Data Analyst Professional Practical Exam Submission

You can use any tool that you want to do your analysis and create visualizations. Use this template to write up your summary for submission.

You can use any markdown formatting you wish. If you are not familiar with Markdown, read the Markdown Guide before you start.

Task List

Your written report should include written text summaries and graphics of the following:

- Data validation:
  - Describe validation and cleaning steps for every column in the data
- Exploratory Analysis:
  - Include two different graphics showing single variables only to demonstrate the characteristics of data
  - Include at least one graphic showing two or more variables to represent the relationship between features
  - Describe your findings
- Definition of a metric for the business to monitor
  - How should the business use the metric to monitor the business problem
  - Can you estimate initial value(s) for the metric based on the current data
- Final summary including recommendations that the business should undertake

Start writing report here..

Sales Strategy Analysis for New Office Stationery Line

Summary

we introduced a new line of office stationery aimed at stimulating creativity and improving brainstorming skills. In order to optimize the sales approach for this new product line, three different sales strategies were tested: email, call and a combination email and call. The aim of this analysis is to evaluate the efficiency of these sales approaches and make the best suggestions for future sales strategies.

The following report will detail the data validation and cleaning process, present the results of the exploratory data analysis, define a metric that the business will monitor, and make recommendations based on those findings.

With data-driven insights, we can hopefully make better decisions to drive improvement in sales performance that can guarantee success for this new product line.

```
#Setting up the environment with importing required libraries and data
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
%matplotlib inline
```

Data Validation

The raw data from the original dataset is made up of 15,000 rows and 8 columns.

Week Column

- This column is an integer type, ranging between 1 and 6, each representing a week, respectively, with no null values.

Sales Method Column

- The column has 5 different values: `Email`, `Call`, `Email + Call`, `em + call`, `email`, and no null values. The column should only have 3 different values per sales method type. An overview showed that, aside from simple mistakes of different capitalizations, the use of abbreviations resulted in two other sets of values.

Customer ID Column

- It includes a unique identifier for each customer, and there were no null values within this column.

Number Sold Column

- It describes the number of items sold, and there are no missing values within this column.

Revenue Column

- This column had 1,074 null values. The null values were replaced with the mean revenue corresponding to each sales method.

Years as Customer Column

-This column had no null values; however, it had two invalid rows: 47 and 63 years. Given that the company was founded in the year 1984, the maximum number of years possible as a customer is 39. The two invalid values had been replaced with 39.

Number of Site Visits Column

• There are no missing values in this column. • All data in this column is valid.

State Column

This column has no missing values and contains 50 unique values for each state.

After cleaning up the data and validating it, there were no duplicate rows. Now, the changed dataset consists of 15,000 rows and 8 columns with no missing values.

```
# import data
sales_data = pd.read_csv('product_sales.csv')
sales_data.head()
```

	week	sales_method	customer_id
0	2	Email	2e72d641-95ac-497b-bbf8-4861764a7097
1	6	Email + Call	3998a98d-70f5-44f7-942e-789bb8ad2fe7
2	5	Call	d1de9884-8059-4065-b10f-86eef57e4a44
3	4	Email	78aa75a4-ffeb-4817-b1d0-2f030783c5d7
4	3	Email	10e6d446-10a5-42e5-8210-1b5438f70922

5 rows

```
sales_data.shape
```

```
(15000, 8)
```

```
sales_data.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            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
```

```
sales_data.describe()
```

	week	nb_sold	revenue
count	15000	15000	
mean	3.0982666667	10.0846666667	
std	1.6564198071	1.8122133327	
min	1	7	
25%	2	9	
50%	3	10	
75%	5	11	
max	6	16	

8 rows

```
# Check the value counts to ensure there are only 3 unique values
print(sales_data['sales_method'].value_counts())
```

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

```
# Define a mapping dictionary to correct the inconsistent values
sales_method_mapping = {
    'Email': 'Email',
    'Call': 'Call',
    'Email + Call': 'Email + Call',
    'em + call': 'Email + Call',
    'email': 'Email'
}

# Apply the mapping to the 'sales_method' column
sales_data['sales_method'] = sales_data['sales_method'].map(sales_method_mapping)

# Check the value counts to ensure there are only 3 unique values
print(sales_data['sales_method'].value_counts())
```

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

```
# find mean revenue for each sales method
mean_revenue_by_sales_method = sales_data.groupby('sales_method')['revenue'].mean()
print(mean_revenue_by_sales_method)
```

```
sales_method
Call             47.597467
Email            97.127684
Email + Call    183.651233
Name: revenue, dtype: float64
```

```
def replace_null_revenue(row):
    """
    Replaces null (NaN) values in the 'revenue' column of a pandas DataFrame with the mean
(or median) revenue
    for the corresponding 'sales_method' group.

    Parameters:
    -----------
    row : pandas Series
        A single row of a pandas DataFrame containing the 'revenue' and 'sales_method'
columns.

    Returns:
    --------
    float
        The value of the 'revenue' column for the given row, either the original value if it
is not null, or
        the mean (or median) revenue for the corresponding 'sales_method' group if it is
null.
    """
    if pd.isnull(row['revenue']):
        return mean_revenue_by_sales_method[row['sales_method']]
    else:
        return row['revenue']
```

```
# apply function to the revenue column
sales_data['revenue'] = sales_data.apply(replace_null_revenue, axis=1)

# check for any null values in the revenue column
print(sales_data['revenue'].isnull().sum())
```

```
0
```

```
sales_data[sales_data['years_as_customer'] > 39]
```

	week	sales_method	customer_id
13741	2	Email	18919515-a618-430c-9a05-2c7d8fea96af
13800	4	Call	2ea97d34-571d-4e1b-95be-fea1c404649f

2 rows

```
# find all values > 39 and replace with 39
```

```
sales_data.loc[sales_data['years_as_customer'] > 39, 'years_as_customer'] = 39
```

```
# check to see if replacement worked
sales_data[sales_data['years_as_customer'] > 39]
```

Your query ran successfully but returned no results.

```
# check number of unique values for state
sales_data['state'].nunique()
```

```
50
```

```
# check if any duplicate rows
duplicate_rows = sales_data[sales_data.duplicated()]
duplicate_rows
```

Your query ran successfully but returned no results.

```
sales_data.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  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
```

Exploratory Analysis

A. Number of Customers by Sales Method

The selling method that was most implemented during the 6-week period was via Email, at 7,466 customers. Following that was the Call method, which amounted to 4,962, and last was Email + Call, amounting to 2,572.

```
# find number of customers for each sales method
customers_by_sales_method = sales_data['sales_method'].value_counts()
```

```
print(customers_by_sales_method)
```

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

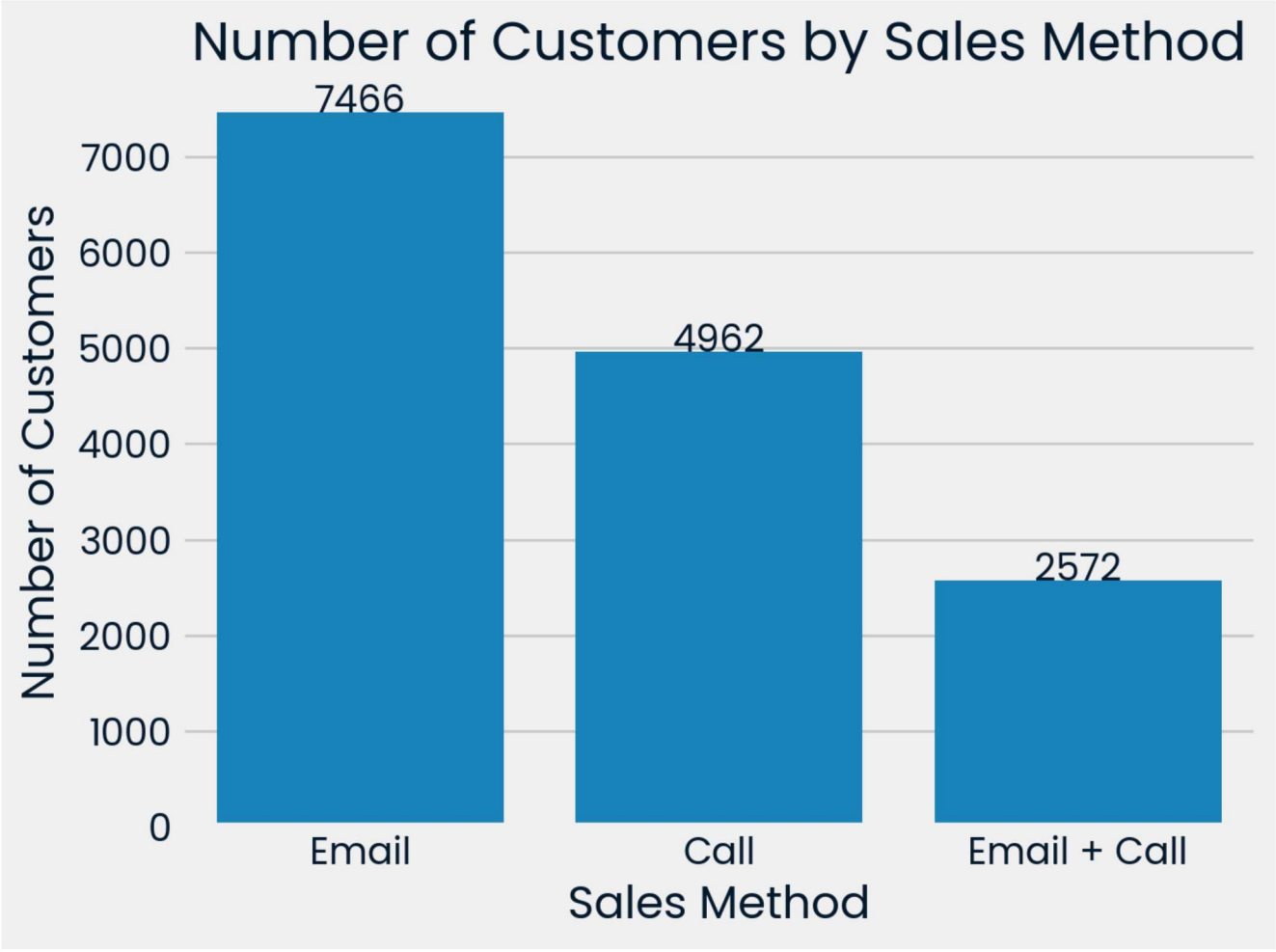
```
customers_by_sales_method = sales_data['sales_method'].value_counts()

ax = sns.barplot(x=customers_by_sales_method.index, y=customers_by_sales_method.values)

plt.title("Number of Customers by Sales Method")
plt.xlabel("Sales Method")
plt.ylabel("Number of Customers")

# Add value labels to each bar
for i, v in enumerate(customers_by_sales_method.values):
    ax.text(i, v + 0.5, str(v), ha='center')

plt.show()
```



B. Revenue Distribution: Overall and by Sales Method

Overall Revenue Distribution

First, we analyzed the overall revenue distribution to see how the revenue is spread.

Histogram Analysis

A histogram of overall revenue depicts the distribution of revenue across all methods of sales. This pictorial representation enhances understanding and thereby allows for identification of the central tendency, dispersion, and any skewness in the revenue data.

Revenue Distribution by Sales Method

We further analyzed the revenue distribution across the different sales methods in order to compare their performances.

BoxPlot Analysis

Box plots here show the distribution of revenues for each of the sales methods: Email, Call, and Email + Call. This offers the opportunity to compare the median revenues, quartiles, and outliers across the different methods.

Email Method: The below box plot illustrates the median of the Email method against its inter-quartile range and extreme values.

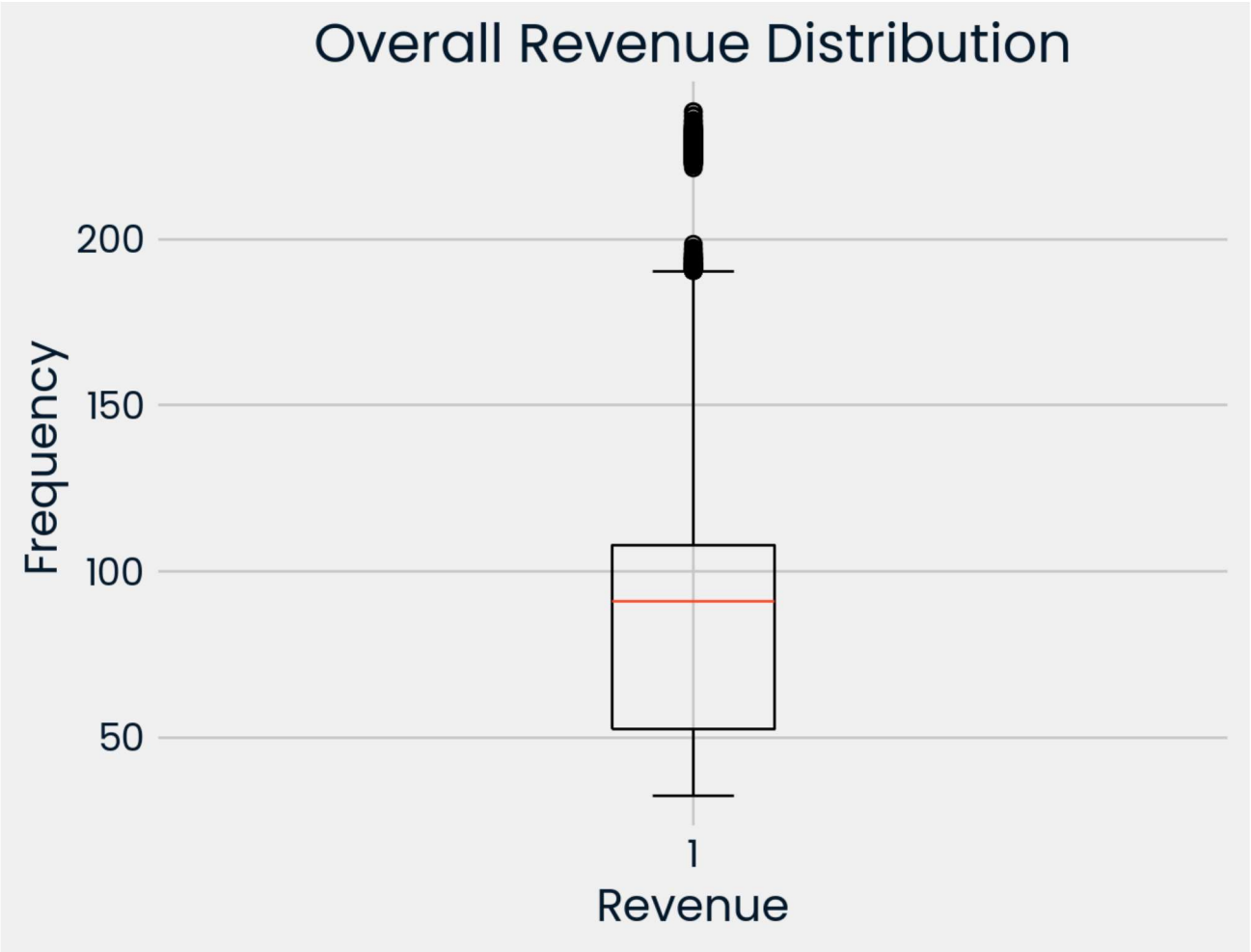
Call Method: The revenue distribution of the Call method is depicted by the box plot.

Email + Call Method: The revenue distribution of the Email + Call method is plotted on the box plot.

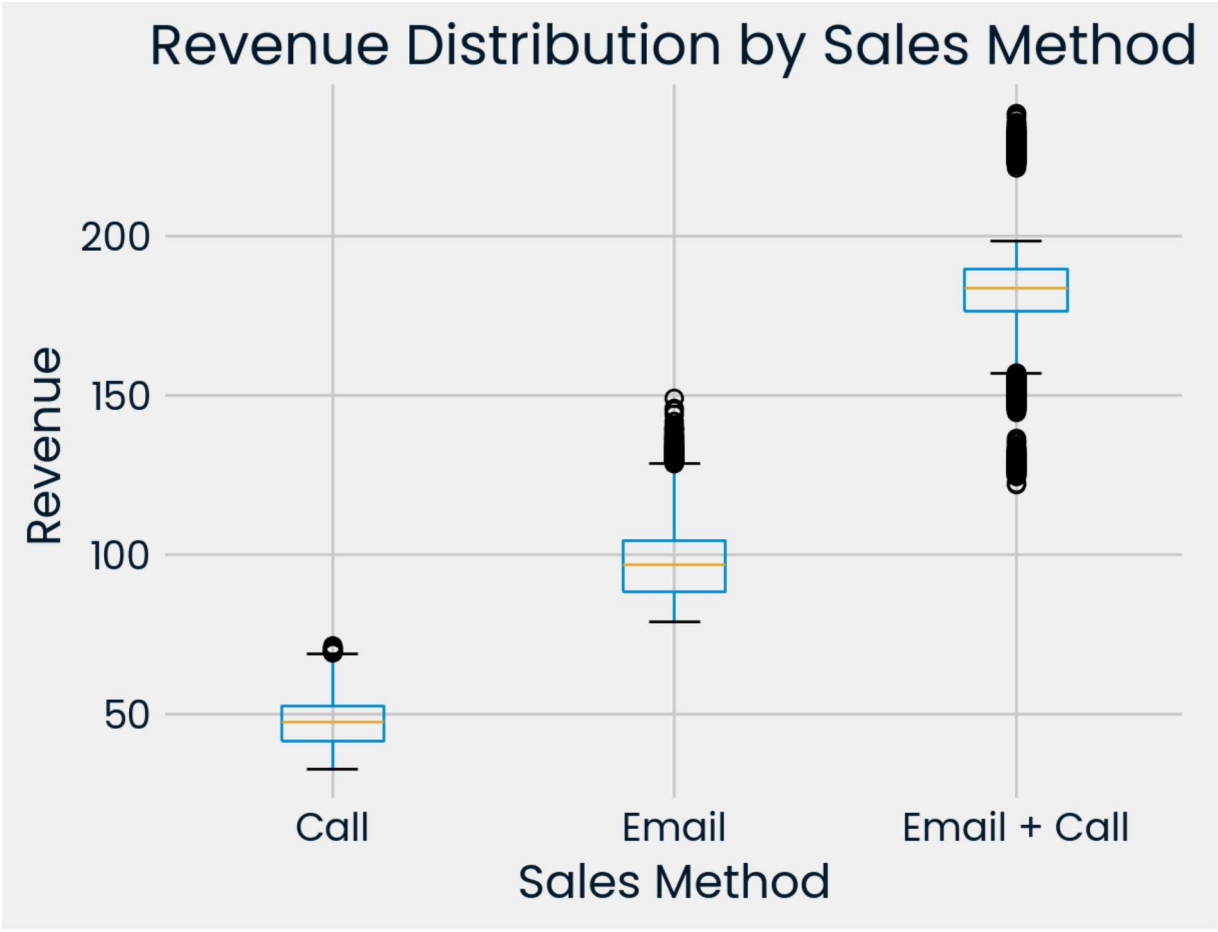
Key Observations

From the box plots, the following are the key observations: Median Revenue: The median revenue for each of the methods brings out the central tendency of the revenues that are made. Dispersion: The IQR is the measure of dispersion or scatter of revenue recorded by each method. Outliers: Any outliers within the revenues signify unusual or extreme revenue values recorded.

```
# Histogram for overall revenue
plt.boxplot(sales_data['revenue'])
plt.xlabel('Revenue')
plt.ylabel('Frequency')
plt.title('Overall Revenue Distribution')
plt.show()
```




```
sales_data.boxplot(column='revenue', by='sales_method')
plt.xlabel('Sales Method')
plt.ylabel('Revenue')
plt.title('Revenue Distribution by Sales Method')
plt.suptitle('')  # Remove auto-generated sup-title
plt.show()
```



C. Revenue Over Time for Each Sales Method

The revenue trajectory across the various sales methods created a number of interesting patterns throughout the six-week period:

Week 1 Revenue Insights:

- Email sales method generated the highest revenue at $48,122.68
- Call method produced $27,015.93
- Email + Call method generated $20,007.40

Revenue Trend Observations:

1. **Email Method:**
    - Started with the highest revenue Trend of a regular decline in the six weeks Indicates the possible diminishing returns with an email-only approach
2. **Call Method:** Total revenue indicated a range of upward trend Some variability of revenue Revenue in the last weeks of the six-week campaign period is downward trending
3. **Email + Call Method:** Revenue demonstrates a smooth and healthy increase

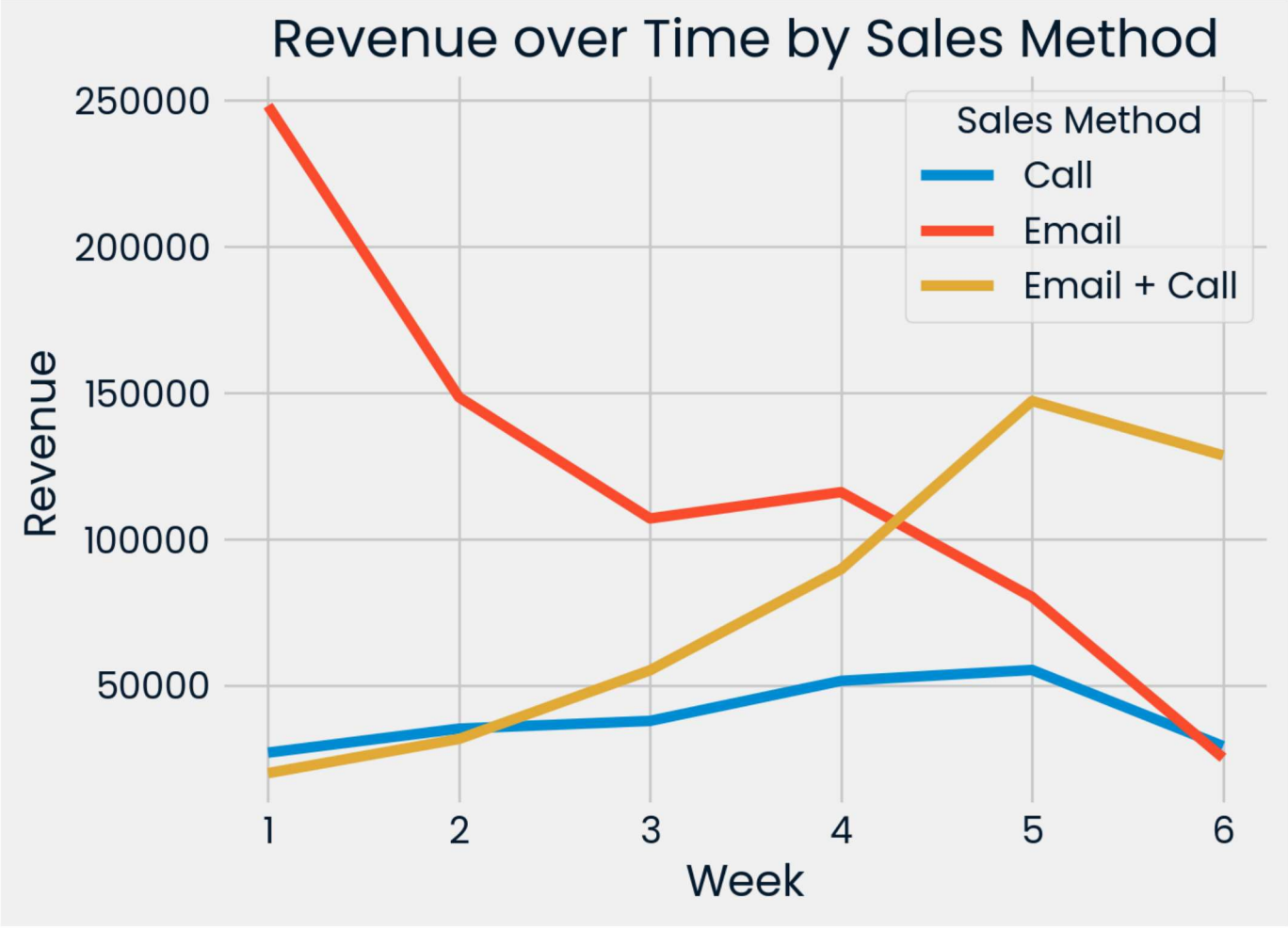
- Demonstrated most consistent growth pattern
    - Suggests potential for long-term revenue generation

Key Takeaways:

- Email + Call method appears most promising for sustained revenue growth
- Email method starts strong but is followed by rapid decline
- Call method shows average but inconsistent performance

Therefore, the Email + Call sales method appears most promising for long-term revenues but simultaneously requires much more effort on the part of a sales team. The Email method starts well but then shows continued decline, while the Call method has an increasing trend in general, though fluctuating more. It is recommended to consider both efficiency and the effort needed for each method when making a decision about continuing a specific sales method.

```
revenue_over_time = sales_data.groupby(['week', 'sales_method'])['revenue'].sum().unstack()
revenue_over_time.plot()
plt.xlabel('Week')
plt.ylabel('Revenue')
plt.title('Revenue over Time by Sales Method')
plt.legend(title='Sales Method')
plt.show()
```




```
revenue_over_time
```

week	Call	Email
1	27015.9344070278	
2	35219.944011713	
3	37865.5838799414	
4	51545.4861493411	
5	55279.2162811127	
6	29252.4668081991	

6 rows

D. Analyzing Other Customer Differences Between Groups

Customer behavior can be further analyzed by looking at some interesting observations related to different sales methods.

Customer Tenure:

- The years we have done business with customers in each sales method are relatively about the same; no substantial difference in customer loyalty or retention.

Sales Volume:

- The Email + Call method of sales is at the top, averaging 12 sales to one customer, beating out the Call and Email methods at 10 sales per customer.
- Whereas the Email + Call customers had 10-13 purchases, the Email and Call customers had much lower sales volumes, 8-11 and 9-11 items bought respectively.
- This means that customers reached through the channel of Email + Call shall tend to buy in bulk, increasing the potentiality of revenue generation.

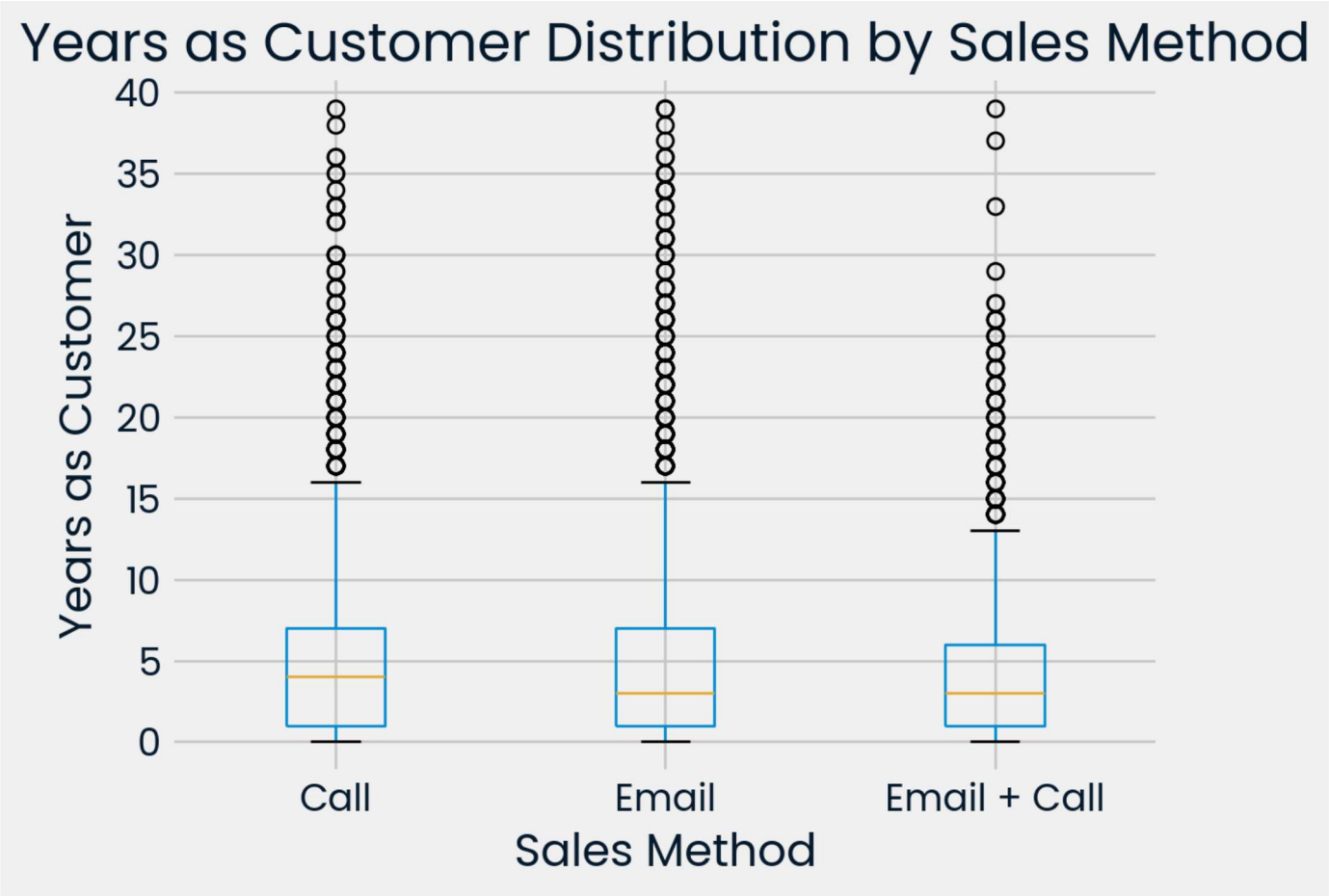
Website Engagement:

- The average site visits were higher for customers reached through the method of Email + Call compared to those reached via the methods of Email and Call individually.
- That would indicate that the Email + Call might drive site traffic better, which in turn may drive revenue.

These findings are useful in generating insight from customer behavior across the sales methods, showing the potential benefit of the email + call method for sales volume and website engagement.

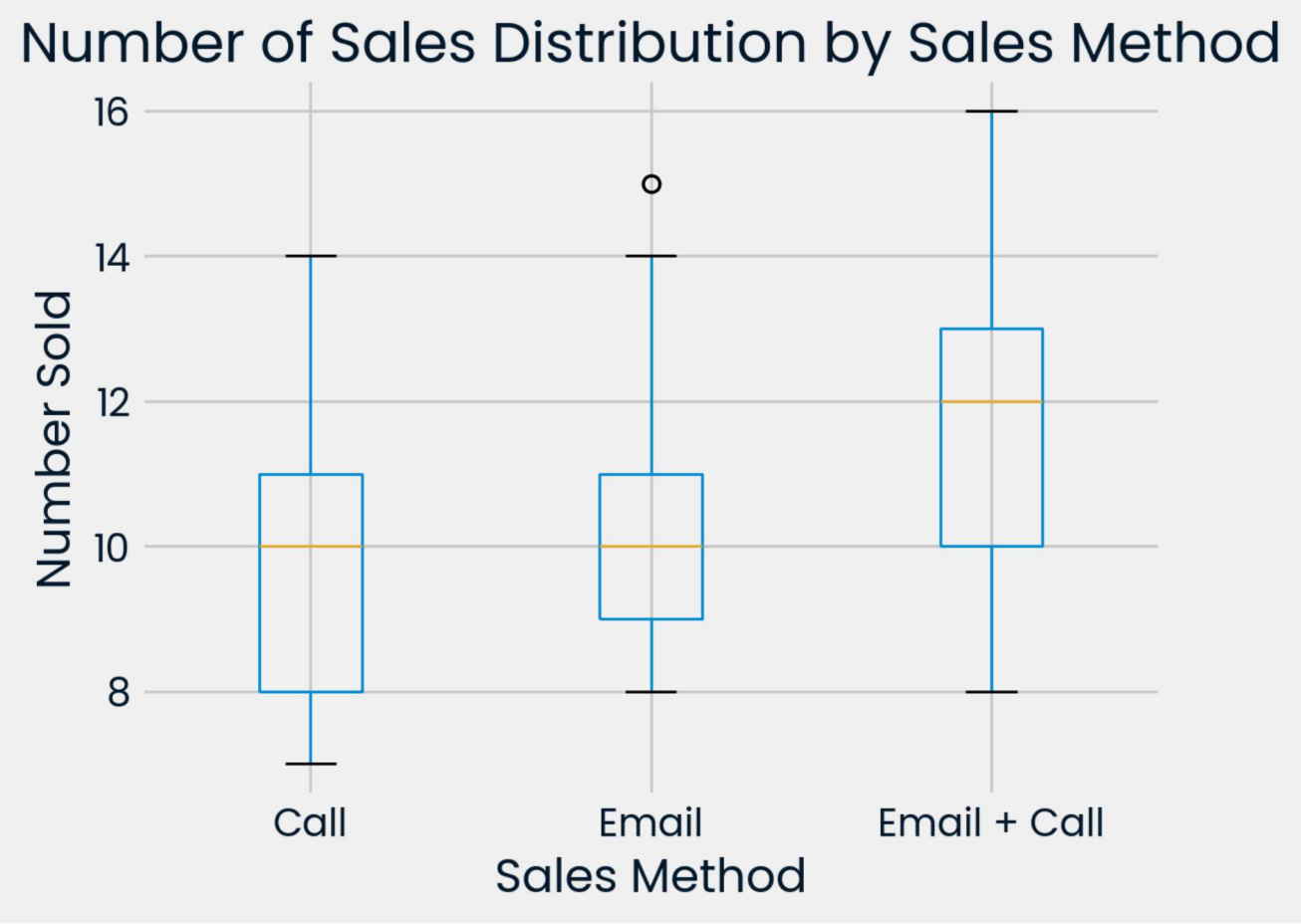
```
# Example: Boxplot for years_as_customer by sales_method
sales_data.boxplot(column='years_as_customer', by='sales_method')
plt.xlabel('Sales Method')
plt.ylabel('Years as Customer')
plt.title('Years as Customer Distribution by Sales Method')
plt.suptitle('')  # Remove auto-generated sup-title
plt.show()
```

```
# Other comparisons can be performed similarly (e.g., nb_site_visits, state, etc.)
```



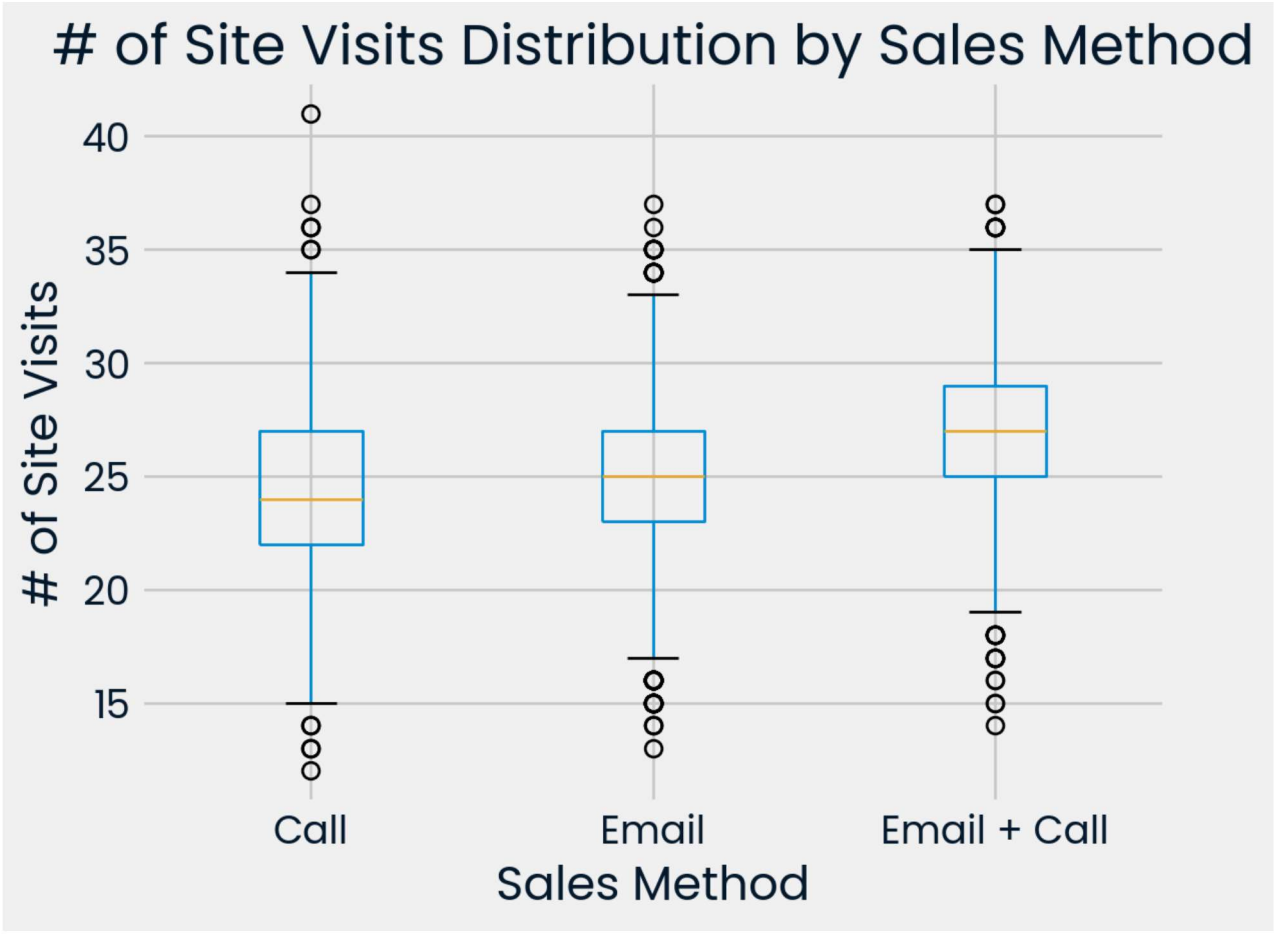

```
# Example: Boxplot for years_as_customer by sales_method
sales_data.boxplot(column='nb_sold', by='sales_method')
plt.xlabel('Sales Method')
plt.ylabel('Number Sold')
plt.title('Number of Sales Distribution by Sales Method')
plt.suptitle('')  # Remove auto-generated sup-title
plt.show()

# Other comparisons can be performed similarly (e.g., nb_site_visits, state, etc.)
```

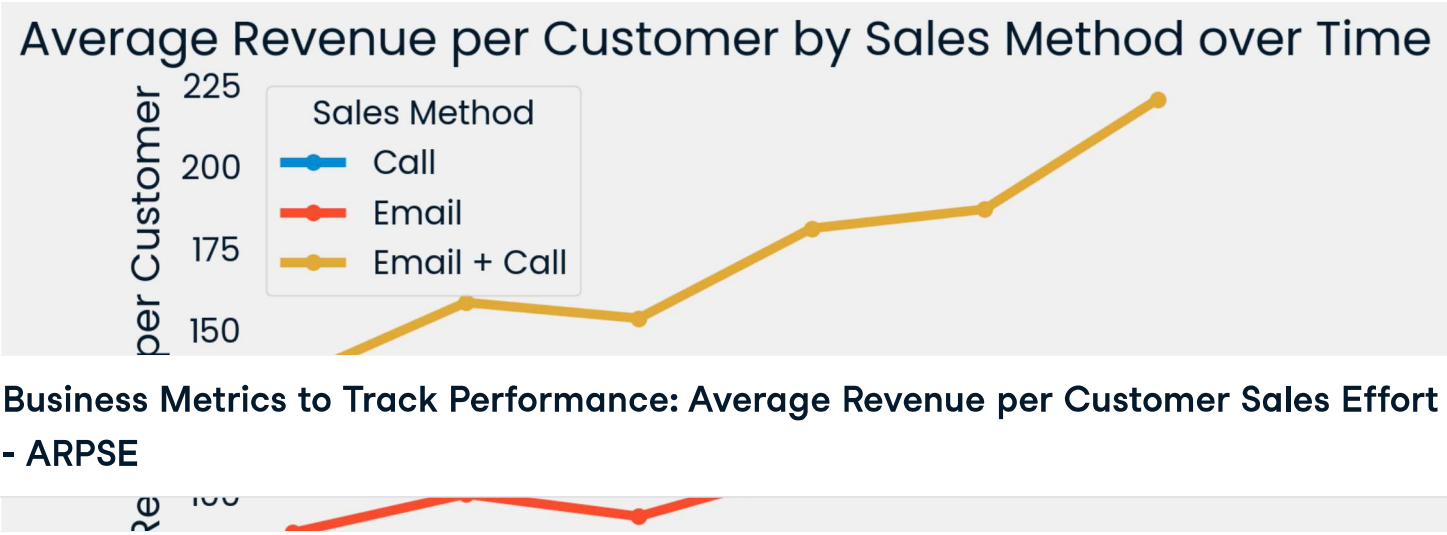



```
# Example: Boxplot for years_as_customer by sales_method
sales_data.boxplot(column='nb_site_visits', by='sales_method')
plt.xlabel('Sales Method')
plt.ylabel('# of Site Visits')
plt.title('# of Site Visits Distribution by Sales Method')
plt.suptitle('')  # Remove auto-generated sup-title
plt.show()

# Other comparisons can be performed similarly (e.g., nb_site_visits, state, etc.)
```




```
grouped_data = sales_data.groupby(['week', 'sales_method']).agg({'revenue': 'sum',
'customer_id': 'count'}).reset_index()
grouped_data['average_revenue_per_customer'] = grouped_data['revenue'] /
grouped_data['customer_id']
pivot_data = grouped_data.pivot_table(index='week', columns='sales_method',
values='average_revenue_per_customer')
pivot_data.plot(kind='line', marker='o')
plt.xlabel('Week')
plt.ylabel('Average Revenue per Customer')
plt.title('Average Revenue per Customer by Sales Method over Time')
plt.legend(title='Sales Method')
plt.grid()
plt.show()
```



Business Metrics to Track Performance: Average Revenue per Customer Sales Effort - ARPSE

Let me introduce a new metric in order to deepen the understanding of each sales method's actual performance: Average Revenue per Customer Sales Effort, which is the acronym for ARPSE. This metric considers the difficulty of each sales method depending on time spent, valued as follows:

- Email: 0.5
- Email + Call: 1
- Call: 3 Average Revenue per Customer Sales Effort will be calculated using the formula:

ARPSE = (Total Revenue for Method) / (Number of Customers * Sales Effort)

This metric provides a whole picture of each sales method by considering revenue generated by a particular method and the effort used to get such revenue. Further, ARPSE will show us which sales method to use more and which one to optimize.

```
# Define the sales effort for each sales method
sales_effort = {
    'Email': 0.5,
    'Call': 3,
```