Sales rep requests

product_sales DataFrame as `df`

```
-- Loading the data in the table as df
SELECT *
FROM 'product_sales.csv'
```

	week	sales_method	customer_id	nb_sold
0	2	Email	2e72d641-95ac-497b-bbf8-4861764a7097	10
1	6	Email + Call	3998a98d-70f5-44f7-942e-789bb8ad2fe7	15
2	5	Call	d1de9884-8059-4065-b10f-86eef57e4a44	11
3	4	Email	78aa75a4-ffeb-4817-b1d0-2f030783c5d7	11
4	3	Email	10e6d446-10a5-42e5-8210-1b5438f70922	9
5	6	Call	6489e678-40f2-4fed-a48e-d0dff9c09205	13
6	4	Email	eb6bd5f1-f115-4e4b-80a6-5e67fcfbfb94	11
7	1	Email	047df079-071b-4380-9012-2bfe9bce45d5	10
8	5	Email	771586bd-7b64-40be-87df-afe884d2af9e	11
9	5	Call	56491dae-bbe7-49f0-a651-b823a01103d8	11
10	3	Email	c40f2602-8a7c-429e-bf13-cb1ec9e5f92f	9
11	2	Call	c20ab049-cbac-4ba7-8868-310aa89e0549	9
12	5	Call	0b026b91-fe12-4af0-86f9-387ba81c8fdb	11
13	2	Email	6103bcac-9da6-4000-a0ce-fa2615cce846	10
14	5	Call	96c8b5b8-cb81-4c75-a284-0e0026a03be8	10
15	4	Email	189d4f1b-9e76-4f64-9e71-7bd9b133a2d1	10

12,500 rows ⚠ truncated from 15,000 rows

```
customer_count_per_method = df.groupby('sales_method')['customer_id'].nunique()
print(customer_count_per_method)
```

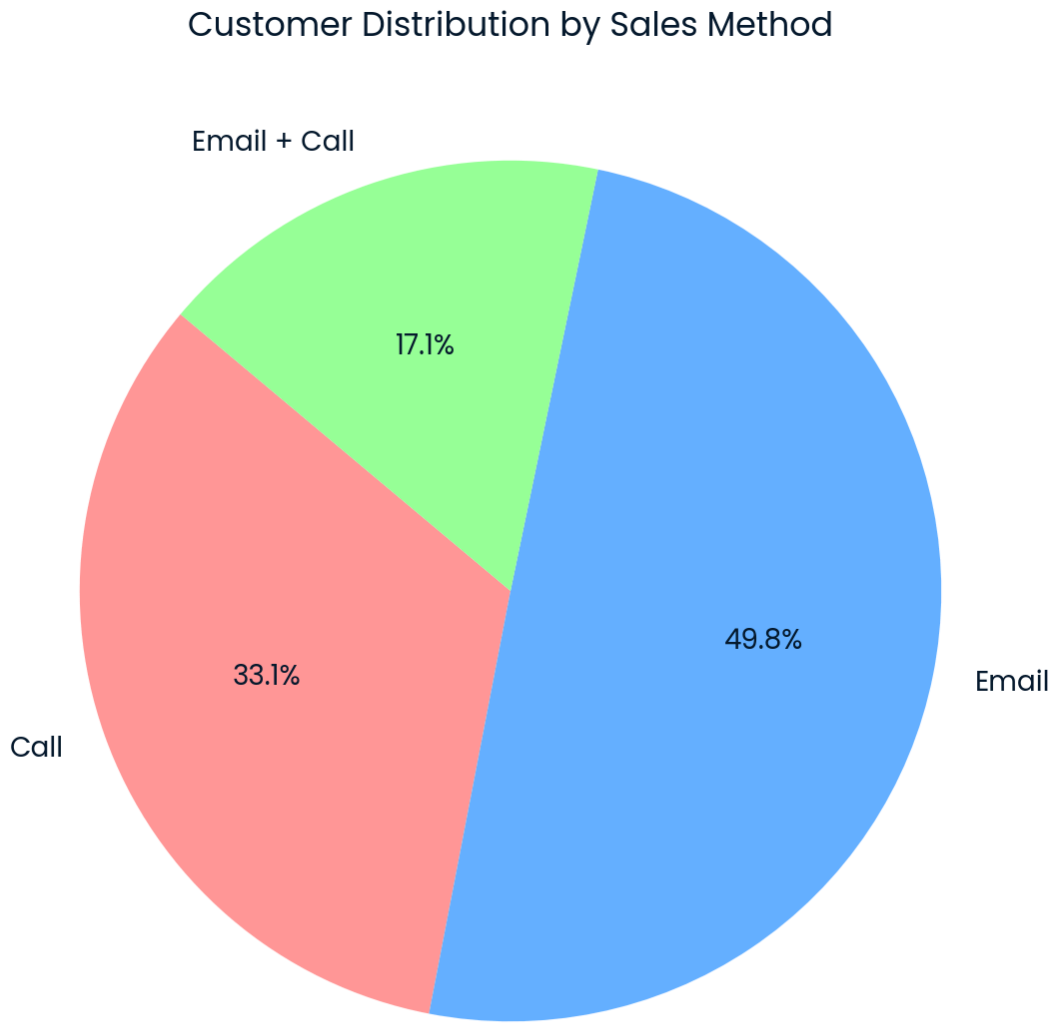
```
sales_method
Call            4962
Email           7456
Email + Call    2549
em + call         23
email             10
Name: customer_id, dtype: int64
```

```
sales_method_mapping = {
    'Call': 'Call',
```

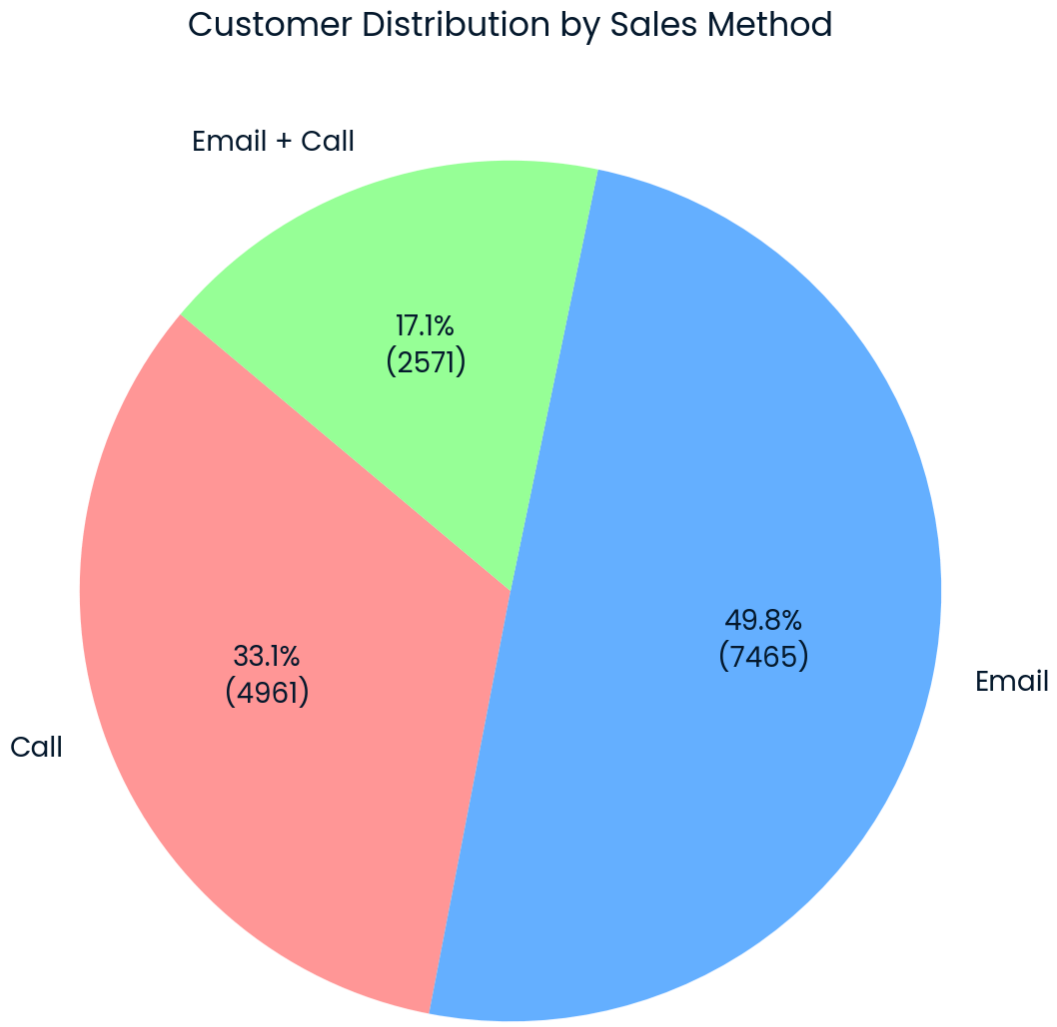
```
    'Email': 'Email',
    'email': 'Email',
    'Email + Call': 'Email + Call',
    'em + call': 'Email + Call'
}

# Apply the mapping to the 'sales_method' column
df['sales_method'] = df['sales_method'].replace(sales_method_mapping)

# Now group by the standardized sales_method and count unique customer_ids
customer_count_per_method = df.groupby('sales_method')['customer_id'].nunique()

print(customer_count_per_method)
```

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

```
import matplotlib.pyplot as plt

# Get the index (sales method names) and values (customer counts)
sales_methods = customer_count_per_method.index.tolist()  # List of sales method names
customer_counts = customer_count_per_method.values        # List of customer counts

# Create the pie chart
plt.figure(figsize=(7,7))
plt.pie(customer_counts, labels=sales_methods, autopct='%1.1f%%', startangle=140, colors=
['#ff9999','#66b3ff','#99ff99'])

# Add a title
plt.title('Customer Distribution by Sales Method')

# Show the pie chart
plt.show()
```




```
# Get the index (sales method names) and values (customer counts)
sales_methods = customer_count_per_method.index.tolist()  # List of sales method names
customer_counts = customer_count_per_method.values        # List of customer counts

# Define a function to format the labels with both percentage and count
def format_autopct(pct, allvals):
    absolute = int(pct/100. * sum(allvals))  # Calculate the raw count
    return "{:.1f}%\n({:d})".format(pct, absolute)  # Format as percentage and count

# Create the pie chart
plt.figure(figsize=(7,7))
plt.pie(customer_counts, labels=sales_methods, autopct=lambda pct: format_autopct(pct,
customer_counts),
        startangle=140, colors=['#ff9999','#66b3ff','#99ff99'])

# Add a title
plt.title('Customer Distribution by Sales Method')

# Show the pie chart
plt.show()
```




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

	revenue
count	15000
unique	6744
top	NA
freq	1074

4 rows

```
# Check the number of missing values in the 'revenue' column
missing_revenue_count = df['revenue'].isna().sum()

print(f"Number of missing or invalid 'revenue' values: {missing_revenue_count}")
```

```
Number of missing or invalid 'revenue' values: 1074
```

```
# Calculate the percentage of missing revenue values
total_rows = len(df)
```

```
missing_percentage = (missing_revenue_count / total_rows) * 100

print(f"Percentage of missing 'revenue' values: {missing_percentage:.2f}%")
```

```
Percentage of missing 'revenue' values: 7.16%
```

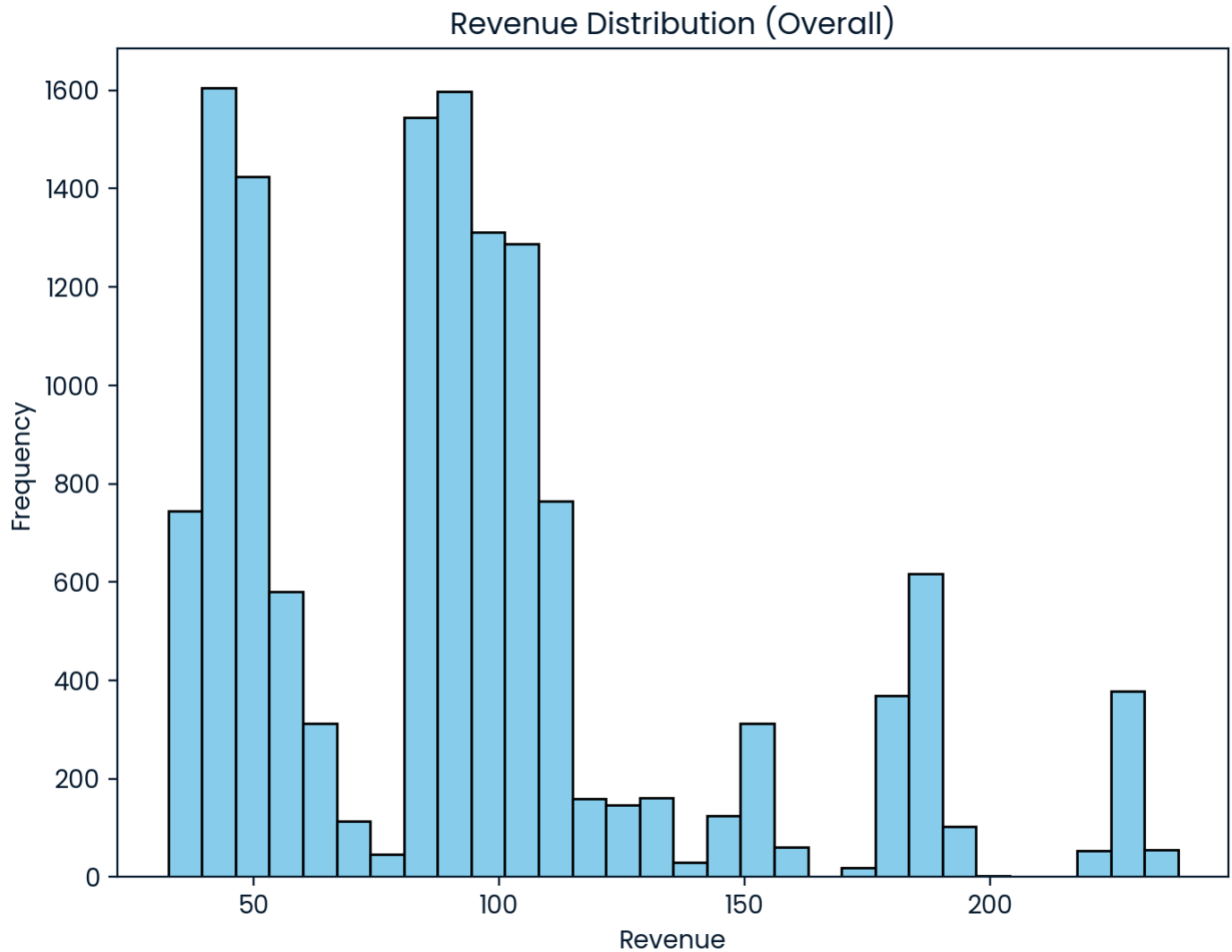
```
# Overall summary statistics
revenue_summary = df['revenue'].describe()

# Summary statistics by sales method
revenue_by_method_summary = df.groupby('sales_method')['revenue'].describe()

print(revenue_summary)
print(revenue_by_method_summary)
```

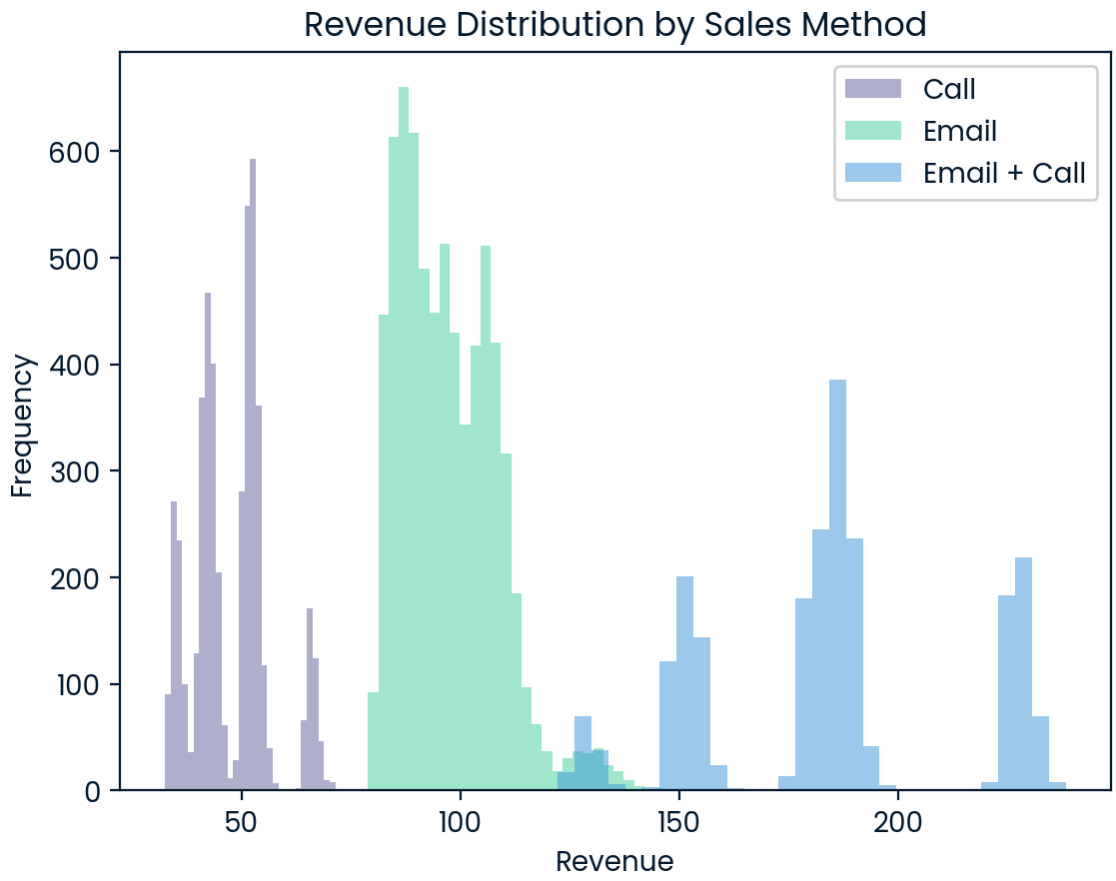
```
count    13926.000000
mean        93.934943
std         47.435312
min         32.540000
25%         52.470000
50%         89.500000
75%        107.327500
max        238.320000
Name: revenue, dtype: float64
               count        mean        std  ...     50%     75%     max
sales_method                                ...
Call          4781.0   47.597467   8.609899  ...   49.07   52.68   71.36
Email         6922.0   97.127684  11.210469  ...   95.58  105.17  148.97
Email + Call  2223.0  183.651233  29.083924  ...  184.74  191.11  238.32

[3 rows x 8 columns]
```

```
import pandas as pd

# Convert 'revenue' column to numeric, invalid parsing will be set to NaN
df['revenue'] = pd.to_numeric(df['revenue'], errors='coerce')

# Optionally, drop rows where 'revenue' is NaN (if you don't want to fill them)
df_clean = df.dropna(subset=['revenue'])
```

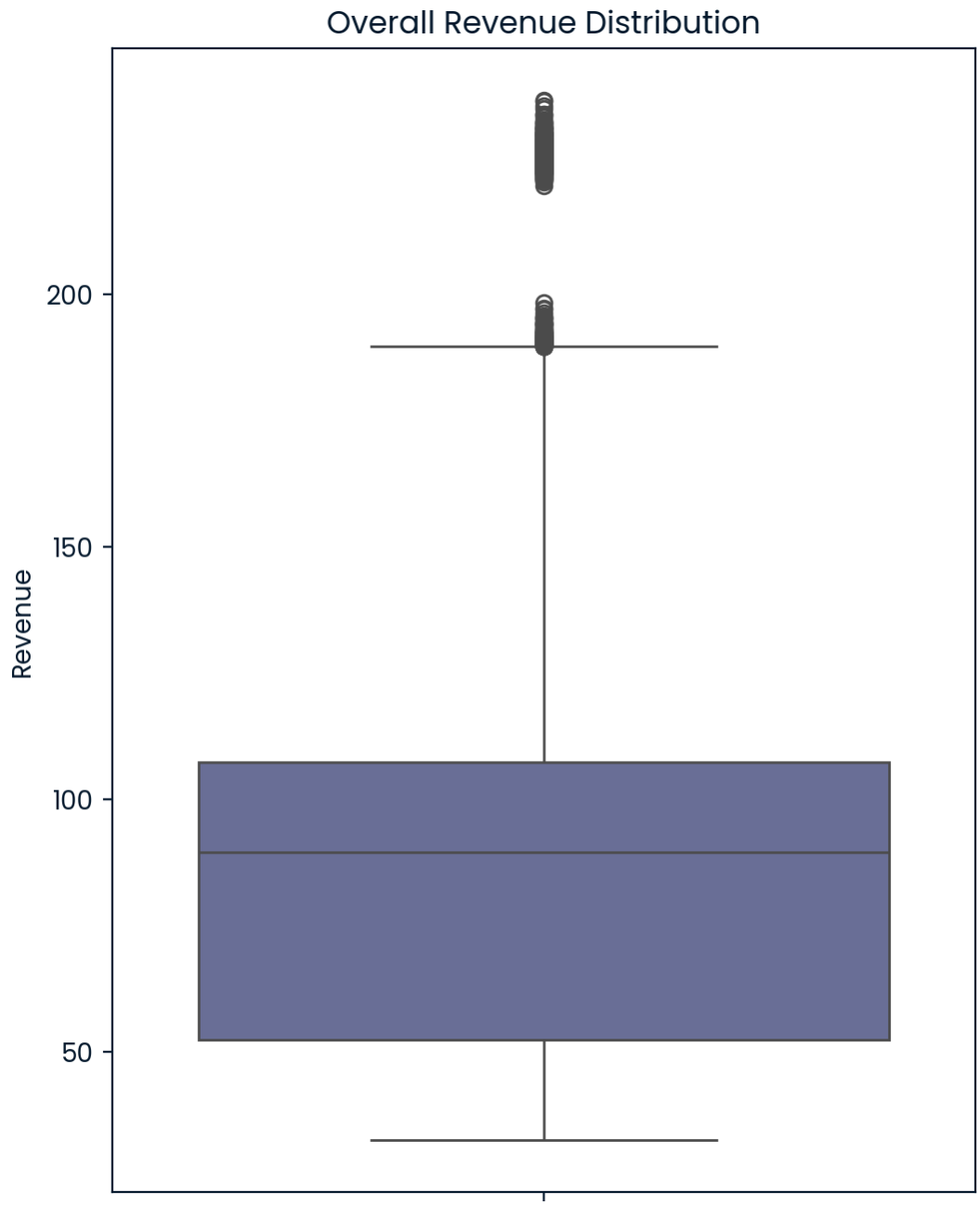
```
# Overall revenue histogram
plt.figure(figsize=(8,6))
plt.hist(df['revenue'], bins=30, color='skyblue', edgecolor='black')
plt.title('Revenue Distribution (Overall)')
plt.xlabel('Revenue')
plt.ylabel('Frequency')
plt.show()

# Histogram by sales method
```

```
df.groupby('sales_method')['revenue'].plot(kind='hist', alpha=0.5, bins=30, legend=True)
plt.title('Revenue Distribution by Sales Method')
plt.xlabel('Revenue')
plt.ylabel('Frequency')
plt.show()
```





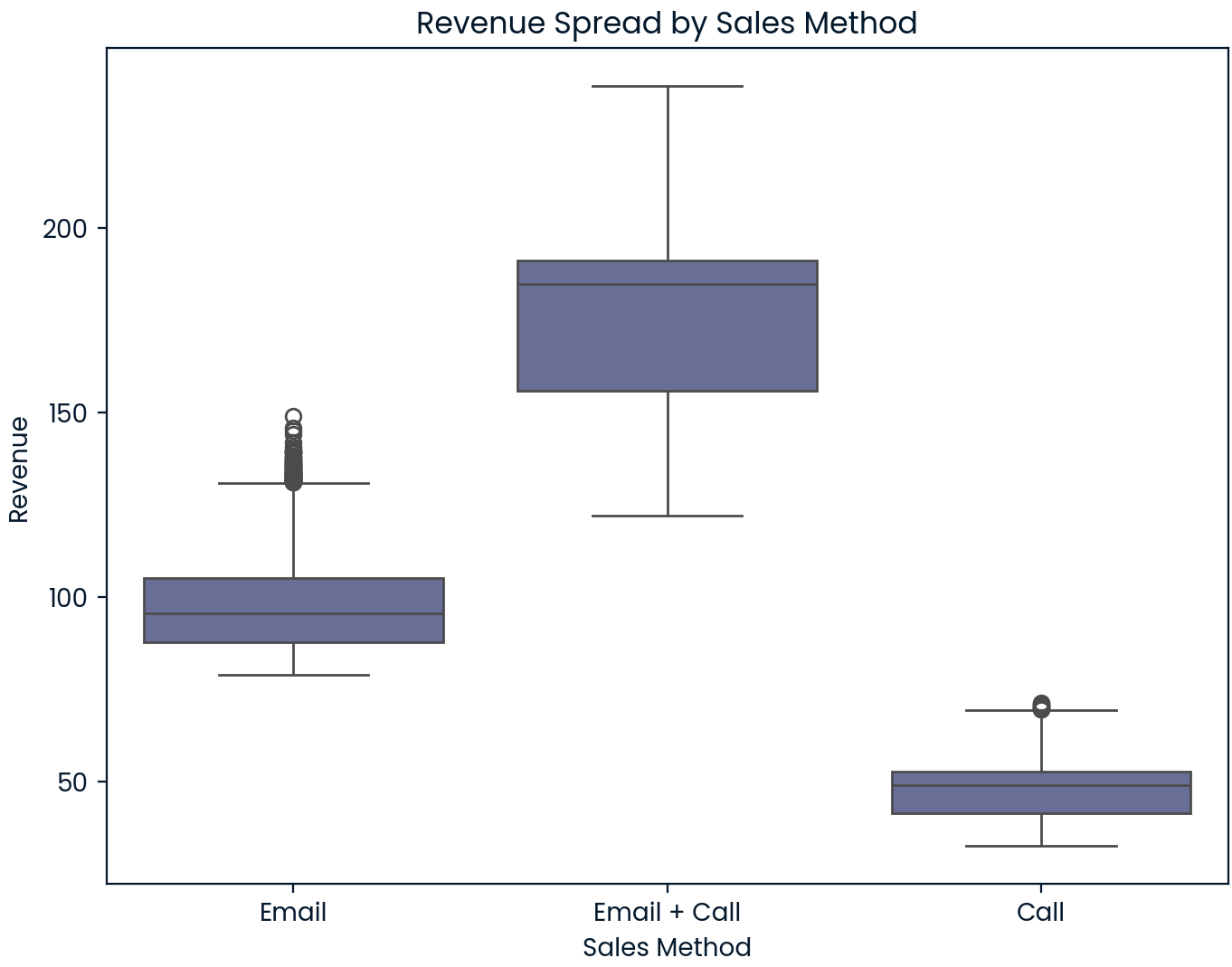

```
# Create the box plot for overall revenue
plt.figure(figsize=(6,8))
sns.boxplot(y=df_clean['revenue'])
plt.title('Overall Revenue Distribution')
plt.ylabel('Revenue')
plt.show()

# Print the summary statistics for overall revenue
revenue_summary = df_clean['revenue'].describe()
print(revenue_summary)
```



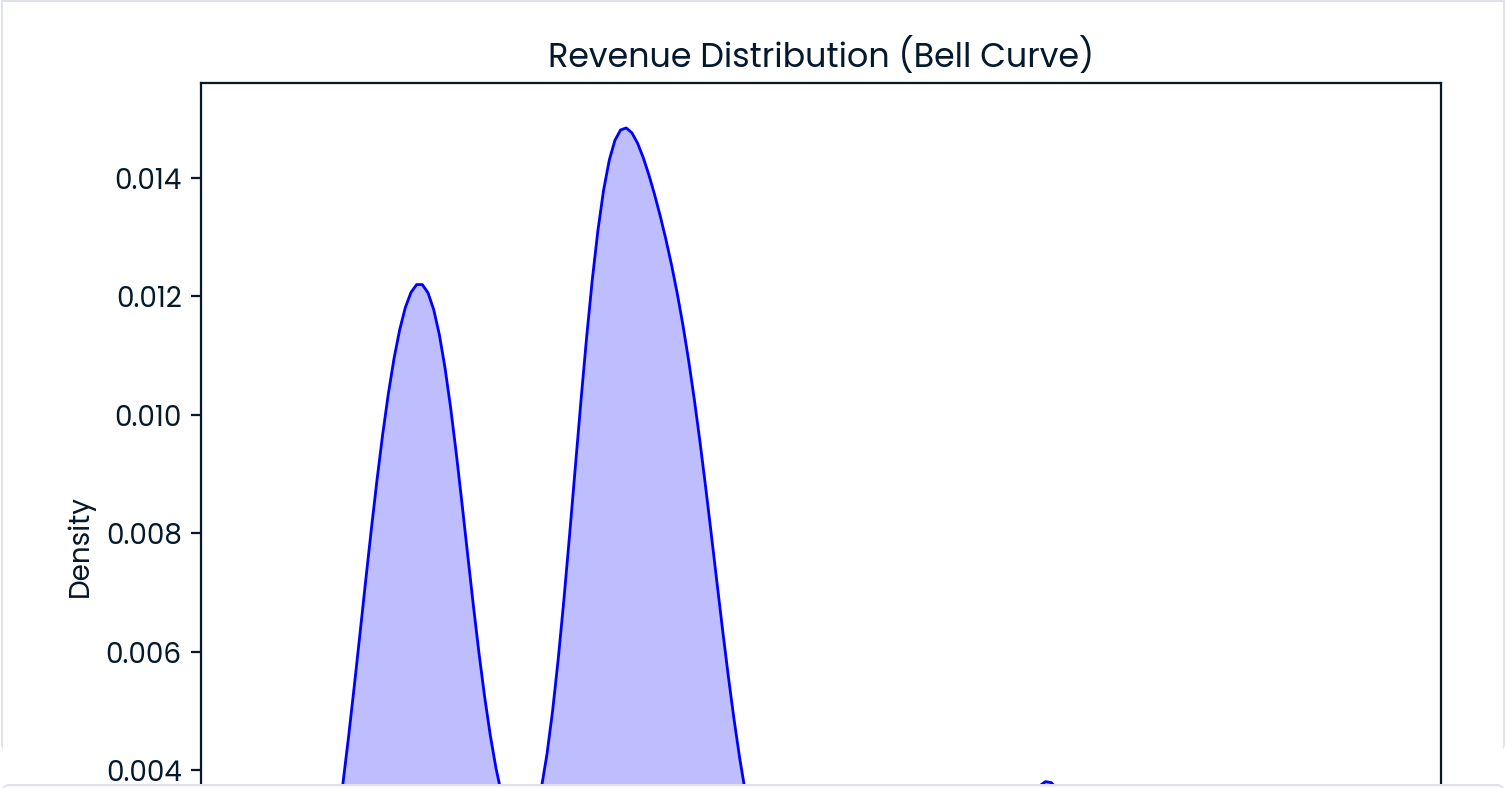

```
count    13926.000000
mean        93.934943
std         47.435312
min         32.540000
25%         52.470000
50%         89.500000
75%        107.327500
max        238.320000
Name: revenue, dtype: float64
```

```
import seaborn as sns

# Box Plot of Revenue by Sales Method
plt.figure(figsize=(8,6))
sns.boxplot(x='sales_method', y='revenue', data=df)
plt.title('Revenue Spread by Sales Method')
plt.ylabel('Revenue')
```

```
plt.xlabel('Sales Method')
plt.show()
```




```
# Create the KDE plot for the overall revenue distribution
plt.figure(figsize=(8,6))
sns.kdeplot(df_clean['revenue'], shade=True, color='blue')

# Add title and labels
plt.title('Revenue Distribution (Bell Curve)')
plt.xlabel('Revenue')
plt.ylabel('Density')

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




```python
# KDE Plot of Revenue by Sales Method

plt.figure(figsize=(8,6))
for method in df['sales_method'].unique():
    sns.kdeplot(df[df['sales_method'] == method]['revenue'], label=method)

plt.title('Revenue Distribution by Sales Method')
plt.xlabel('Revenue')
plt.ylabel('Density')
plt.legend(title='Sales Method')
plt.show()
```