Analysing the Effectiveness of Sales Strategies for the New Pen & Printers' Products

The sales rep provided the new Pen & Printers' product sales dataset and is mainly interested in determining which sales strategy is the most effective, out of the three, namely 'Email', 'Call', and 'Email + Call'. The sales rep also hope that the analysis could answer the following questions:

- How many customers were there for each approach?
- What does the spread of the revenue look like overall? And for each method?
- Was there any difference in revenue over time for each of the methods?
- Based on the data, which method would you recommend we continue to use? Some of these methods take more time from the team so they may not be the best for us to use if the results are similar.

A detailed description and dataset can be found in:
https://s3.amazonaws.com/talent-assets.datacamp.com/Practical+-+DAP+-+Product+Sales.pdf

All the data cleaning, analysis and visualisations were done through Python with the help of several common packages such as Pandas, Seaborn and Matplotlib.

Results Summary

Q1. Which sales approach is the most effective?

Answer: The **'Email + Call'** approach is the **most effective** sales strategy.

Q2. How many customers were there for each approach?

Answer:

- Email: 7466 customers
- Call: 4962 customers
- Email + Call: 2572 customers

Q3. What does the spread of revenue look like overall? And for each method?

Answer: The overall (individual customer) revenue follows a trimodal distribution (a distribution with 3 distinct peaks) with a range of $32.54 to $238.32 and a mean of $95.72. The 3 peaks in the overall revenue distribution to each of the sales approaches. The first on the left corresponds to the 'Call' method with a range of $32.54 to $71.36 and a mean of $47.64; the middle peak corresponds to the 'Email' method with a range of $78.83 to $148.97 and a mean of $97.19; while the peak on the right corresponds to the 'Email + Call' method with a range of $122.11 to $238.32 and a mean of $184.23.

Q4. Was there any difference in revenue over time for each of the methods?

Answer:

A) There was no difference in the average (individual customer) revenue over time. Generally, all 3 methods showed an increasing average (individual) revenue.

B) There were differences in the total revenue over time for each method. The total revenue for the 'Email' method peaked in Week 1 and followed a decreasing trend over time. However, the total revenue for the 'Call' and 'Email + Call' methods followed an increasing trend over time and peaked at Week 5 before decreasing in Week 6.

Q5. Based on the data, which method would you recommend we continue to use? Some of these methods take more time from the team so they may not be the best for us to use if the results are similar.

Answer: I would recommend the 'Email + Call' method as it generates the most individual revenue on average.

Detailed Analysis and Report

Data Validation

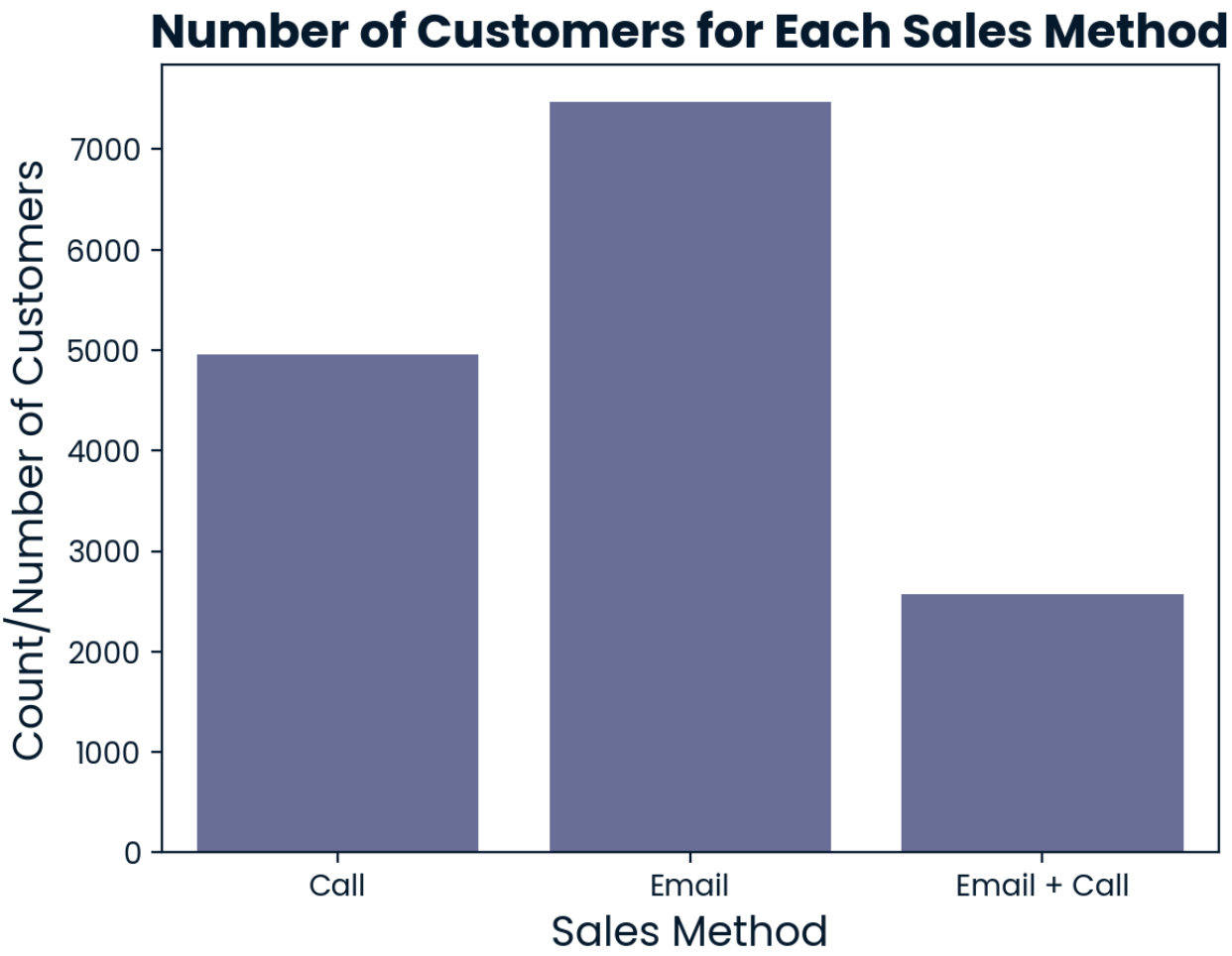
The dataset contained 15000 rows and 8 columns before cleaning and validation. I have cleaned and validated all the columns against the criteria provided in the information table (found in the pdf link above) as summarized in the following:

- week: 6 unique integer values (ranges from 1 to 6), which recorded the week the sale was made after the launch of the new product. No missing values were detected, and the values were in a sensible range. No cleaning is needed.
- sales_method: The raw data contains 5 categories instead of 3. Upon further inspection, the additional 2 categories were misspelt/formatted in a different manner. The labels were then standardized to 'Email', 'Email + Call', and 'Call' to refer to the 3 sales methods. No missing values were detected.
- customer_id: 15000 unique ID, same as description. No cleaning is needed.
- nb_sold: 10 unique integer values (ranges from 7 to 16). No missing values were detected. No cleaning is needed.
- revenue: 2 decimals numeric values with 1074 missing values (7.16% missing from the dataset). Missing data was imputed with the average revenue calculated from the subset of its respective sales method (sales_method) and the number of new products sold (nb_sold). This was possible as all the missing values have at least one non-missing value that corresponds to the same sales method and number of new products sold. Mean was chosen based on the current available information.
- years_as_customer: Integer values with no missing data. However, there are 2 observations that are out of range, which were 47 and 63 years, respectively (the maximum possible value should be 41 years since the company was founded in 1984). Since the incorrect observations only account for 0.013% of the dataset (negligible percentage error in the whole dataset), I decided to treat these observational data as NA as they would not substantially affect the analysis.
- nb_site_visits: Integer values with no missing data, same as description. No cleaning is needed.
- state: 50 possible character values with no missing data, same as description. No cleaning is needed.

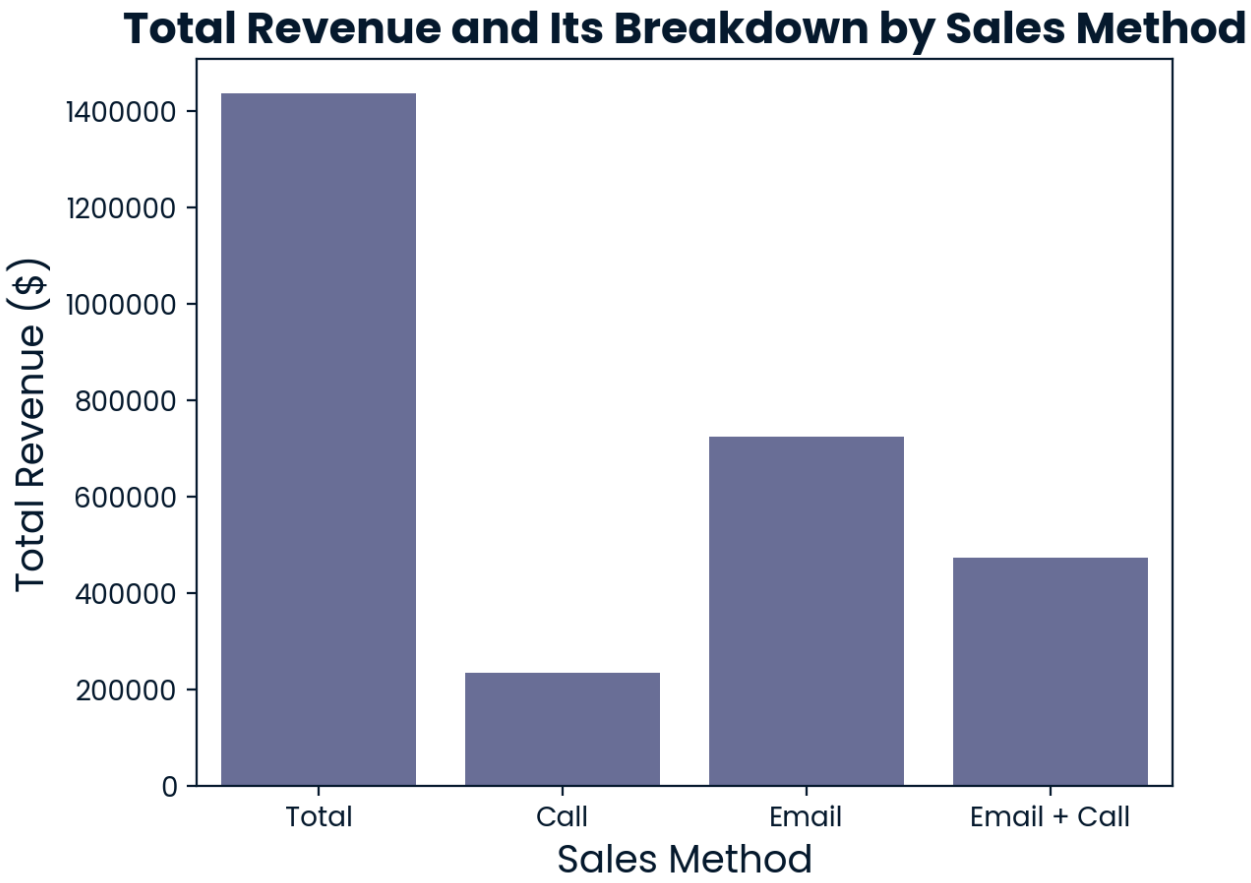
After data validation, there are 15000 rows and 8 columns without missing values except for 2 imputed missing values in the 'years_as_customer' column.

1. How many customers were there for each approach?

From the past 6 weeks' records, 'Email' has the most customers (at 7466 customers), followed by 'Call' (at 4962 customers) and lastly by 'Email + Call' (at 2572 customers). The number of customers targeted with the 'Call' approach is about double the amount of 'Email + Call' while the 'Email' amount is about triple the 'Email + Call'.



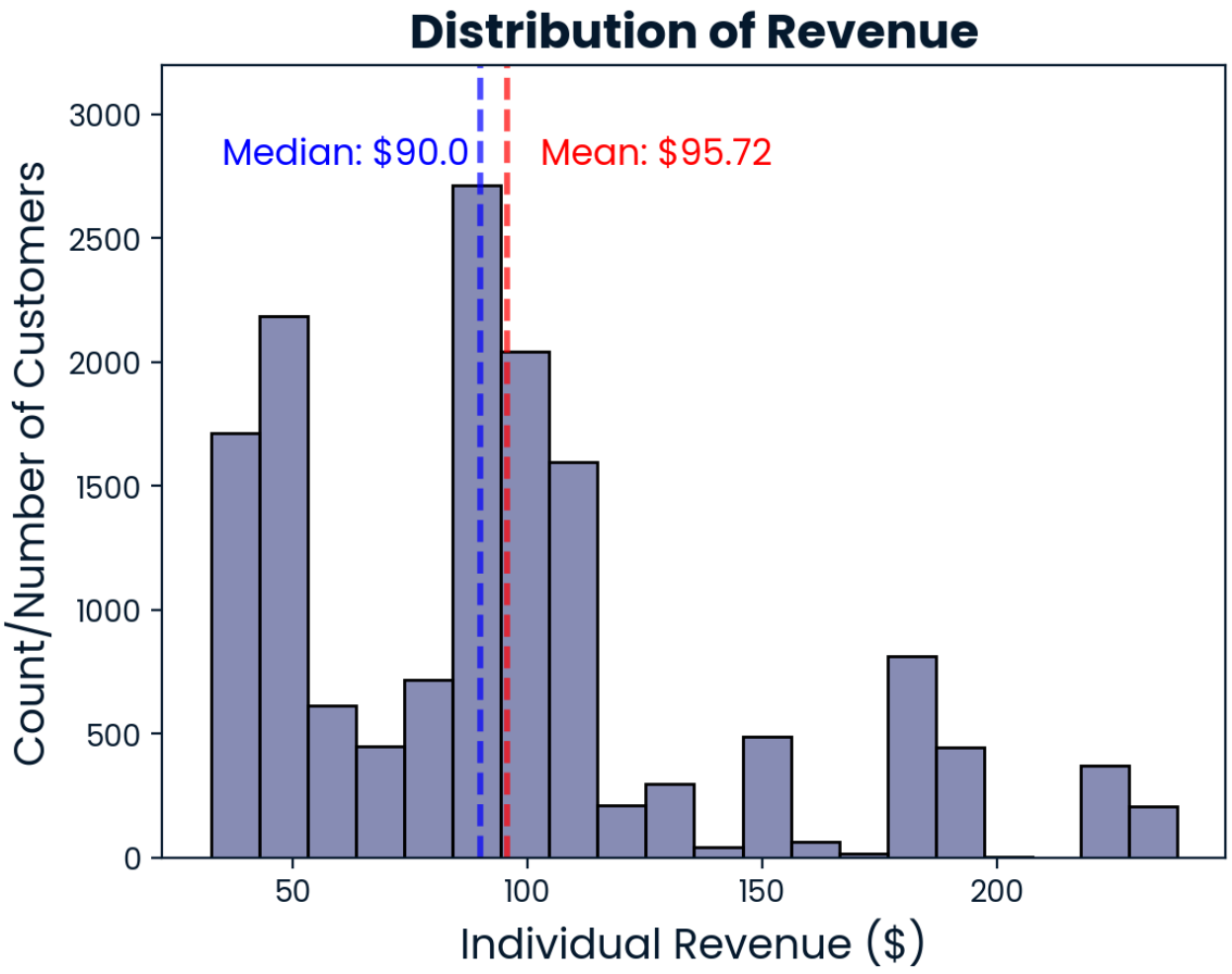
Due to the disproportionate number of customers in each sales method, the total revenue might mislead us in identifying the effectiveness of each sales method. The bar plot below showcases the total revenue and its breakdown by sales method. The new product line managed to produce a total of $1,435,809.00 of revenue from 15000 customers. The 'Email' method generated the most total revenue ($725,583.66), followed by the 'Email + Call' method ($473,827.98), and, lastly, the 'Call' method generated a total of $236,397.36 revenue. Surprisingly, the total revenue of the 'Email + Call' method is around double the 'Call' method, despite having half the number of customers being targeted by the 'Email + Call' method compared to the 'Call' method.



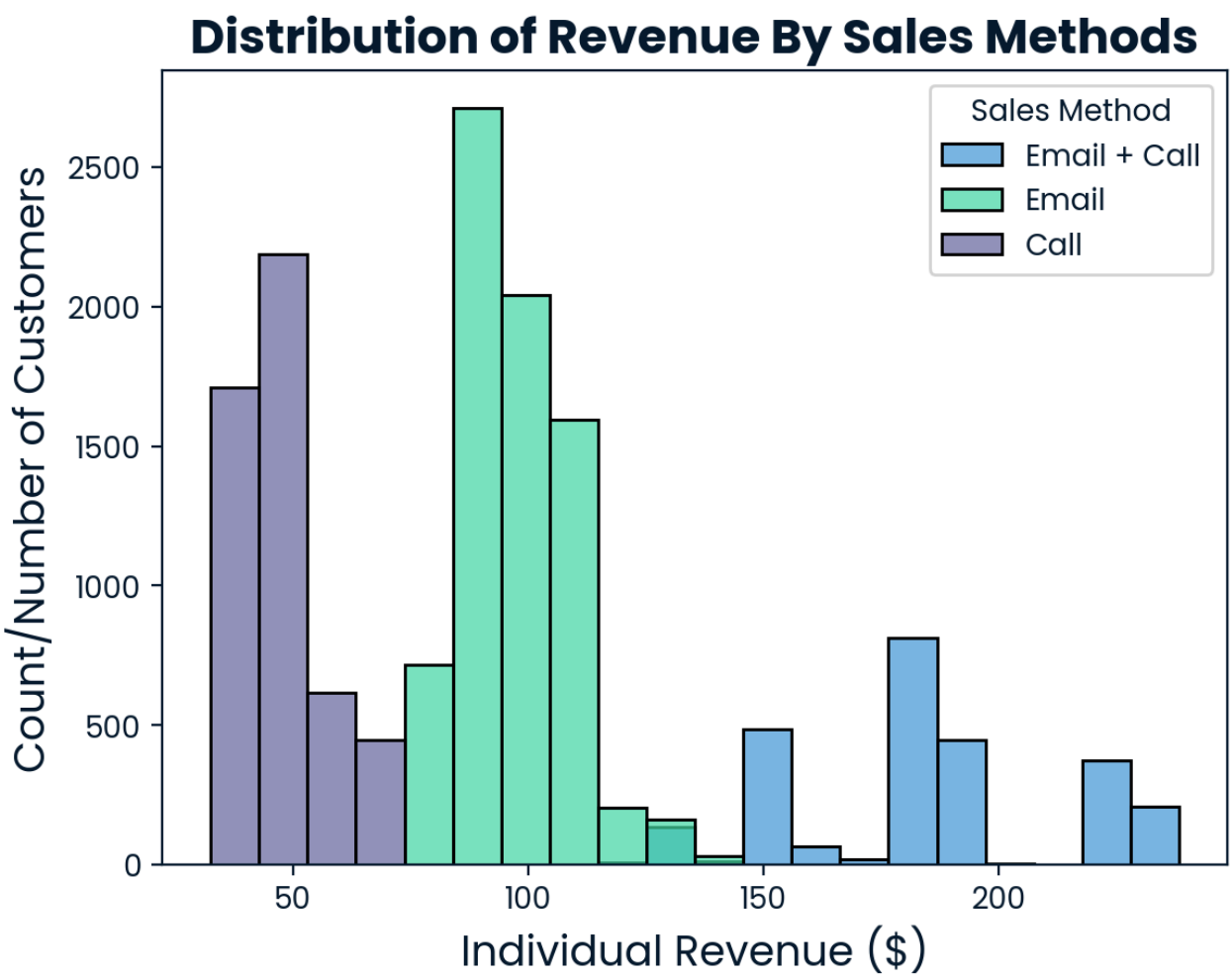
Thus, I recommend we evaluate the effectiveness of each sales method by looking at the revenue generated by each customer instead of the total revenue. This approach can inform us which sales method is best at encouraging our customers to buy more of our new products.

2. What does the spread of the revenue look like overall? And for each method?

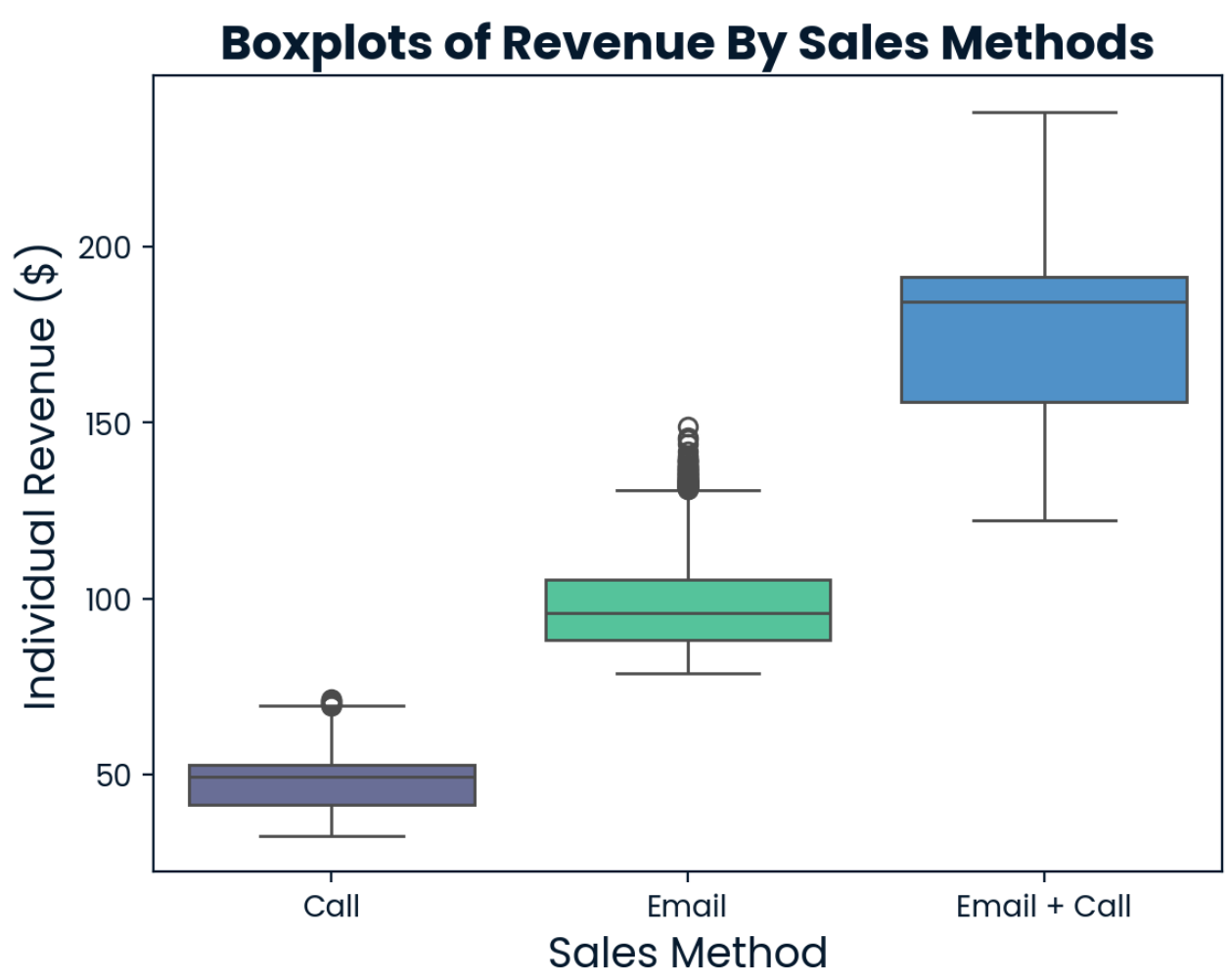
The histogram below showcases the distribution of each customer's revenue. The minimum amount spent by a customer to buy our new products was $32.54, and the most a customer spent was $238.32. On average, customers spent $95.72 on our new products (median was at $90.00). Upon closer inspection, the revenue histogram resembles a trimodal distribution (distribution with 3 peaks).



The histogram was replotted but now by sales methods. The histogram now shows that there are clear distinction in the range of revenue for each sales method. Customers targeted by the 'Email + Call' method mostly spent more in buying the new products compared to customers targeted by 'Email' and 'Call' methods.

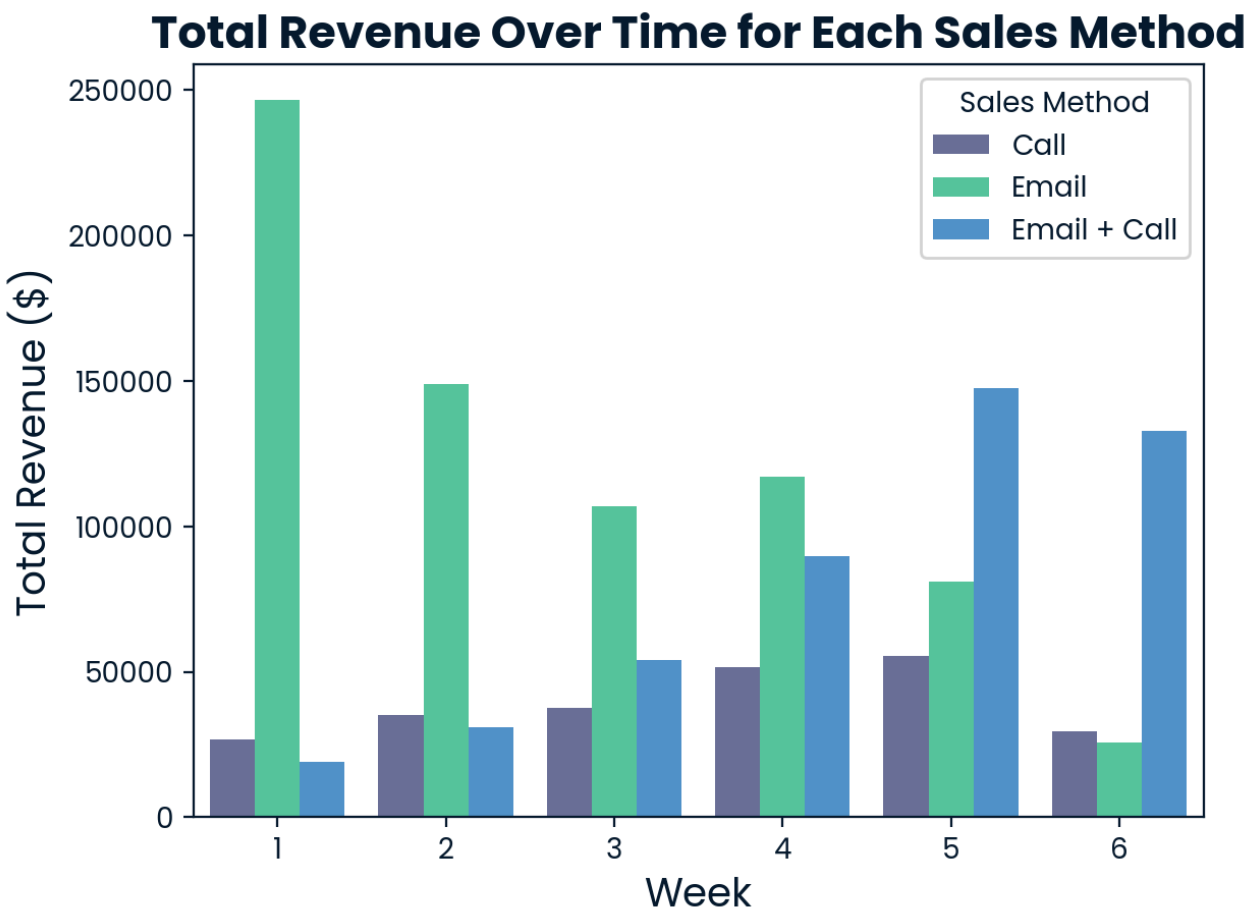


Boxplot of revenue for each sales method was then plotted to have a clearer visualisation of the revenue spread. The revenue of the 'Email + Call' group ranges from \$122.11 to \$238.32 with a mean of \$184.23, which is the highest on average compared to the 'Email' and 'Call' groups. The revenue of the 'Email' group ranges from \$78.83 to \$148.97 with a mean of \$97.19, which is higher than the 'Call' group but lower than the 'Email + Call' group. The 'Call' group has the lowest revenue group, which ranges from \$32.54 to \$71.36 with a mean of \$47.64. The results indicate that the 'Email + Call' sales method is likely the most effective way in encouraging customers to spend more on the new products with the 'Call' sales method being the least effective.

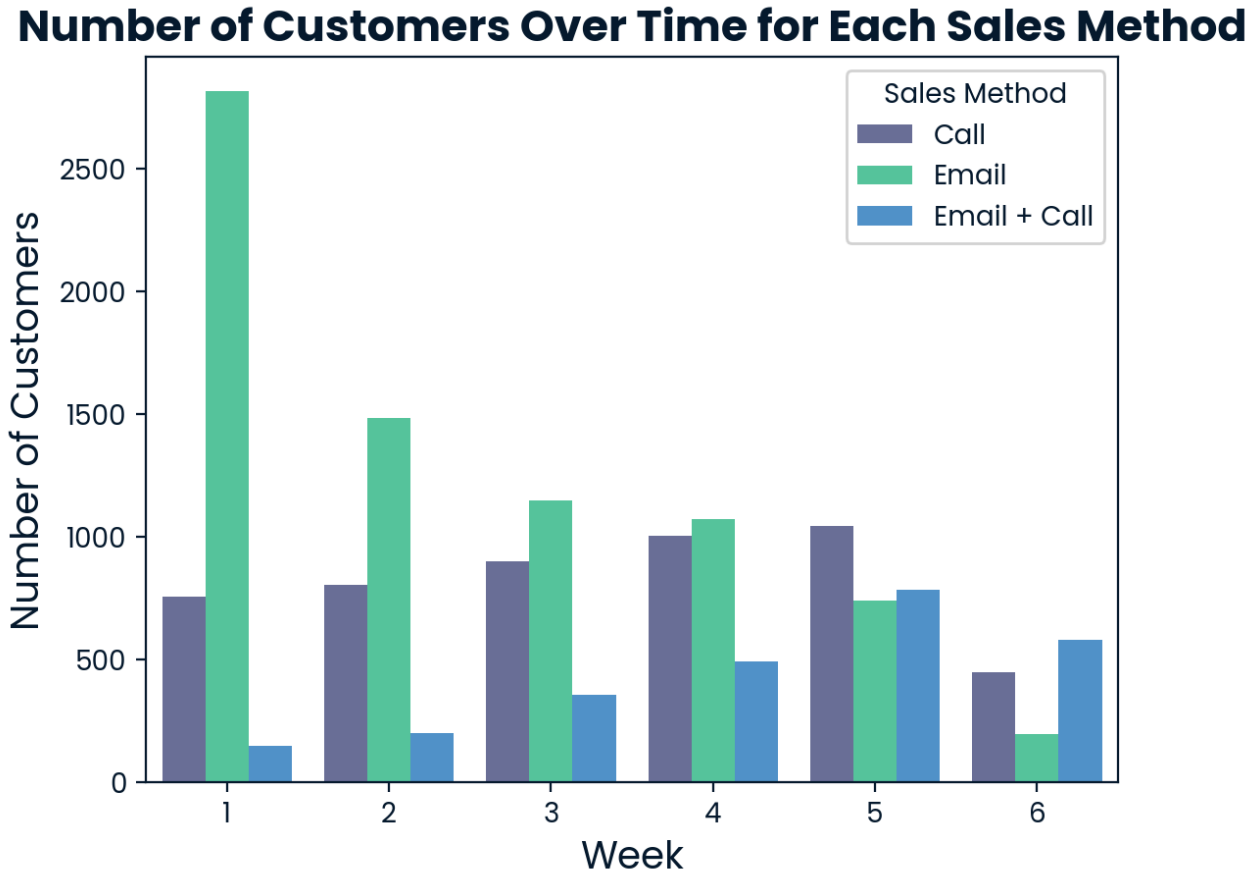


3. Was there any difference in revenue over time for each of the methods?

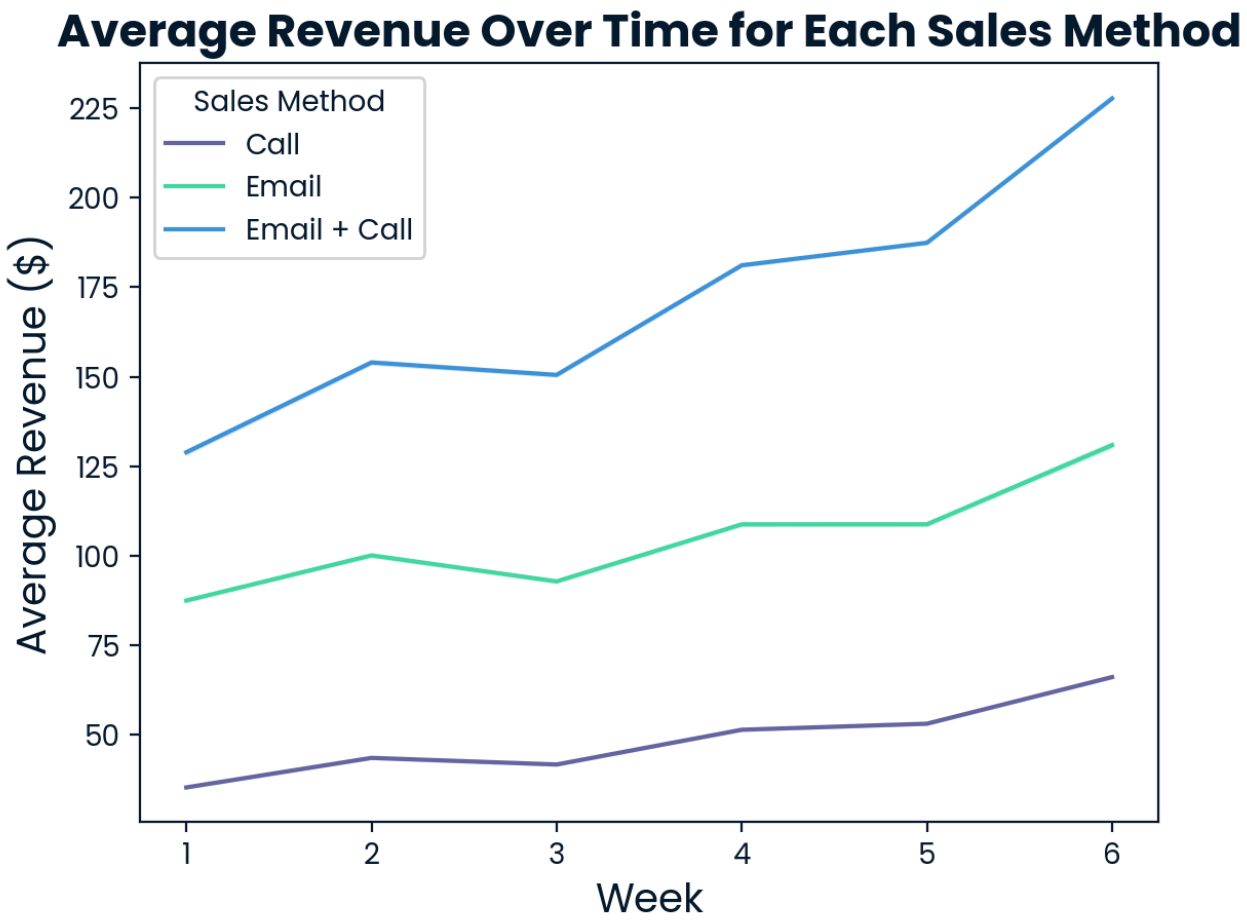
Across the 6-week period, the trend of the total revenue generated by each sales method differs. The total revenue for the 'Email' group peaked in Week 1 and gradually decreased over the period, though there was a slight increase in Week 4, possibly due to the follow-up email sent out in Week 3 to remind customers about the new products. As for the 'Call' and 'Email + Call' groups, the total revenue of both methods increased over the weeks and peaked at Week 5 before they decreased at Week 6. Among the 'Call' and 'Email + Call' groups, the 'Email + Call' group generally generated more total revenue than the 'Call' group, especially in Week 3 and after.



The trend of customers buying the new products over time is consistent with the total revenue trend for each group, except that there were more customers making a purchase in the 'Call' group compared to the 'Email + Call' group in each week, aside from Week 6.



Overall, the 'Email + Call' group has the highest average revenue per customer, followed by the 'Email' group, with the 'Call' group having the least average revenue per customer within these 6 weeks. However, there are no differences between the trend of average revenue per customer for each sales method over time. The average revenue of each sales method fluctuated over time, but they generally followed an increasing trend. The results suggest that the 'Email + Call' group generates the largest average revenue per customer, and the customers in this group are likely to purchase more in later weeks after the product launch.



4. Possible differences between customers in each group?

In summary, there are no other differences between customers in each group. All three groups share a similar trend that the number of new products sold is positively correlated with the revenue (strong positive correlation values in all three sales methods, where 'Call': 0.986, 'Email': 0.970, 'Email + Call': 0.990). This indicates that the more new products a customer buys, the higher the revenue generated from that customer.

Similarly, the revenue for the new products increases when a customer visits our website more frequently. This pattern is consistently seen across all three groups, where 'Call' has a moderate positive correlation value of 0.521, 'Email' has a moderately low positive correlation of 0.380, and 'Email + Call' has a moderate positive correlation value of 0.471.

No correlations were seen between the revenue and the number of years the customer has been buying from us among all three sales methods.

Furthermore, the Kruskal-Wallis-H-test did not show any significant evidence that the median revenue of each state for each sales group is different.

Sales Method	Kruskal-Wallis-H-test (p-value)
Call	0.365
Email	0.247
Email + Call	0.578

Recommended Sales Method

Based on the data from the past 6 weeks, I would recommend the **'Email + Call'** sales method as the data indicated that customers targeted by this method generally **spent more on buying the new products**.

The average revenue of the new products generated from the 'Email + Call' group is around **1.9 times that of 'Email'** and around **3.9 times that of 'Call'**.

Recommendations

Based on the analysis, I recommend the following:

- Target customers with the 'Email + Call' approach when new products are launched. If the 'Email + Call' approach is operationally too expensive, we could adopt the second-best approach, which is the 'Email' method.
- We could implement a secondary follow-up call after Week 5 to remind customers about the new products, as the analysis showed an increasing number of customers buying the new products after a call was made.
- When another new line of products is launched, track the number of customers buying the new products over time. If the number of customers targeted by the 'Email + Call' approach, buying the new products has not increased after Week 1, then immediate measures need to be taken to drive up sales, e.g. follow-up call to remind customers about the new products. Furthermore, we should also keep track of the average revenue per customer. If the average revenue per customer at Week 4 is less than 24% of Week 1, then attention is needed to drive more sales of the new products (the analysis above showed that the average revenue per customer increased by at least 24% in Week 4 compared to Week 1 for all sales methods).
- Data collection for in-depth analysis:
    - Continue collecting data after Week 6, as the customer's purchasing pattern might change over an extended period.
    - New related data - past purchase history of customers. To evaluate whether past purchase history will influence the customer's future purchase of new products.
    - New related data - customer purchase invoice. To evaluate the possible preference type of products our customer would like to purchase.

1 hidden cell

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.impute as imp
import scipy.stats as stats
import statsmodels.api as sm

product_sales = pd.read_csv('product_sales.csv')
print(product_sales.shape)
product_sales.head()
```

Hidden output

```
# Quick check on the data type and for any missing data in the dataframe
print(product_sales.dtypes) #Check for data type of each column in dataframe
print(product_sales.isna().any()) #Check for any potential missing data
# Result: Only 'revenue' column has missing data, possibly need imputation

# Identify the unique values in the 'week' column
print(product_sales['week'].sort_values().unique())

# Identify the unique values in the 'sales_method' column
print(product_sales['sales_method'].sort_values().unique()) # Some 'Email + Call'
got spelt as 'em + call' and 'Email' as 'email'. Fixed the problem in the
following.
product_sales['sales_method'] = product_sales['sales_method'].replace('email',
'Email')
product_sales['sales_method'] = product_sales['sales_method'].replace('em +
call', 'Email + Call')

# Check that the labels have been fixed.
print(product_sales['sales_method'].sort_values().unique()) # Results look okay
now

# Check through the 'customer_id' column
print(product_sales['customer_id'].value_counts().describe())

# Identify the unique values in the 'nb_sold' column
print(product_sales['nb_sold'].sort_values().unique())

# Check through the 'revenue' column
print(product_sales['revenue'].isna().value_counts()) # 1074 missing values
# Check whether it is possible to impute the missing 'revenue' values with mean,
estimated from the revenue grouped by sales method and number of new product sold
temp_na = product_sales[product_sales['revenue'].isna()] # Subset the rows with
missing values
temp_na = temp_na.value_counts(subset = ['sales_method', 'nb_sold'], dropna =
False).reset_index().sort_values(['sales_method', 'nb_sold'], ascending = [True,
True]).rename(columns = {0: 'NA_count'}) # Summarize the number of missing values
by sales method and by number of new products sold
temp_non_na = product_sales[~product_sales['revenue'].isna()] # Subset the rows
without missing values
temp_non_na = temp_non_na.value_counts(subset = ['sales_method', 'nb_sold'],
dropna = True).reset_index().sort_values(['sales_method', 'nb_sold'], ascending =
[True, True]).rename(columns = {0: 'non_NA_count'}) # Summarize the number of
non-missing values by sales method and by number of new products sold
temp_check = temp_non_na.merge(temp_na, how = 'outer', on = ['sales_method',
'nb_sold'])
temp_check['NA/non_NA (%)'] =
round(temp_check['NA_count']/temp_check['non_NA_count']*100, 1)
temp_check # Confirmed that missing values in each sales method and each nb_sold
has at least one non-missing value that correspond to that respective sales
method and nb_sold. Can proceed to impute missing values with mean of sales
method and nb_sold.
```

```
# Impute missing revenue values with mean calculated from each sales method and
each nb_sold
product_sales['revenue'] = product_sales.groupby(['sales_method', 'nb_sold'])
['revenue'].transform(lambda x: x.fillna(round(x.mean(), 2)))

# Check through the 'years_as_customer' column
print(product_sales['years_as_customer'].sort_values().unique()) # Print all
unique values in the 'years_as_customer' column
product_sales.loc[product_sales.years_as_customer > 2025 - 1984,
'years_as_customer'] = pd.NA # Replace out of bound years with NA
product_sales['years_as_customer'] =
product_sales['years_as_customer'].astype('Int64') # Cast the 'years_as_customer'
column as integer
print(round(product_sales['years_as_customer'],0).sort_values().unique()) # Print
the unique values in the 'years_as_customer' column to check the imputation is
successful

# Identify the unique values in the 'nb_site_visits' column
print(product_sales['nb_site_visits'].sort_values().unique())

# Identify the unique values in the 'state' column
print(product_sales['state'].sort_values().unique())
print(product_sales['state'].nunique())

product_sales.dtypes
```

Hidden output

```
state_sm = product_sales.groupby(['state', 'sales_method'])
['customer_id'].count().reset_index().rename(columns = {'customer_id': 'Count'})
sm = product_sales.groupby('state')
['customer_id'].count().reset_index().rename(columns = {'customer_id':
'Count_sm'})
state_sm = state_sm.merge(sm, how = 'inner', on = 'state')
state_sm['percent'] = round(state_sm['Count']/state_sm['Count_sm']*100,2)

total_rev_by_week = product_sales.groupby(['week', 'sales_method'])
['revenue'].sum().reset_index()
cust_count_by_week = product_sales.groupby(['week', 'sales_method'])
['customer_id'].count().reset_index()

sns.barplot(data = total_rev_by_week, x = 'week', y = 'revenue', hue =
'sales_method')
plt.title('Total Revenue Over Time for Each Sales Method', fontdict =
{'fontsize': 16}, weight = 'bold', loc = 'center')
plt.xlabel('Week', fontsize = 14, loc = 'center')
plt.ylabel('Total Revenue ($)', fontsize = 14, loc = 'center')
plt.legend(title = 'Sales Method')
plt.show()

sns.barplot(data = cust_count_by_week, x = 'week', y = 'customer_id', hue =
'sales_method')
plt.title('Number of Customers Over Time for Each Sales Method', fontdict =
{'fontsize': 16}, weight = 'bold', loc = 'center')
plt.xlabel('Week', fontsize = 14, loc = 'center')
plt.ylabel('Number of Customers', fontsize = 14, loc = 'center')
plt.legend(title = 'Sales Method')
plt.show()
```

Hidden output

```
p = sns.PairGrid(product_sales, hue = 'sales_method')
p.map(sns.scatterplot, alpha = 0.5)
plt.show()
```

Hidden output

```
# Total revenue?
total_rev_by_sm = product_sales.groupby('sales_method')
['revenue'].sum().reset_index().sort_values('sales_method', ascending = True)
total_rev = pd.DataFrame([['Total', product_sales['revenue'].sum()]], columns =
['sales_method', 'revenue'])

total_rev_by_sm = pd.concat([total_rev,total_rev_by_sm])

sns.barplot(data = total_rev_by_sm, x = 'sales_method', y = 'revenue')
plt.title('Total Revenue and Its Breakdown by Sales Method', fontdict =
{'fontsize': 16}, weight = 'bold', loc = 'center')
plt.xlabel('Sales Method', fontsize = 14, loc = 'center')
plt.yticks(np.arange(0, 1600000, step=200000), labels = np.arange(0, 1600000,
step=200000))
plt.ylabel('Total Revenue ($)', fontsize = 14, loc = 'center')
plt.show()

total_rev_by_sm
```

Hidden output

```
product_sales = product_sales.sort_values('sales_method', ascending = True)

# How many customers were there for each approach?
sns.countplot(data = product_sales, x = 'sales_method')
plt.title('Number of Customers for Each Sales Method', fontdict = {'fontsize':
16}, weight = 'bold', loc = 'center')
plt.xlabel('Sales Method', fontsize = 14, loc = 'center')
plt.ylabel('Count/Number of Customers', fontsize = 14, loc = 'center')
plt.show()

# What does the revenue looks like overall? And for each method?
all_rev_mean_label = 'Mean: $' + str(round(product_sales['revenue'].mean(),2))
all_rev_median_label = 'Median: $' +
str(round(product_sales['revenue'].quantile(0.5),2))

sns.histplot(data = product_sales, x = 'revenue', bins = 20)
plt.vlines(round(product_sales['revenue'].mean(),2), ymin = 0, ymax = 3500,
colors = 'red', linestyles = 'dashed', linewidths = 2, alpha = 0.7)
plt.text(round(product_sales['revenue'].mean(),2) + 7, 2800, all_rev_mean_label,
fontdict = {'fontsize': 12}, color = 'red')
plt.vlines(round(product_sales['revenue'].quantile(0.5),2), ymin = 0, ymax =
3500, colors = 'blue', linestyles = 'dashed', linewidths = 2, alpha = 0.7)
plt.text(round(product_sales['revenue'].quantile(0.5),2) - 55, 2800,
all_rev_median_label, fontdict = {'fontsize': 12}, color = 'blue')
plt.ylim((0, 3200))
plt.title('Distribution of Revenue', fontdict = {'fontsize': 16}, weight =
'bold', loc = 'center')
plt.xlabel('Individual Revenue ($)', fontsize = 14, loc = 'center')
plt.ylabel('Count/Number of Customers', fontsize = 14, loc = 'center')
plt.show()

sns.histplot(data = product_sales, x = 'revenue', hue = 'sales_method', bins =
20, alpha = 0.7)
plt.title('Distribution of Revenue By Sales Methods', fontdict = {'fontsize':
16}, weight = 'bold', loc = 'center')
plt.xlabel('Individual Revenue ($)', fontsize = 14, loc = 'center')
plt.ylabel('Count/Number of Customers', fontsize = 14, loc = 'center')
plt.legend(title = 'Sales Method', labels = ['Email + Call', 'Email', 'Call'])
plt.show()

sns.boxplot(data = product_sales, y = 'revenue', x = 'sales_method', hue =
'sales_method')
plt.title('Boxplots of Revenue By Sales Methods', fontdict = {'fontsize': 16},
weight = 'bold', loc = 'center')
plt.xlabel('Sales Method', fontsize = 14, loc = 'center')
plt.ylabel('Individual Revenue ($)', fontsize = 14, loc = 'center')
plt.show()

mean_rev_sm_week = product_sales.groupby(['sales_method', 'week'])
['revenue'].mean().round(2).reset_index().rename(columns = {'revenue':
'Mean_rev'})
```

```
median_rev_sm_week = product_sales.groupby(['sales_method', 'week'])
['revenue'].quantile(0.5).round(2).reset_index().rename(columns = {'revenue':
'Median_rev'})

mean_median_rev_sm_week = mean_rev_sm_week.merge(median_rev_sm_week, how =
'outer', on = ['sales_method', 'week'])
```

Hidden output

```
#sns.scatterplot(data = product_sales, x = 'week', y = 'revenue', hue =
'sales_method')
#plt.show()
print(product_sales.groupby('sales_method')[['nb_sold', 'revenue']].corr())
print(product_sales.groupby('sales_method')[['years_as_customer',
'revenue']].corr())
print(product_sales.groupby('sales_method')[['nb_site_visits',
'revenue']].corr())

state_cust_count = product_sales.pivot_table(values = 'customer_id', index =
'state', columns = 'sales_method', aggfunc = 'count')

res = stats.chi2_contingency(state_cust_count)

print(res[0:2])

state_rev_sum = product_sales.pivot_table(values = 'revenue', index = 'state',
columns = 'sales_method', aggfunc = 'sum')

state_rev_avg = product_sales.pivot_table(values = 'revenue', index =
'sales_method', columns = 'state', aggfunc = 'mean')

state_rev_call = product_sales[product_sales['sales_method'] ==
'Call'].pivot(columns = 'state', values = 'revenue')

state_rev_email = product_sales[product_sales['sales_method'] ==
'Email'].pivot(columns = 'state', values = 'revenue')

state_rev_email_call = product_sales[product_sales['sales_method'] == 'Email +
Call'].pivot(columns = 'state', values = 'revenue')

temp_list = []
for i in range(0, 50):
    temp_list.append(state_rev_call.iloc[:, i])
    temp_list[i] = temp_list[i].dropna()

print(stats.kruskal(temp_list[0], temp_list[1], temp_list[2], temp_list[3],
temp_list[4], temp_list[5], temp_list[6], temp_list[7], temp_list[8],
temp_list[9], temp_list[10], temp_list[11], temp_list[12], temp_list[13],
temp_list[14], temp_list[15], temp_list[16], temp_list[17], temp_list[18],
temp_list[19], temp_list[20], temp_list[21], temp_list[22], temp_list[23],
temp_list[24], temp_list[25], temp_list[26], temp_list[27], temp_list[28],
temp_list[29], temp_list[30], temp_list[31], temp_list[32], temp_list[33],
temp_list[34], temp_list[35], temp_list[36], temp_list[37], temp_list[38],
temp_list[39], temp_list[40], temp_list[41], temp_list[42], temp_list[43],
temp_list[44], temp_list[45], temp_list[46], temp_list[47], temp_list[48],
temp_list[49]))

temp_list = []
for i in range(0, 50):
    temp_list.append(state_rev_email.iloc[:, i])
```

```
    temp_list[i] = temp_list[i].dropna()

print(stats.kruskal(temp_list[0], temp_list[1], temp_list[2], temp_list[3],
temp_list[4], temp_list[5], temp_list[6], temp_list[7], temp_list[8],
temp_list[9], temp_list[10], temp_list[11], temp_list[12], temp_list[13],
temp_list[14], temp_list[15], temp_list[16], temp_list[17], temp_list[18],
temp_list[19], temp_list[20], temp_list[21], temp_list[22], temp_list[23],
temp_list[24], temp_list[25], temp_list[26], temp_list[27], temp_list[28],
temp_list[29], temp_list[30], temp_list[31], temp_list[32], temp_list[33],
temp_list[34], temp_list[35], temp_list[36], temp_list[37], temp_list[38],
temp_list[39], temp_list[40], temp_list[41], temp_list[42], temp_list[43],
temp_list[44], temp_list[45], temp_list[46], temp_list[47], temp_list[48],
temp_list[49]))

temp_list = []
for i in range(0, 50):
    temp_list.append(state_rev_email_call.iloc[:, i])
    temp_list[i] = temp_list[i].dropna()

print(stats.kruskal(temp_list[0], temp_list[1], temp_list[2], temp_list[3],
temp_list[4], temp_list[5], temp_list[6], temp_list[7], temp_list[8],
temp_list[9], temp_list[10], temp_list[11], temp_list[12], temp_list[13],
temp_list[14], temp_list[15], temp_list[16], temp_list[17], temp_list[18],
temp_list[19], temp_list[20], temp_list[21], temp_list[22], temp_list[23],
temp_list[24], temp_list[25], temp_list[26], temp_list[27], temp_list[28],
temp_list[29], temp_list[30], temp_list[31], temp_list[32], temp_list[33],
temp_list[34], temp_list[35], temp_list[36], temp_list[37], temp_list[38],
temp_list[39], temp_list[40], temp_list[41], temp_list[42], temp_list[43],
temp_list[44], temp_list[45], temp_list[46], temp_list[47], temp_list[48],
temp_list[49]))

sns.scatterplot(data = product_sales, x = 'nb_sold', y = 'revenue', hue =
'sales_method')
plt.title('Revenue vs Number of New Products Sold', fontdict = {'fontsize': 16},
weight = 'bold', loc = 'center')
plt.xlabel('Number of New Products Sold', fontsize = 14, loc = 'center')
plt.ylabel('Revenue ($)', fontsize = 14, loc = 'center')
plt.legend(title = 'Sales Method')
plt.show()

year_cust_rev = product_sales[product_sales['years_as_customer'].notna()]
year_cust_rev['years_as_customer'] =
year_cust_rev['years_as_customer'].astype('float64')

sns.scatterplot(data = year_cust_rev, x = 'years_as_customer', y = 'revenue', hue
= 'sales_method')
plt.title('Revenue vs Years of Customer', fontdict = {'fontsize': 16}, weight =
'bold', loc = 'center')
plt.xlabel('Years of Customer', fontsize = 14, loc = 'center')
plt.ylabel('Revenue ($)', fontsize = 14, loc = 'center')
plt.legend(title = 'Sales Method')
plt.show()
```

```
sns.scatterplot(data = product_sales, x = 'nb_site_visits', y = 'revenue', hue =
'sales_method')
plt.title('Revenue vs Number of Times Website Visited', fontdict = {'fontsize':
16}, weight = 'bold', loc = 'center')
plt.xlabel('Number of Times Website Visited', fontsize = 14, loc = 'center')
plt.ylabel('Revenue ($)', fontsize = 14, loc = 'center')
plt.legend(title = 'Sales Method')
plt.show()
```

Hidden output

```
# Was there any difference in revenue over time for each of the methods?
sns.lineplot(data = product_sales, x = 'week', y = 'revenue', hue =
'sales_method', estimator = np.mean)
plt.title('Average Revenue Over Time for Each Sales Method', fontdict =
{'fontsize': 16}, weight = 'bold', loc = 'center')
plt.xlabel('Week', fontsize = 14, loc = 'center')
plt.ylabel('Average Revenue ($)', fontsize = 14, loc = 'center')
plt.legend(title = 'Sales Method')
plt.show()

week1_avg_rev_sm_week = mean_median_rev_sm_week[mean_median_rev_sm_week['week']
== 1].drop(columns = ['Median_rev', 'week']).rename(columns = {'Mean_rev':
'week_1'})
week4_avg_rev_sm_week = mean_median_rev_sm_week[mean_median_rev_sm_week['week']
== 4].drop(columns = ['Median_rev', 'week']).rename(columns = {'Mean_rev':
'week_4'})
week6_avg_rev_sm_week = mean_median_rev_sm_week[mean_median_rev_sm_week['week']
== 6].drop(columns = ['Median_rev','week']).rename(columns = {'Mean_rev':
'week_6'})

diff_avg_rev_sm_week = week6_avg_rev_sm_week.merge(week4_avg_rev_sm_week, how =
'inner', on = 'sales_method').merge(week1_avg_rev_sm_week, how = 'inner', on =
'sales_method')

diff_avg_rev_sm_week['percent_diff_week6_1'] =
round((diff_avg_rev_sm_week['week_6'] -
diff_avg_rev_sm_week['week_1'])/diff_avg_rev_sm_week['week_1']*100,1)

diff_avg_rev_sm_week['percent_diff_week4_1'] =
round((diff_avg_rev_sm_week['week_4'] -
diff_avg_rev_sm_week['week_1'])/diff_avg_rev_sm_week['week_1']*100,1)

diff_avg_rev_sm_week
# Based on the data, which method would you recommend we continue to use? Some of
these methods take more time from the team so they may not be the best for us to
use if the results are similar
```

Hidden output