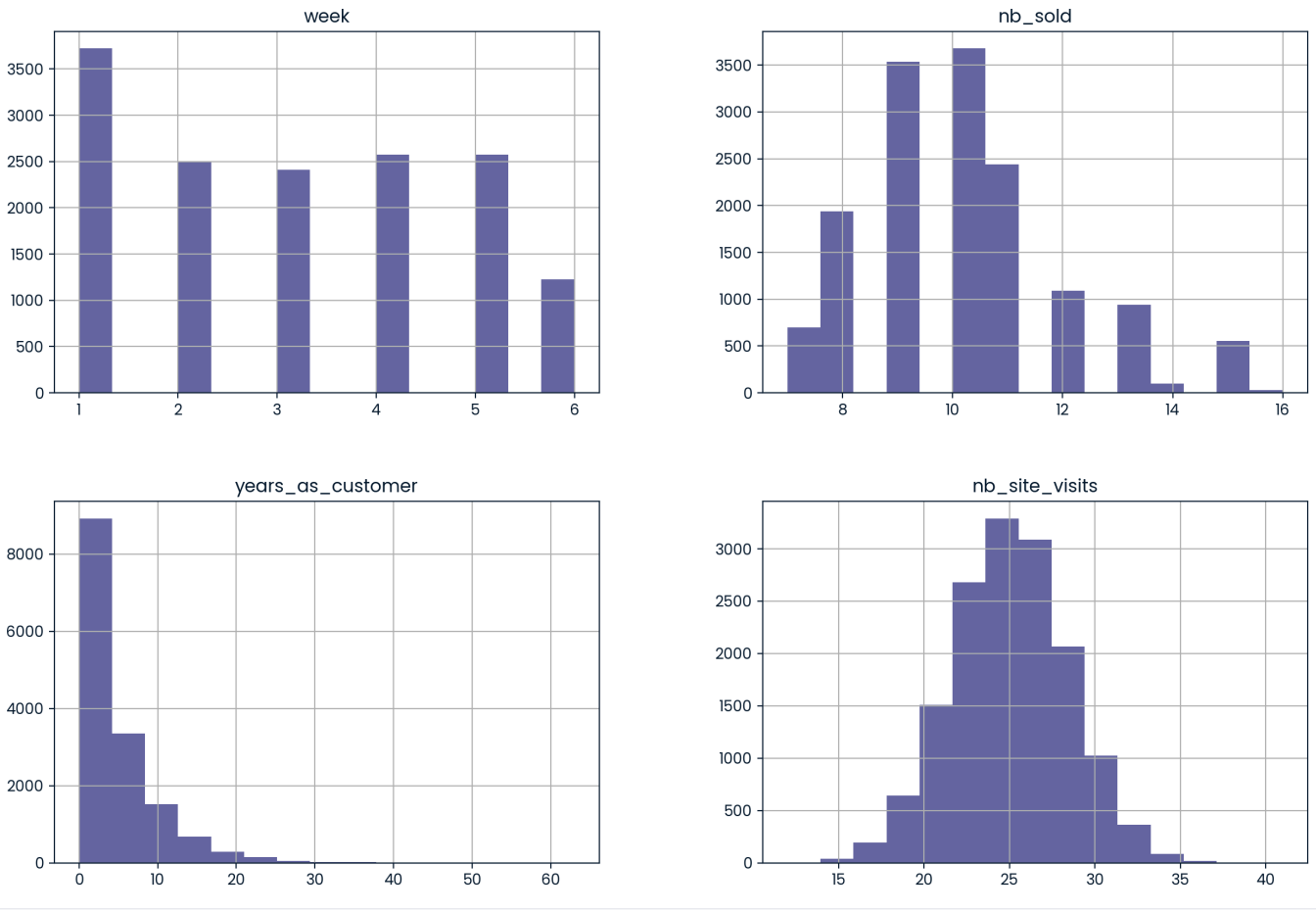

```
import matplotlib.pyplot as plt
import seaborn as sns

# Create the bar plot for percentage distribution
```

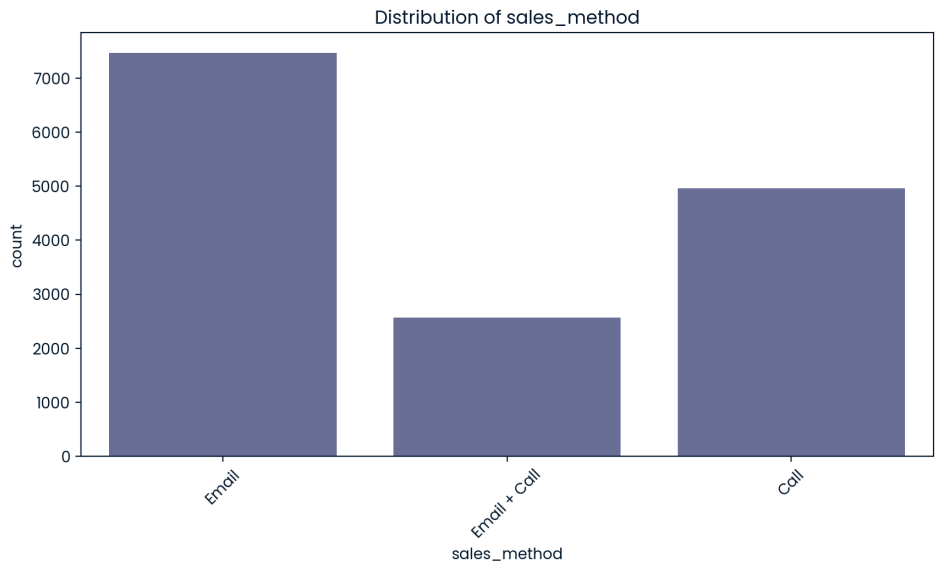
```
plt.figure(figsize=(8, 6))  # Set the figure size
sns.barplot(x=percentage_distribution.index, y=percentage_distribution.values, palette='Blues_d')

# Add titles and labels
plt.title('Percentage Distribution of Weeks', fontsize=14)
plt.xlabel('Week', fontsize=12)
plt.ylabel('Percentage (%)', fontsize=12)

# Show the percentage on top of the bars
for i, value in enumerate(percentage_distribution.values):
    plt.text(i, value + 0.5, f'{value:.2f}%', ha='center', fontsize=10)

# Show the plot
plt.show()
```

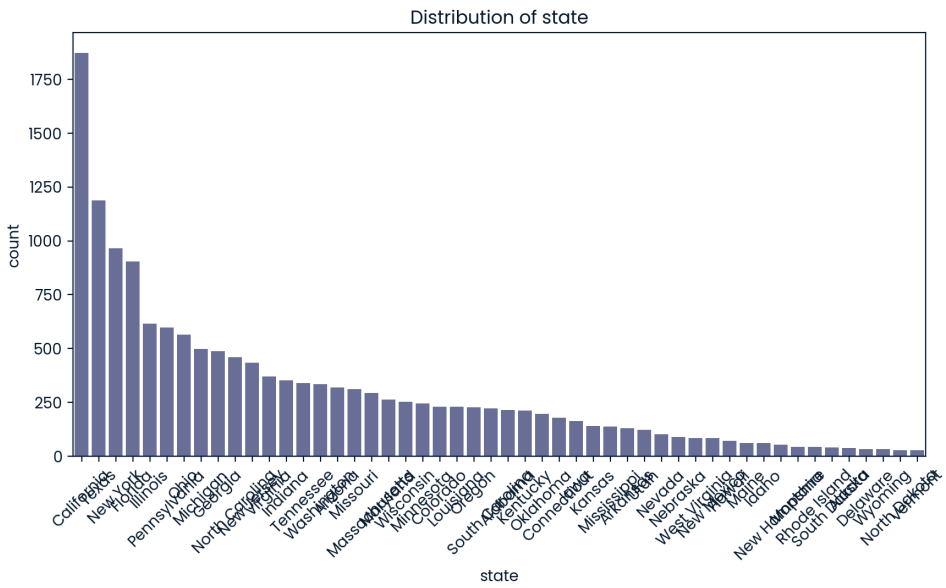


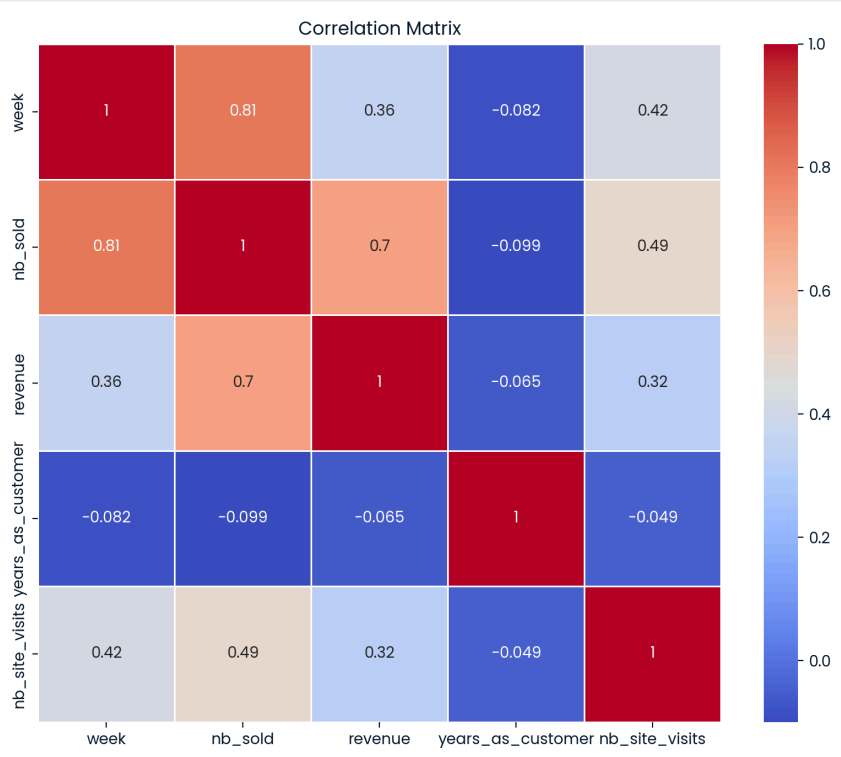

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming df is already defined and loaded with data
# Display basic information about the dataframe
df.info()

# Display basic statistics of the dataframe
df.describe(include='all')

# Check for missing values
missing_values = df.isnull().sum()

# Display the first few rows of the dataframe
df.head()

# Plot the distribution of numerical columns
numerical_columns = ['week', 'nb_sold', 'years_as_customer', 'nb_site_visits']
df[numerical_columns].hist(bins=15, figsize=(15, 10), layout=(2, 2))

# Plot the distribution of categorical columns
categorical_columns = ['sales_method', 'state']
for column in categorical_columns:
    plt.figure(figsize=(10, 5))
    # Sort the state column based on the count (for 'state' column specifically)
    if column == 'state':
        order = df['state'].value_counts().index  # Sort states by count in descending order
        sns.countplot(data=df, x=column, order=order)
    else:
        sns.countplot(data=df, x=column)

    plt.title(f'Distribution of {column}')
    plt.xticks(rotation=45)
    plt.show()

# Plot the correlation matrix
plt.figure(figsize=(10, 8))
# Select only numerical columns for correlation matrix
numerical_df = df.select_dtypes(include=['number'])
correlation_matrix = numerical_df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   week               15000 non-null  int64
 1   sales_method       15000 non-null  object
 2   customer_id        15000 non-null  object
 3   nb_sold            15000 non-null  int64
 4   revenue            13926 non-null  float64
 5   years_as_customer  15000 non-null  int64
 6   nb_site_visits     15000 non-null  int64
 7   state              15000 non-null  object
dtypes: float64(1), int64(4), object(3)
memory usage: 937.6+ KB
```










```
# Get the number of rows in the dataframe
num_rows = df.shape[0]
num_rows
```

```
15000
```

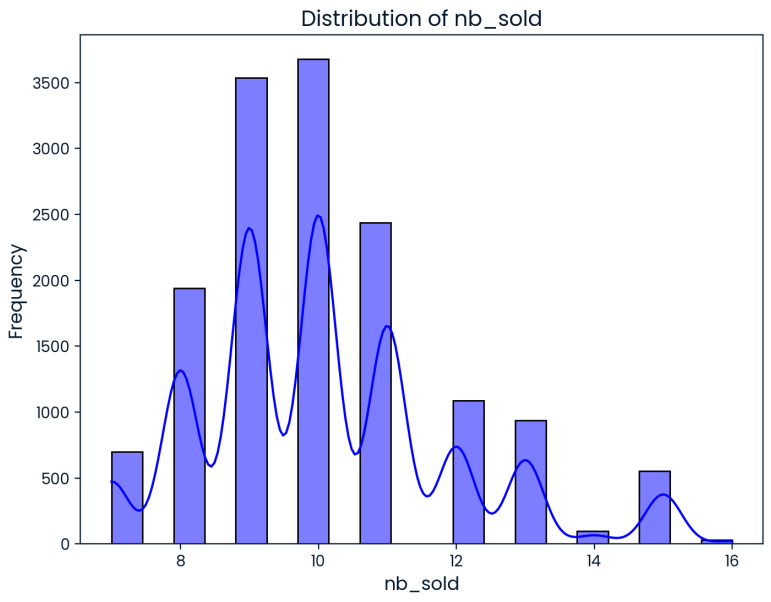
```
nb_sold_summary = df['nb_sold'].describe()
print(nb_sold_summary)
```

```
count    15000.000000
mean        10.084667
std          1.812213
min          7.000000
25%          9.000000
50%         10.000000
75%         11.000000
max         16.000000
Name: nb_sold, dtype: float64
```

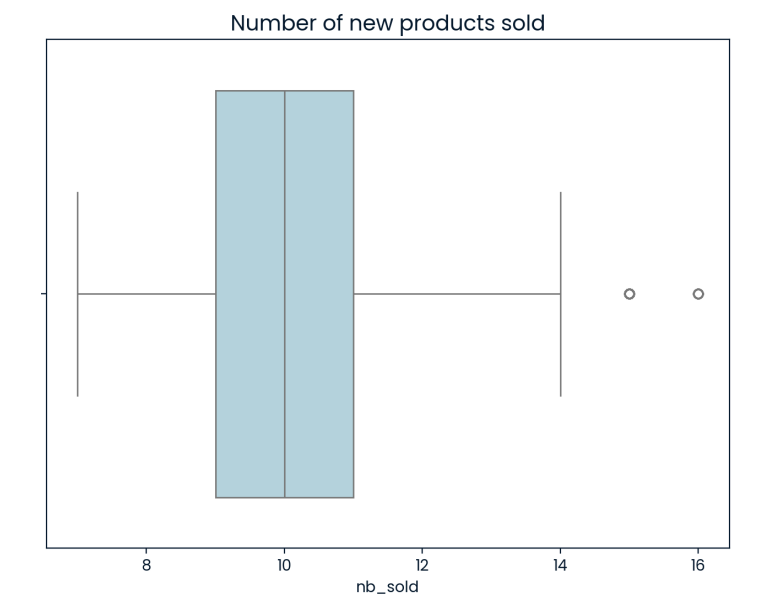
```
# Check for missing values
missing_nb_sold = df['nb_sold'].isnull().sum()
print(f"Missing values in nb_sold: {missing_nb_sold}")
```

```
Missing values in nb_sold: 0
```

```
# Plot the distribution of nb_sold using a histogram
plt.figure(figsize=(8, 6))
sns.histplot(df['nb_sold'], bins=20, kde=True, color='blue')

# Add titles and labels
plt.title('Distribution of nb_sold', fontsize=14)
plt.xlabel('nb_sold', fontsize=12)
plt.ylabel('Frequency', fontsize=12)

# Show the plot
plt.show()
```




```
# Create a box plot for nb_sold
plt.figure(figsize=(8, 6))
sns.boxplot(x=df['nb_sold'], color='lightblue')

# Add a title
plt.title('Number of new products sold', fontsize=14)

# Show the plot
plt.show()
```




```
# Select only numeric columns from the DataFrame
numeric_df = df.select_dtypes(include='number')

# Calculate correlation between nb_sold and other numerical columns
if 'nb_sold' in numeric_df.columns:
    correlations = numeric_df.corr()['nb_sold'].sort_values(ascending=False)
    print(correlations)
else:
    print("The 'nb_sold' column is not found in the numeric columns.")
```

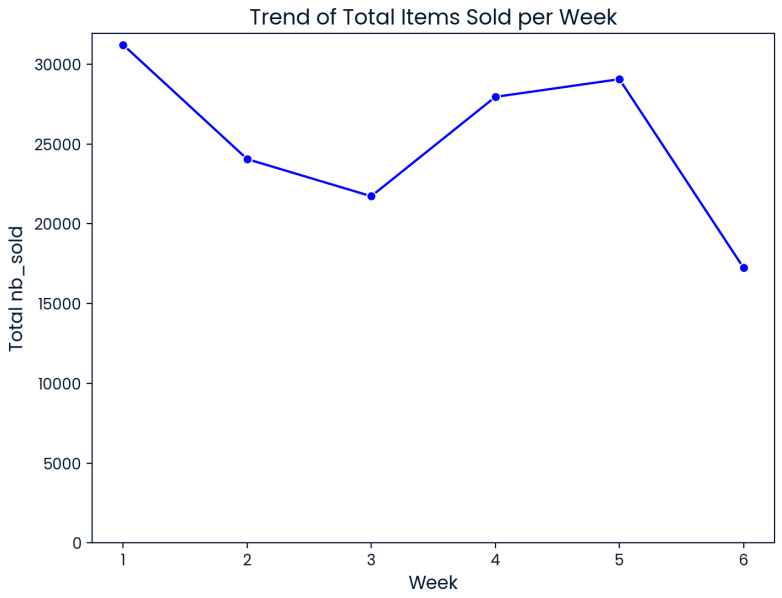
```
nb_sold              1.000000
week                 0.809887
revenue              0.696165
nb_site_visits       0.490718
years_as_customer   -0.099117
Name: nb_sold, dtype: float64
```

```
# Example: Group by week and calculate the sum of nb_sold for each week
nb_sold_by_week = df.groupby('week')['nb_sold'].sum()

print(nb_sold_by_week)
```

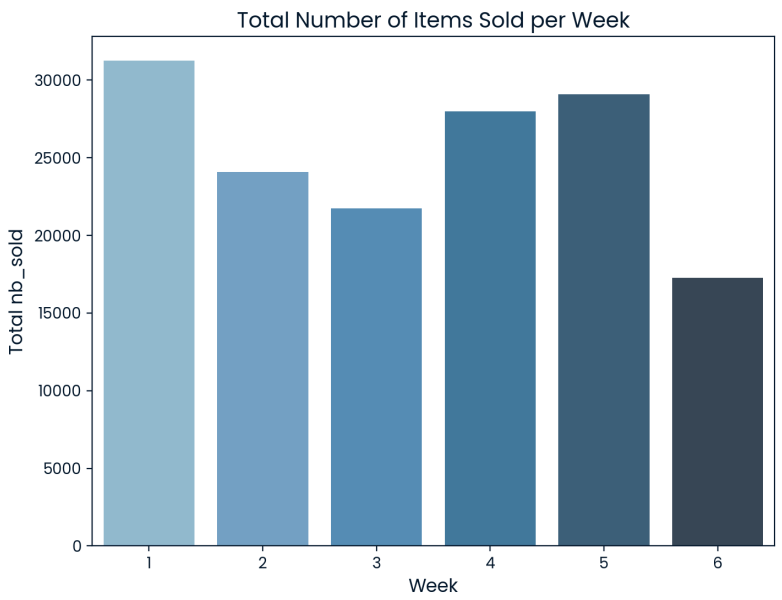
```
week
1    31220
2    24056
3    21728
4    27955
5    29063
6    17248
Name: nb_sold, dtype: int64
```

```
# Create a line plot for the sum of nb_sold by week
plt.figure(figsize=(8, 6))
sns.lineplot(x=nb_sold_by_week.index, y=nb_sold_by_week.values, marker='o', color='blue')

# Add titles and labels
plt.title('Trend of Total Items Sold per Week', fontsize=14)
plt.xlabel('Week', fontsize=12)
plt.ylabel('Total nb_sold', fontsize=12)

# Set the y-axis limit to start from 0
plt.ylim(0)

# Show the plot
```

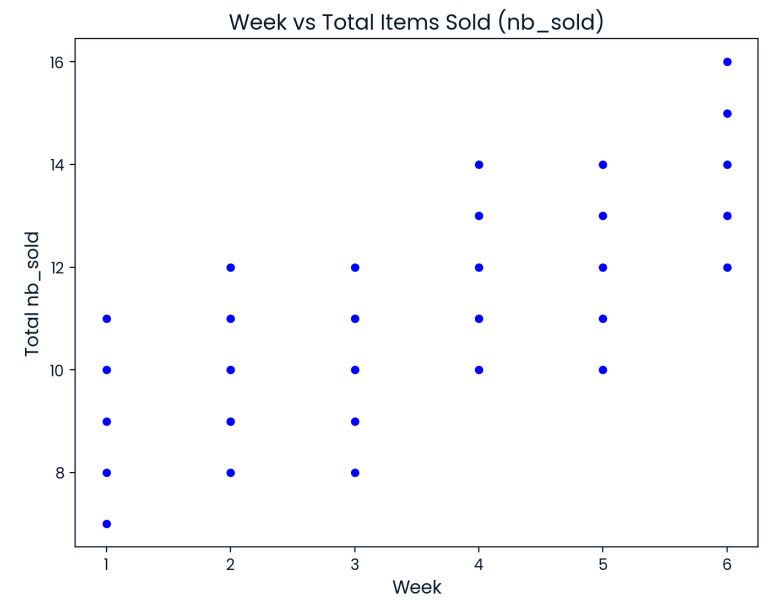
```
plt.show()
```




```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming nb_sold_by_week has been computed as follows:
# nb_sold_by_week = df.groupby('week')['nb_sold'].sum()

# Create a bar plot for the sum of nb_sold by week
plt.figure(figsize=(8, 6))
sns.barplot(x=nb_sold_by_week.index, y=nb_sold_by_week.values, palette='Blues_d')

# Add titles and labels
plt.title('Total Number of Items Sold per Week', fontsize=14)
plt.xlabel('Week', fontsize=12)
plt.ylabel('Total nb_sold', fontsize=12)

# Show the plot
plt.show()
```

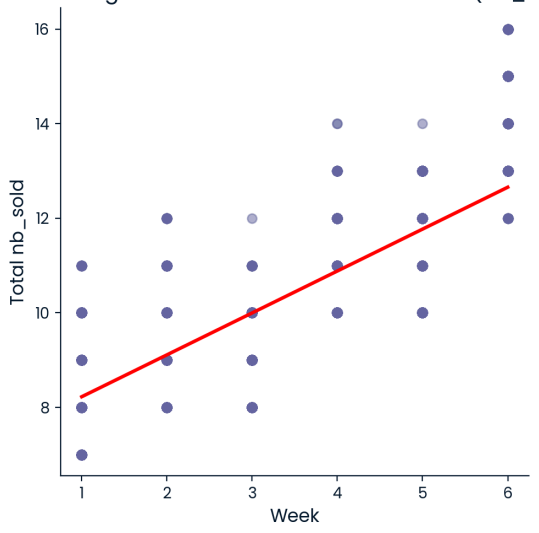



```
import matplotlib.pyplot as plt
import seaborn as sns

# Scatter plot for week vs nb_sold
plt.figure(figsize=(8, 6))
sns.scatterplot(x=df['week'], y=df['nb_sold'], color='blue')

# Add titles and labels
plt.title('Week vs Total Items Sold (nb_sold)', fontsize=14)
plt.xlabel('Week', fontsize=12)
plt.ylabel('Total nb_sold', fontsize=12)

# Show the plot
plt.show()

# Scatter plot for nb_site_visits vs nb_sold
plt.figure(figsize=(8, 6))
sns.scatterplot(x=df['nb_site_visits'], y=df['nb_sold'], color='green')

# Add titles and labels
plt.title('Site Visits vs Total Items Sold (nb_sold)', fontsize=14)
plt.xlabel('Number of Site Visits', fontsize=12)
plt.ylabel('Total nb_sold', fontsize=12)

# Show the plot
plt.show()
```



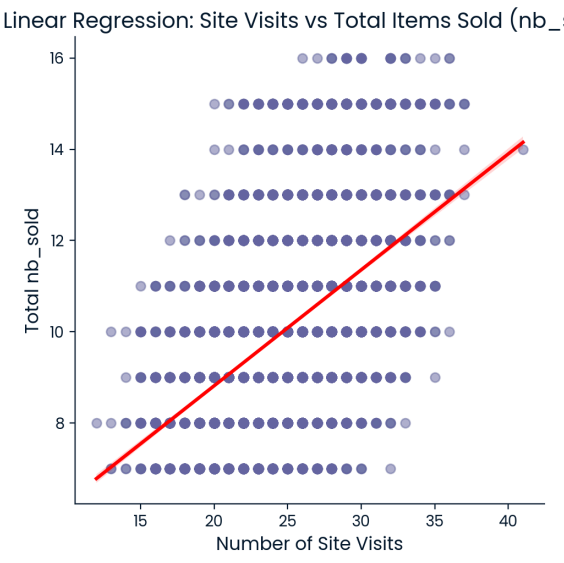

```
import seaborn as sns
import matplotlib.pyplot as plt

# Linear regression plot for week vs nb_sold
plt.figure(figsize=(8, 6))
sns.lmplot(x='week', y='nb_sold', data=df, line_kws={'color': 'red'}, scatter_kws={'alpha':0.5})

# Add titles and labels
plt.title('Linear Regression: Week vs Total Items Sold (nb_sold)', fontsize=14)
plt.xlabel('Week', fontsize=12)
plt.ylabel('Total nb_sold', fontsize=12)

plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```




```
# Linear regression plot for nb_site_visits vs nb_sold
plt.figure(figsize=(8, 6))
sns.lmplot(x='nb_site_visits', y='nb_sold', data=df, line_kws={'color': 'red'}, scatter_kws={'alpha':0.5})

# Add titles and labels
plt.title('Linear Regression: Site Visits vs Total Items Sold (nb_sold)', fontsize=14)
plt.xlabel('Number of Site Visits', fontsize=12)
plt.ylabel('Total nb_sold', fontsize=12)

plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```




```
from sklearn.linear_model import LinearRegression
import numpy as np

# Prepare data
X = df[['week']]  # Independent variable
y = df['nb_sold']  # Dependent variable

# Create and fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Print the coefficients and intercept
print(f"Intercept: {model.intercept_}")
print(f"Coefficient (slope): {model.coef_[0]}")

# R-squared value to evaluate the model fit
r_squared = model.score(X, y)
print(f"R-squared: {r_squared}")
```

```
Intercept: 7.339413727962314
Coefficient (slope): 0.8860608959970155
R-squared: 0.6559176669278741
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
import numpy as np

# Prepare the data
X = df[['week']]  # Independent variable (reshape as 2D array)
y = df['nb_sold']  # Dependent variable
```

```
# Create and fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict values based on the model
y_pred = model.predict(X)

# Calculate regression coefficients
intercept = model.intercept_
slope = model.coef_[0]

# Calculate R-squared value
r_squared = model.score(X, y)

# Create the scatter plot and regression line
plt.figure(figsize=(8,6))
plt.scatter(df['week'], df['nb_sold'], color='blue', alpha=0.5, label='Actual Data')  # Scatter plot
plt.plot(df['week'], y_pred, color='red', label=f'Regression Line\n$y={intercept:.2f} + {slope:.2f}x$')  # Regression line

# Add titles and labels
plt.title(f'Linear Regression: Week vs Total Items Sold (nb_sold)\n$R^2={r_squared:.2f}$', fontsize=14)
plt.xlabel('Week', fontsize=12)
plt.ylabel('Total nb_sold', fontsize=12)

# Add legend
plt.legend()

# Show the plot
plt.show()
```


```
state_sales = df.groupby('state')['nb_sold'].sum().reset_index()
state_sales = state_sales.sort_values(by='nb_sold', ascending=False)
print(state_sales)

# Sort values to get the top 5 and bottom 5
# top_5_states = state_sales.sort_values(by='nb_sold', ascending=False).head(5)
# bottom_5_states = state_sales.sort_values(by='nb_sold', ascending=True).head(5)

# Display the results
# print("Top 5 States by Number of Items Sold:")
#print(top_5_states)

# print("\nBottom 5 States by Number of Items Sold:")
# print(bottom_5_states)
```

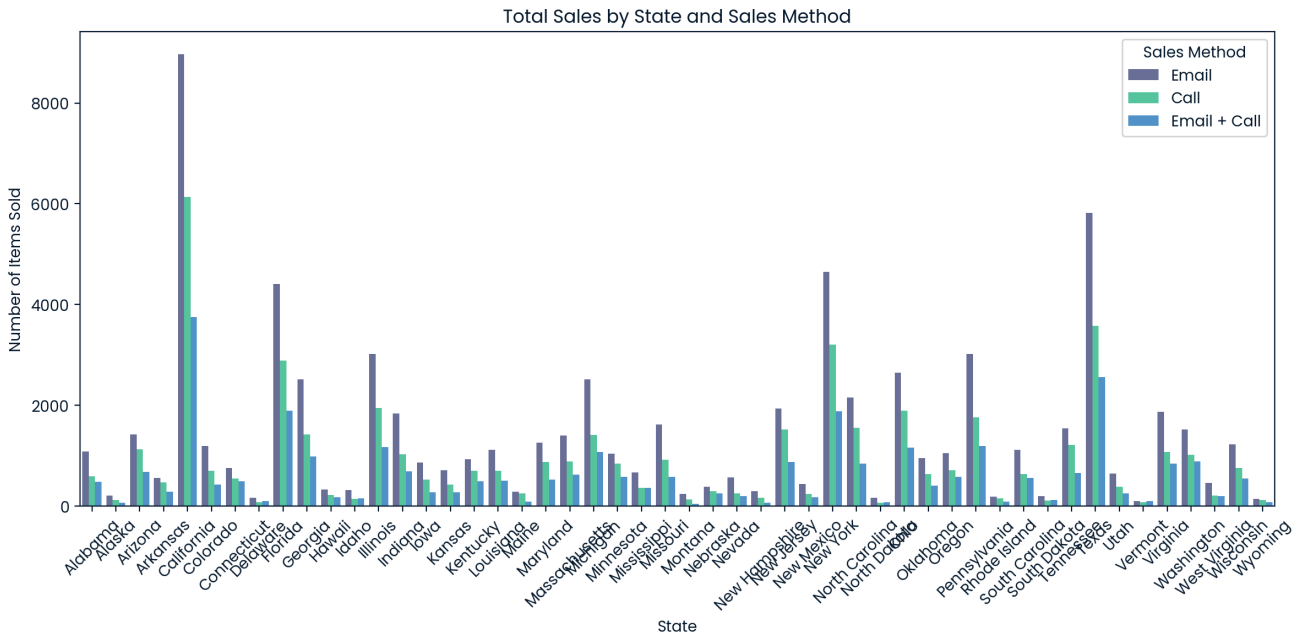
```
            state  nb_sold
4      California    18859
42          Texas    11957
31       New York     9734
8         Florida     9201
12        Illinois     6143
37   Pennsylvania     5979
34           Ohio     5699
21       Michigan     4998
9         Georgia     4930
32  North Carolina     4559
29     New Jersey     4338
45       Virginia     3790
13        Indiana     3558
46     Washington     3424
41      Tennessee     3414
2         Arizona     3238
24       Missouri     3122
20  Massachusetts     2913
19       Maryland     2669
48      Wisconsin     2528
22      Minnesota     2475
36         Oregon     2347
17      Louisiana     2325
5         Colorado     2322
39  South Carolina     2313
0          Alabama     2161
16        Kentucky     2131
35        Oklahoma     1998
```

```
# Group by 'state' and 'sales_method', summing 'nb_sold'
sales_by_state_method = df.groupby(['state', 'sales_method'])['nb_sold'].sum().reset_index()

# Sort by 'state' and 'nb_sold' for better readability
sales_by_state_method = sales_by_state_method.sort_values(by=['state', 'nb_sold'], ascending=[True, False])

# Display the results
print("Total Sales by State and Sales Method:")
print(sales_by_state_method)
```

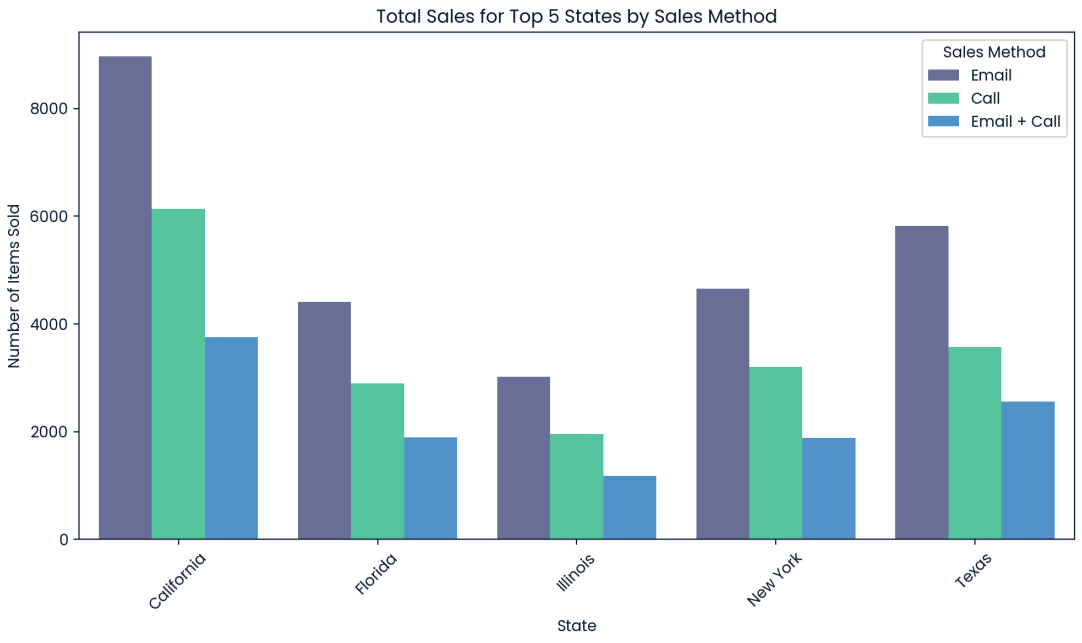
```
Total Sales by State and Sales Method:
         state  sales_method  nb_sold
1      Alabama         Email     1084
0      Alabama          Call      591
2      Alabama  Email + Call      486
4       Alaska         Email      211
3       Alaska          Call      128
..         ...           ...      ...
144  Wisconsin          Call      759
146  Wisconsin  Email + Call      545
148    Wyoming         Email      142
147    Wyoming          Call      119
149    Wyoming  Email + Call       79

[150 rows x 3 columns]
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Create a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(data=sales_by_state_method, x='state', y='nb_sold', hue='sales_method')

# Adding titles and labels
plt.title('Total Sales by State and Sales Method')
plt.xlabel('State')
plt.ylabel('Number of Items Sold')
```

```
plt.legend(title='Sales Method')
plt.xticks(rotation=45)
plt.tight_layout()

# Show the plot
plt.show()
```




```
# Group by 'state' and 'sales_method', summing 'nb_sold'
sales_by_state_method = df.groupby(['state', 'sales_method'])['nb_sold'].sum().reset_index()

# Filter for the top five states
top_states = ['California', 'Texas', 'New York', 'Florida', 'Illinois']
sales_by_state_method = sales_by_state_method[sales_by_state_method['state'].isin(top_states)]

# Sort by 'state' and 'nb_sold' for better readability
sales_by_state_method = sales_by_state_method.sort_values(by=['state', 'nb_sold'], ascending=[True, False])

# Create a bar plot
plt.figure(figsize=(10, 6))
sns.barplot(data=sales_by_state_method, x='state', y='nb_sold', hue='sales_method')

# Adding titles and labels
plt.title('Total Sales for Top 5 States by Sales Method')
plt.xlabel('State')
plt.ylabel('Number of Items Sold')
plt.legend(title='Sales Method')
plt.xticks(rotation=45)
plt.tight_layout()

# Show the plot
plt.show()
```




```
# Group by 'week' and 'sales_method', summing 'nb_sold'
weekly_sales = df.groupby(['week', 'sales_method'])['nb_sold'].sum().reset_index()

# Create a bar plot for visualization
plt.figure(figsize=(14, 7))
sns.barplot(data=weekly_sales, x='week', y='nb_sold', hue='sales_method')

# Adding titles and labels
plt.title('Weekly Sales by Sales Method')
plt.xlabel('Week')
plt.ylabel('Number of Items Sold')
plt.legend(title='Sales Method')
plt.xticks(rotation=45)
plt.tight_layout()

# Show the plot
plt.show()
```