CERTIFICATION - DATA ANALYST - PRODUCT SALES

datalab

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..

✅ When you have finished...

- Publish your Workspace using the option on the left
- Check the published version of your report:
  - Can you see everything you want us to grade?
  - Are all the graphics visible?
- Review the grading rubric. Have you included everything that will be graded?
- Head back to the Certification Dashboard to submit your practical exam report and record your presentation

🚀 Sales Data Analysis Report

Objective:
Validate the dataset, analyze sales performance across different sales methods (`email`, `call`, `email+call`), define a business metric, and provide actionable recommendations.

Dataset Columns:

- `week`, `sales_method`, `customer_id`, `nb_sold`, `revenue`, `years_as_customer`, `nb_site_visits`, `state`

📋 Dataset Description

This dataset contains sales data for a new product launch, including customer and sales details. Below is a summary of each column:

Column Name	Data Type	Description
`week`	Numeric	Number of weeks since the product launch when the sale was made.
`sales_method`	Character	Sales method used: one of `email`, `call`, or `email+call`.
`customer_id`	Character	Unique identifier for each customer.
`nb_sold`	Numeric	Number of new products sold in that transaction.
`revenue`	Numeric	Revenue generated from the sales, rounded to 2 decimal places.
`years_as_customer`	Numeric	Number of years the customer has been buying from the company (founded in 1984).
`nb_site_visits`	Numeric	Number of times the customer visited the website in the last 6 months.
`state`	Character	Customer location where orders are shipped.

Import Libraries and Loading Dataset

```
#import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime as dt

#load data
sales = pd.read_csv("product_sales.csv")
```

Data Validation and Data Cleaning

🧹 Dirty Data Checks

We validated the dataset for:

- Missing values
- Duplicates
- Outliers and negative values
- Invalid or unexpected categories
- Incorrect data types
- Logical inconsistencies

Below are the checks and results.

```
sales.head()
```

	week	sales_...	customer_id	nb_sold	revenue	years_as_custo...	nb_site_vi...
0	2	Email	2e72d641-95ac-497b-bbf8-4861764a7097	10		0	
1	6	Email + Call	3998a98d-70f5-44f7-942e-789bb8ad2fe7	15	225.47	1	
2	5	Call	d1de9884-8059-4065-b10f-86eef57e4a44	11	52.55	6	
3	4	Email	78aa75a4-ffeb-4817-b1d0-2f030783c5d7	11		3	
4	3	Email	10e6d446-10a5-42e5-8210-1b5438f70922	9	90.49	0	

Rows: 5 Expand

```
#Data Summary
sales.info()
sales.describe()
```

```
<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
```

ind...	week	nb_sold	revenue	years_as_customer	nb_si...
count	15000	15000	13926	15000	
mean	3.0982666667	10.0846666667	93.9349425535	4.9659333333	
std	1.6564198071	1.8122133327	47.4353122457	5.0449515589	
min	1	7	32.54	0	
25%	2	9	52.47	1	
50%	3	10	89.5	3	
75%	5	11	107.3275	7	
max	6	16	238.32	63	

Rows: 8 Expand

```
sales.describe(include = "O")
```

index	sales_method	customer_id	st
count	15000	15000	15
unique	5	15000	50
top	Email	2e72d641-95ac-497b-bbf8-4861764a7097	Cc
freq	7456	1	18

Rows: 4 Expand

-Data types are correct.

-Revenue has missing values.

-years_as_customer must not exceed start year of company(1984).

```
sales.sales_method.unique()
```

```
array(['Email', 'Email + Call', 'Call', 'em + call', 'email'],
      dtype=object)
```

```
#replacing inconsistant data
sales["sales_method"] = sales["sales_method"].replace("em + call", "Email + Call").replace("email", "Email")
sales.sales_method.unique()
```

```
array(['Email', 'Email + Call', 'Call'], dtype=object)
```

```
#check revenue column
sales["revenue"].isna().sum()
```

```
1074
```

```
# Revenue stats across sales group and nb of sold
revenue_stats = (
    sales.groupby(["sales_method", "nb_sold"])["revenue"]
    .agg(
        mean='mean',
        median='median',
        min='min',
        max='max',
        q25=lambda x: x.quantile(0.25),
        q75=lambda x: x.quantile(0.75)
    )
    .round(2)
)
revenue_stats
```

sales...							
Call	7	35.09	35.06	32.54	37.48	34.28	35.84
Call	8	41.04	41.23	37.5	42.49	40.45	41.87
Call	9	43.87	43.68	42.51	47.5	43.05	44.5
Call	10	51.18	51.35	47.5	52.5	50.5	51.91
Call	11	53.73	53.47	52.51	57.47	52.95	54.19
Call	12	57.78	57.77	57.56	58.01	57.7	57.85
Call	13	65.75	65.74	63.54	67.5	65.03	66.42
Call	14	68.62	68.32	67.52	71.36	67.9	68.93
Email	8	83.06	83.26	78.83	85	82.17	84.2
Email	9	89.6	89.32	85	95	87.2	91.92
Email	10	99.76	99.43	95	104.99	97.07	102.5
Email	11	108.72	108.35	105.01	114.98	106.51	110.44
Email	12	118.64	117.86	115	124.95	116.36	120.42
Email	13	129.7	129.64	125.04	134.97	127.26	131.99
Email	14	137.67	137.14	135.1	144.01	136.1	139.15
Email	15	146.61	145.71	145.15	148.97	145.43	147.34

Rows: 24 Expand

-Mean and Median of Revenue are mostly nearly the same across sales_method and nb_sold.

```
#filling missing value with group means
median_gr = sales.groupby(["sales_method","nb_sold"])["revenue"].transform('median')
sales["revenue"] = sales['revenue'].fillna(median_gr).round(2)
sales["revenue"].describe()
```

	revenue
count	15000
mean	95.711558
std	48.3844508778
min	32.54
25%	52.76
50%	90
75%	108.35
max	238.32

Rows: 8 Expand

After imputing missing values, overall mean value change from about 93.9 to 95.7. Median form 89.5 to 90.

This indicates that the imputed values were generally higher than the original dataset's average, likely because higher-revenue transactions were missing.

```
#tenure year validation
company_started = 1984
max_years_tenure = dt.now().year - company_started
sales['years_as_customer'] = sales['years_as_customer'].where(sales['years_as_customer'] < max_years_tenure, np.nan)

print(sales['years_as_customer'].describe())
print(sales['years_as_customer'].isna().sum())
```

```
count    14998.000000
mean         4.959261
std          5.011237
min          0.000000
25%          1.000000
50%          3.000000
75%          7.000000
max         39.000000
Name: years_as_customer, dtype: float64
2
```

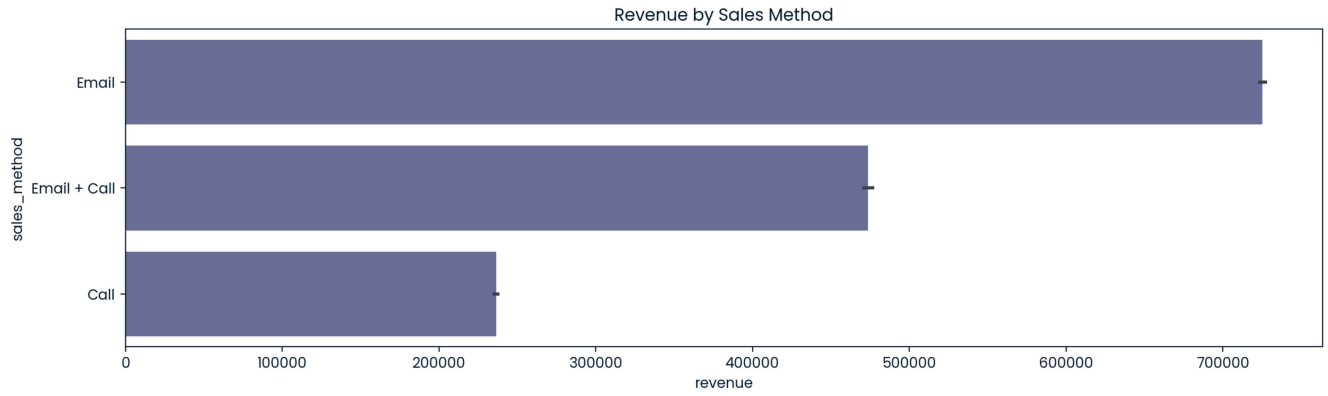
EDA Analysis

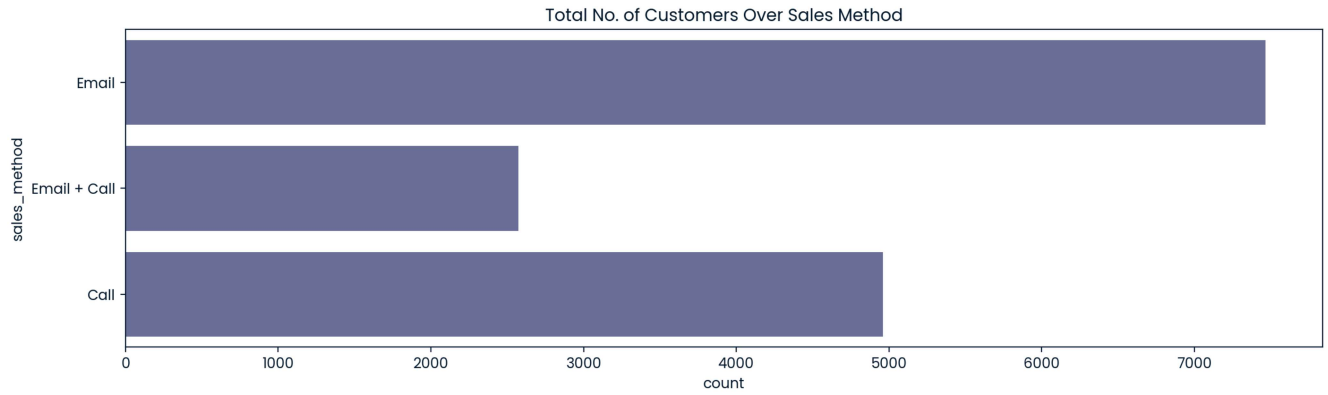
```
#Sales summary for each sales approach
sales_summary = sales.groupby(["sales_method"]).agg(
    total_revenue = ("revenue", 'sum'),
    num_customers = ("customer_id",'count'),
    revenue_per_customer  = ("revenue", 'mean'),
    total_units_sold = ("nb_sold", 'sum'),
    avg_site_visits = ("nb_site_visits", 'mean')
).round(2).reset_index()
sales_summary
```

	sales...	total_r...	num_c...	revenue_per_custo...	total_units_...	avg_site_...
0	Call	236395.16	4962	47.64	47187	24.42
1	Email	725440.63	7466	97.17	72639	24.75
2	Email + Call	473837.58	2572	184.23	31444	26.77

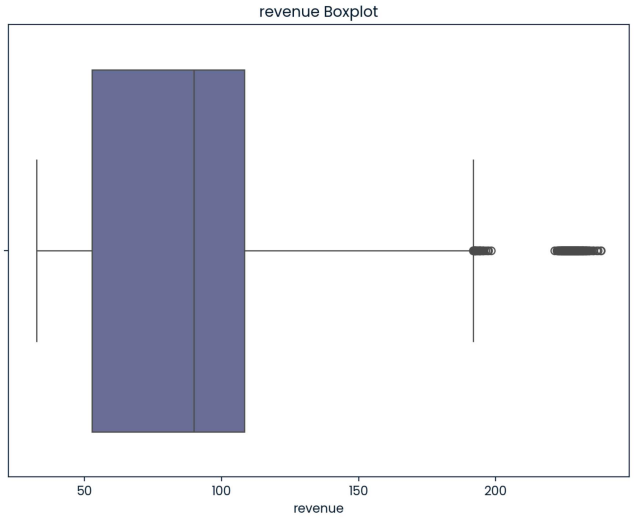
Rows: 3 Expand

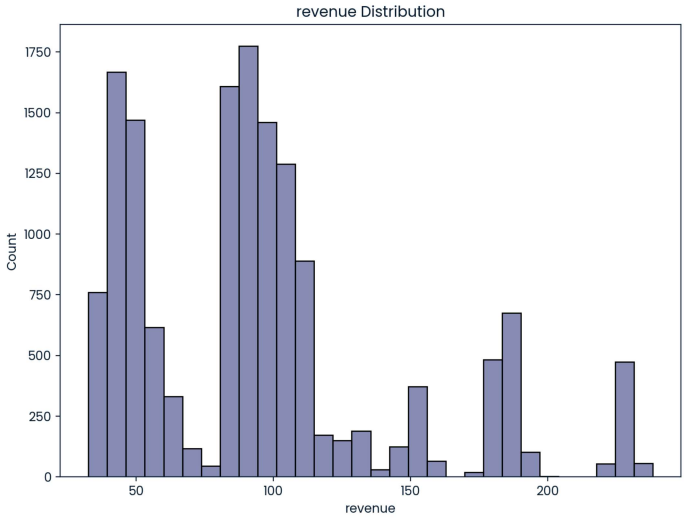
```
# total revenue by sales method
plt.figure(figsize=(15,4))
sns.barplot(sales,
            x = "revenue",
            y = "sales_method",
            estimator = 'sum'
           )
plt.title("Revenue by Sales Method")
plt.show()

#total num of customers by sales method
plt.figure(figsize=(15,4))
sns.countplot(sales,
            y = "sales_method")
plt.title("Total No. of Customers Over Sales Method")
plt.show()
```

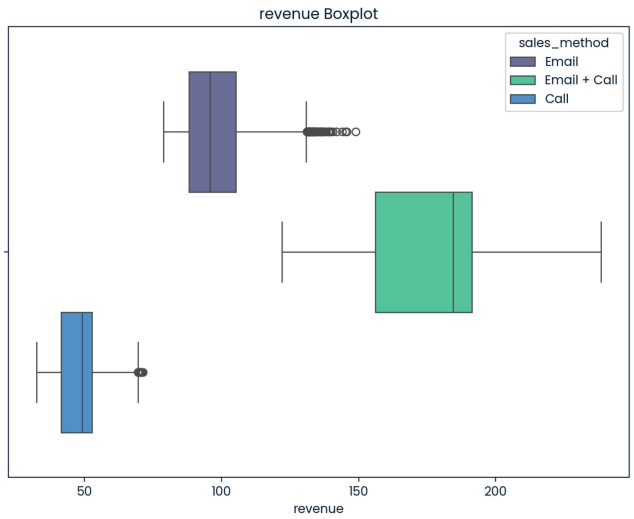


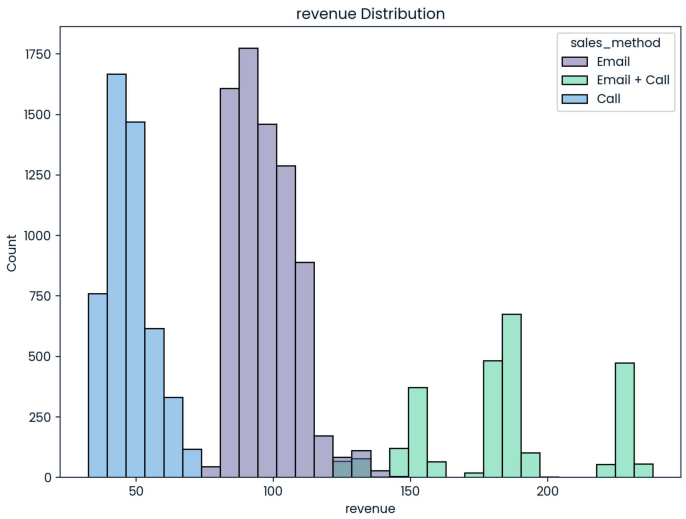


```
#What does the spread of the revenue look like overall? And for each method?
def box_hist(df, col, hue = None):
    fig, axes = plt.subplots(1, 2, figsize=(15, 6))

    # Boxplot
    sns.boxplot(data=df, x=col, ax=axes[0], hue = hue)
    axes[0].set_title(f"{col} Boxplot")
    axes[0].set_xlabel(col)

    # Histplot
    sns.histplot(data=df, x=col, bins=30, ax=axes[1], hue = hue)
    axes[1].set_title(f"{col} Distribution")
    axes[1].set_xlabel(col)

    plt.tight_layout()
    plt.show()

box_hist(sales,"revenue")
box_hist(sales, "revenue", "sales_method")
```

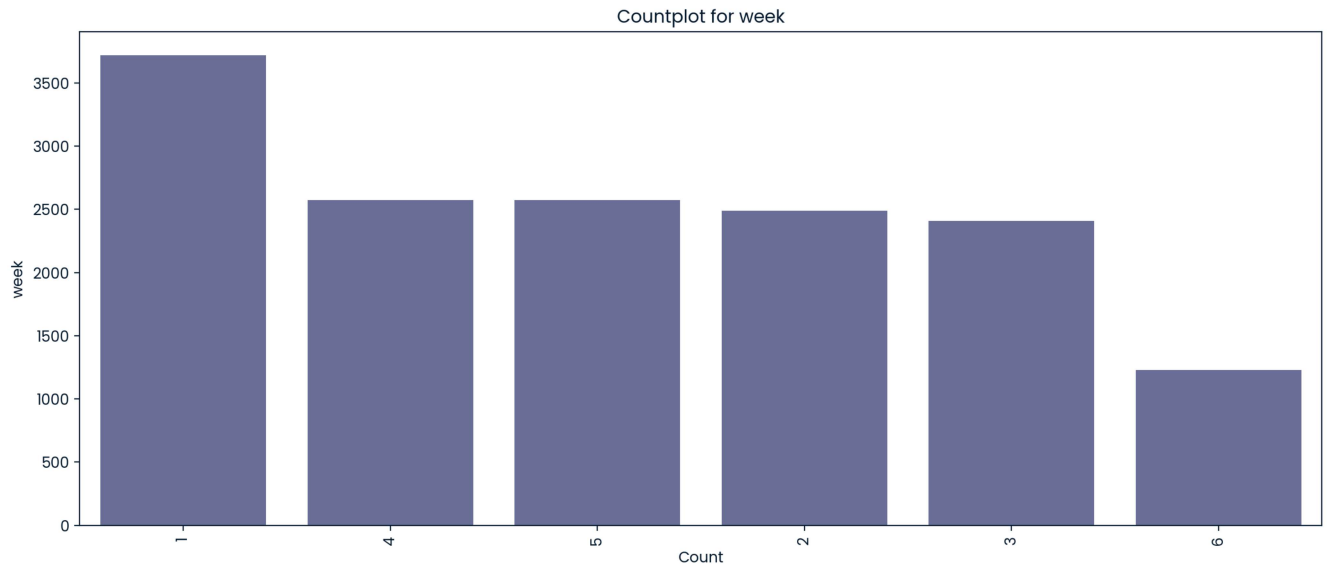




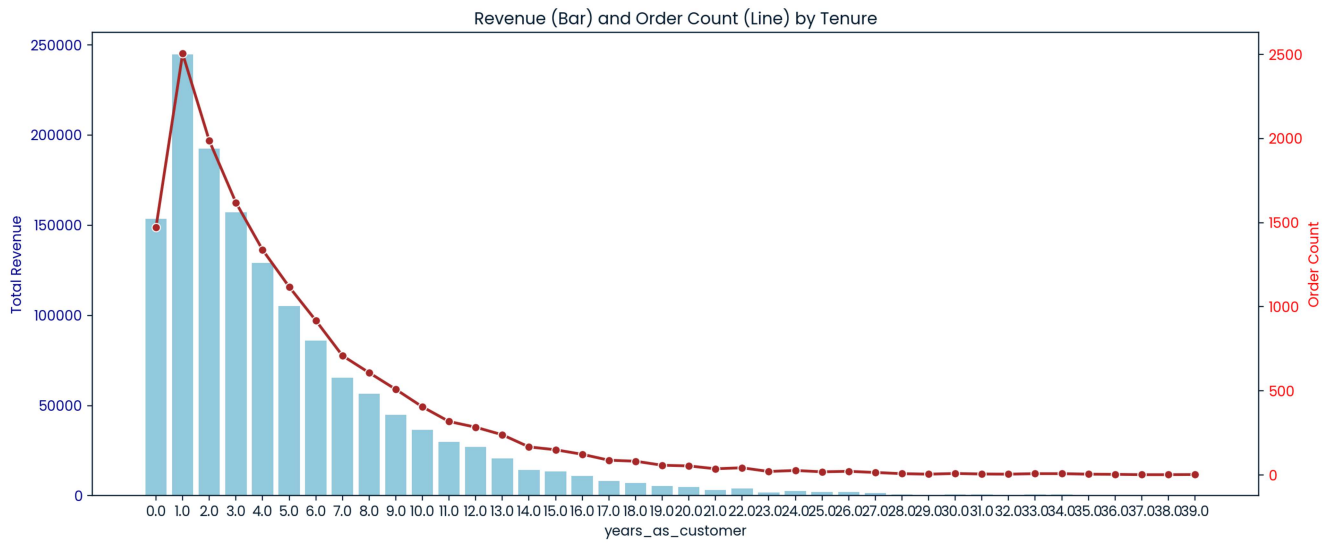




```
#the countplot of years_as_customers and week
def countplot(df, col):
    plt.figure(figsize=(15, 6))
    # Countplot with sorted order (most frequent first)
    order = sales[col].value_counts().index
    sns.countplot(data=df, x=col, order=order)
    plt.title(f"Countplot for {col}")
    plt.xlabel("Count")
    plt.ylabel(col)
    plt.xticks(rotation = 90)
    plt.show()

countplot(sales, "week")
```



```
# Aggregate the data first
# Revenue by tenure
revenue_by_tenure = (
    sales.groupby("years_as_customer")["revenue"]
    .sum()
    .reset_index()
    .sort_values("years_as_customer")
)

# Order Count by tenure
order_count_by_tenure = (
    sales.groupby("years_as_customer")["revenue"]
    .count()
    .reset_index()
    .sort_values("years_as_customer")
    .rename(columns={"revenue": "order_count"})
)

# Merge to align x-axis if needed
combined = revenue_by_tenure.merge(order_count_by_tenure, on="years_as_customer")

# Create the combined plot
fig, ax1 = plt.subplots(figsize=(15, 6))

# Barplot: Revenue
sns.barplot(
    data=combined,
    x="years_as_customer",
    y="revenue",
    color="skyblue",
    ax=ax1
)
ax1.set_ylabel("Total Revenue", color="darkblue")
ax1.tick_params(axis='y', labelcolor="darkblue")

# 2nd y-axis: Order Count Line
ax2 = ax1.twinx()
sns.lineplot(
    data=combined,
    x="years_as_customer",
    y="order_count",
    color="brown",
    marker="o",
    linewidth=2,
    ax=ax2
)
ax2.set_ylabel("Order Count", color="red")
ax2.tick_params(axis='y', labelcolor="red")

# Title
plt.title("Revenue (Bar) and Order Count (Line) by Tenure")

plt.show()
```



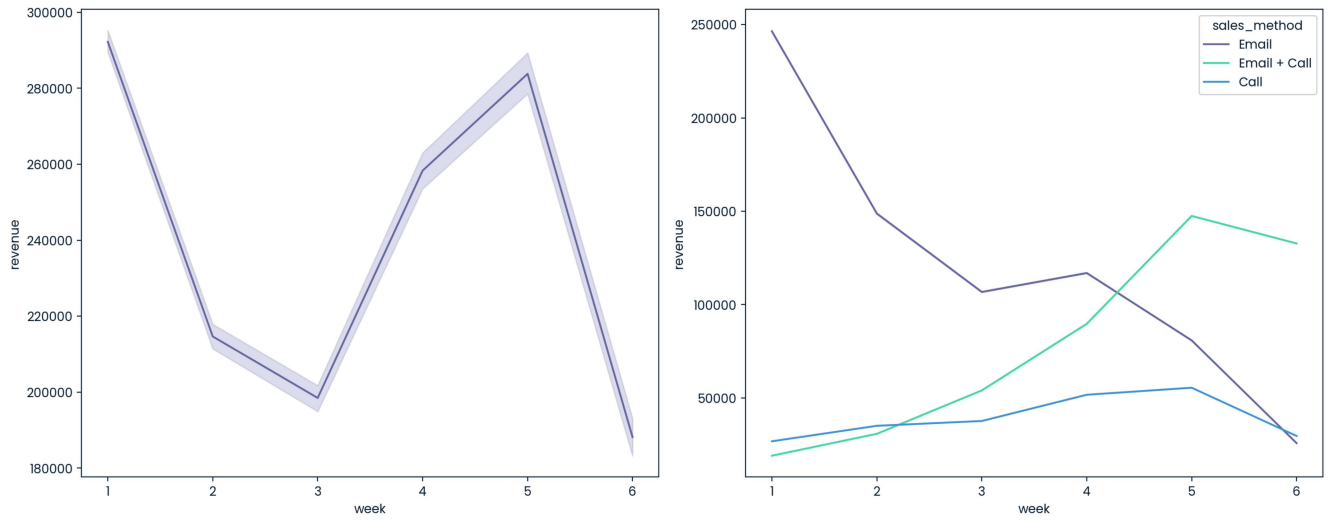
Older Customers bought less because of needing them

```
sales.corr()
```

index	week	nb_sold	revenue	years_as_cus...	nb_site_...
week	1	0.8098874409	0.3745641292	-0.082688297	0.4190377111
nb_sold	0.8098874409	1	0.7088664702	-0.0997278646	0.4907181923
revenue	0.3745641292	0.7088664702	1	-0.0671329822	0.3309390893
years_as_customer	-0.082688297	-0.0997278646	-0.0671329822	1	-0.049295465
nb_site_visits	0.4190377111	0.4907181923	0.3309390893	-0.049295465	1

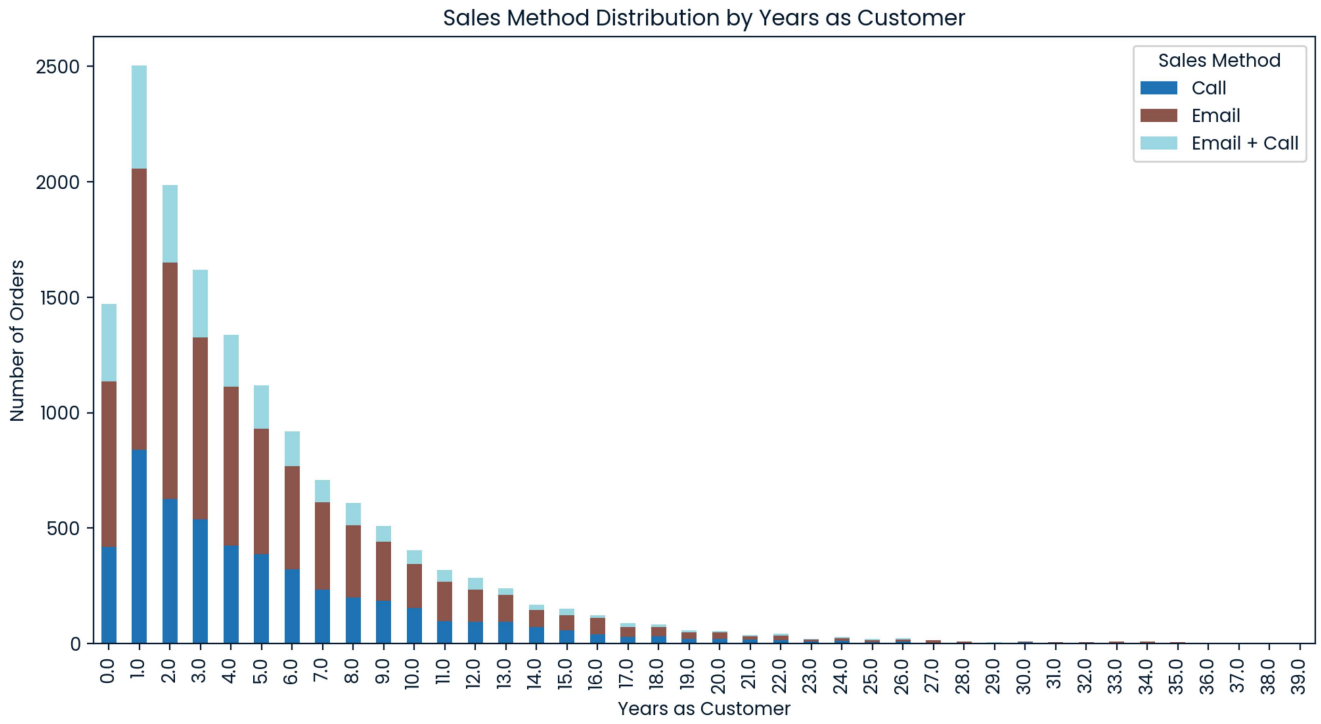
Rows: 5 Expand

```
# Was there any difference in revenue over time for each of the methods?
fig,axes = plt.subplots(1,2, figsize =(15,6))
sns.lineplot(sales,
             x= "week",
             y= "revenue",
             estimator = "sum",
            ax = axes[0])
sns.lineplot(sales,
             x= "week",
             y= "revenue",
             hue = "sales_method",
             estimator = "sum",
            ax = axes[1])
plt.tight_layout()
plt.show()
```



```
#year_as_customer and sales_method
# Make a frequency table
stacked = pd.crosstab(sales['years_as_customer'], sales['sales_method'])
print(stacked.head())

stacked.plot(
    kind='bar',
    stacked=True,
    figsize=(12,6),
    colormap='tab20'  # choose a nice color map
)

plt.title("Sales Method Distribution by Years as Customer")
plt.xlabel("Years as Customer")
plt.ylabel("Number of Orders")
plt.legend(title="Sales Method")
plt.show()
```

sales_method	Call	Email	Email + Call
years_as_customer			
0.0	417	718	336
1.0	839	1217	448
2.0	627	1024	336
3.0	537	789	293
4.0	425	686	227



3. Definition of Metric for the Business

Metric: Revenue per Hour of Sales Effort (RPH) — total revenue generated divided by total sales team hours spent.

How to Monitor Revenue per Hour of Sales Effort (RPH)

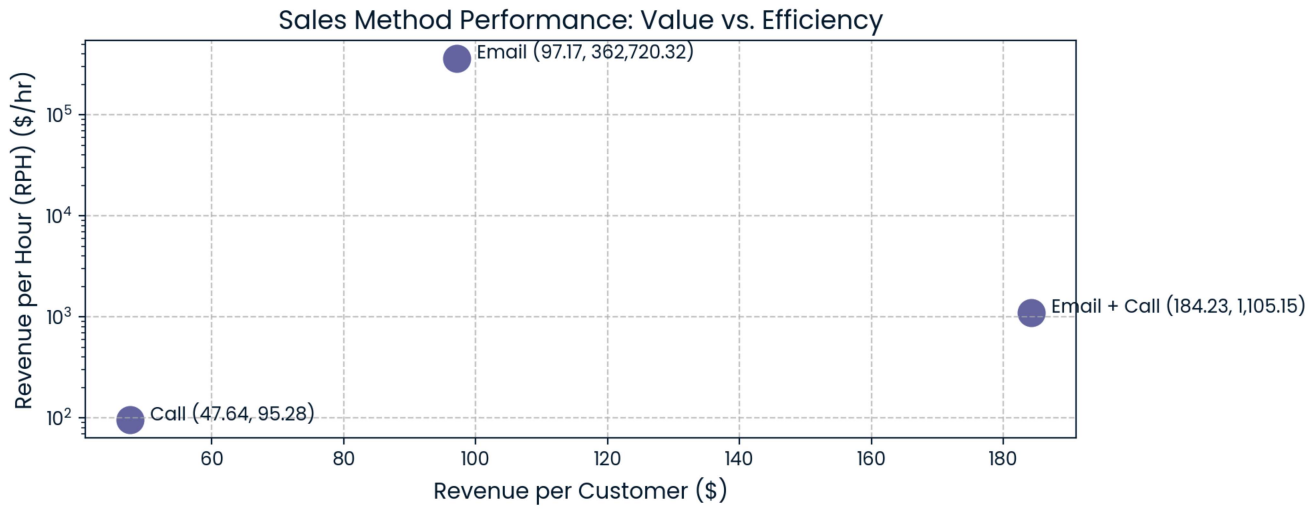
To monitor this metric, the business should:

- Track total revenue by sales method on a weekly basis.
- Estimate the total hours spent per sales method based on time per customer.
- Calculate RPH = Total Revenue ÷ Total Hours spent.
- Compare the RPH for each sales method weekly to allocate resources effectively.

```
sales_metrics = sales_summary[['sales_method', 'total_revenue', 'num_customers']]
# Time spent assumptions (in hours)
# Email setup total time (approximate)
email_time_total = 2

# Calculate total hours per method
sales_metrics['time_per_customer_hr'] = sales_metrics['sales_method'].map({
    'Email': email_time_total / sales_metrics.loc[sales_metrics['sales_method'] == 'Email', 'num_customers'].values[0],  #
distribute setup time per customer
    'Call': 0.5,  # 30 mins = 0.5 hours
    'Email + Call': 0.1667  # 10 mins = 1/6 hour
})

sales_metrics['total_hours'] = sales_metrics['time_per_customer_hr'] * sales_metrics['num_customers']

# Calculate Revenue per Hour (RPH)
sales_metrics['RPH'] = (sales_metrics['total_revenue'] / sales_metrics['total_hours']).round(2)

print(sales_metrics[['sales_method', 'total_revenue', 'total_hours', 'RPH']].sort_values('RPH', ascending = True))
```

```
   sales_method  total_revenue  total_hours        RPH
0          Call      236395.16    2481.0000      95.28
2  Email + Call      473837.58     428.7524    1105.15
1         Email      725440.63       2.0000  362720.32
```

```python
# Sales Method Performance by the matrix of RPH and Revenue per Customer
metrics_df = pd.merge(sales_metrics, sales_summary, on = 'sales_method')[['sales_method', 'revenue_per_customer','RPH']]
plt.figure(figsize=(10,4))
plt.scatter(
    metrics_df['revenue_per_customer'],
    metrics_df['RPH'],
    s=200
)

# Add (x, y) labels for each point
for i in range(len(metrics_df)):
    x = metrics_df['revenue_per_customer'][i]
    y = metrics_df['RPH'][i]
    label = f"{metrics_df['sales_method'][i]} ({x:.2f}, {y:,.2f})"

    plt.text(
        x + 3,     # shift label right
        y ,
        label,
        fontsize=10
    )

plt.xlabel('Revenue per Customer ($)', fontsize=12)
plt.ylabel('Revenue per Hour (RPH) ($/hr)', fontsize=12)
plt.title('Sales Method Performance: Value vs. Efficiency', fontsize=14)
plt.grid(True, linestyle='--', alpha=0.7)

# Log scale for RPH :
plt.yscale('log')

plt.tight_layout()
plt.show()
```




```
Python...
```