In [ ]:
```
import pandas as pd
```

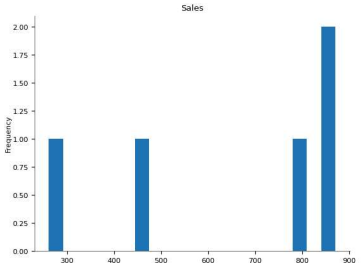
In [ ]:
```
df = pd.read_csv('sales_data.csv')
```

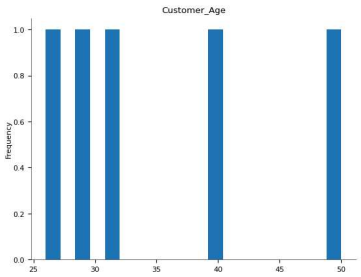
In [ ]:
```
df.head()
```

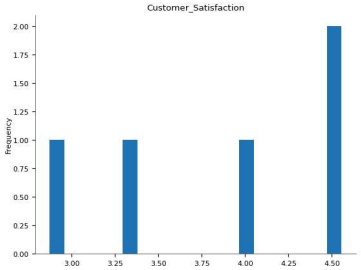
Out[ ]:

	Date	Product	Region	Sales	Customer_Age	Customer_Gender	Customer_Satisfacti
0	2022-01-01	Widget C	South	786	26	Male	2.8744
1	2022-01-02	Widget D	East	850	29	Male	3.3652
2	2022-01-03	Widget A	North	871	40	Female	4.5473
3	2022-01-04	Widget C	South	464	31	Male	4.5554
4	2022-01-05	Widget C	South	262	50	Female	3.9829

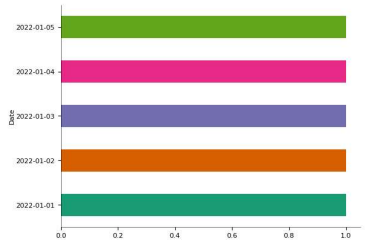
Distributions

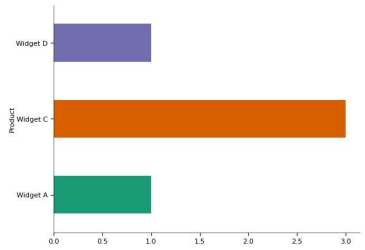


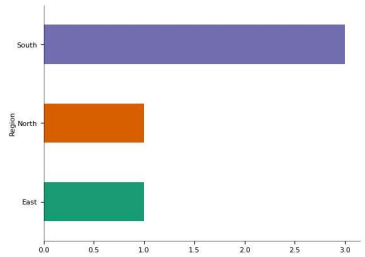


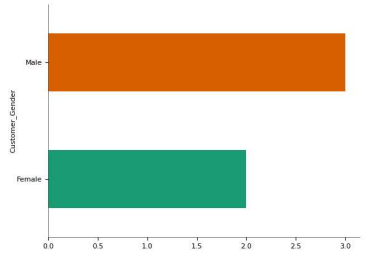


Categorical distributions

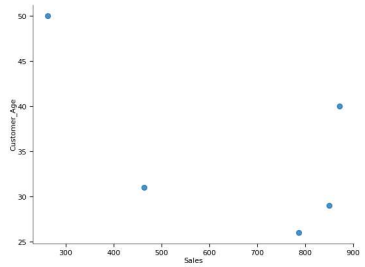


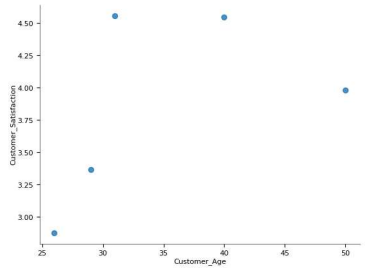




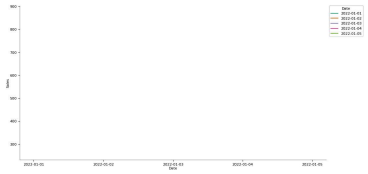


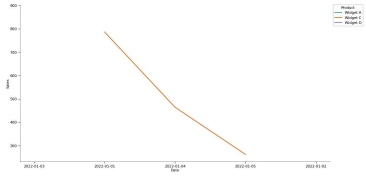
2-d distributions

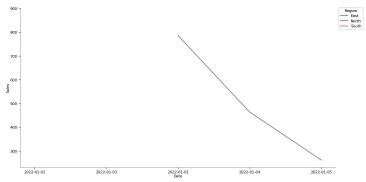


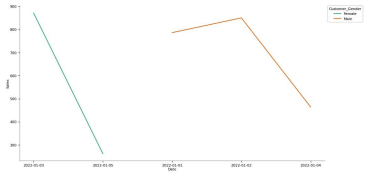


Time series

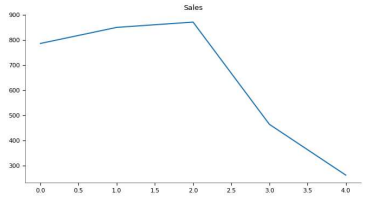


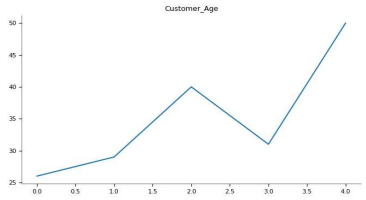


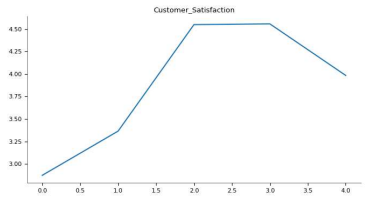




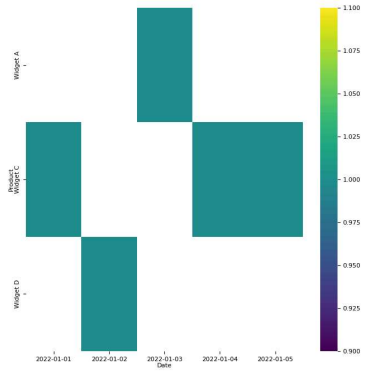
Values

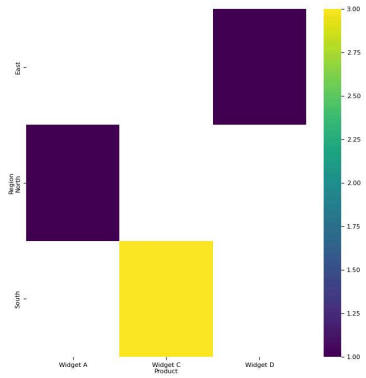


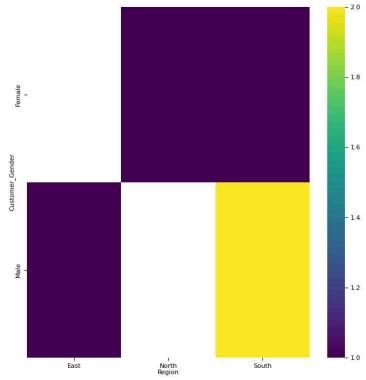




2-d categorical distributions

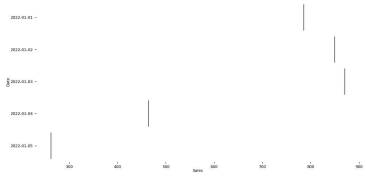




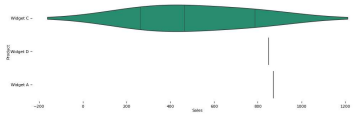


Faceted distributions

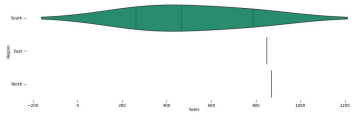
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.
```



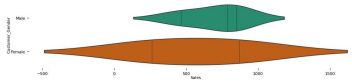
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.
```



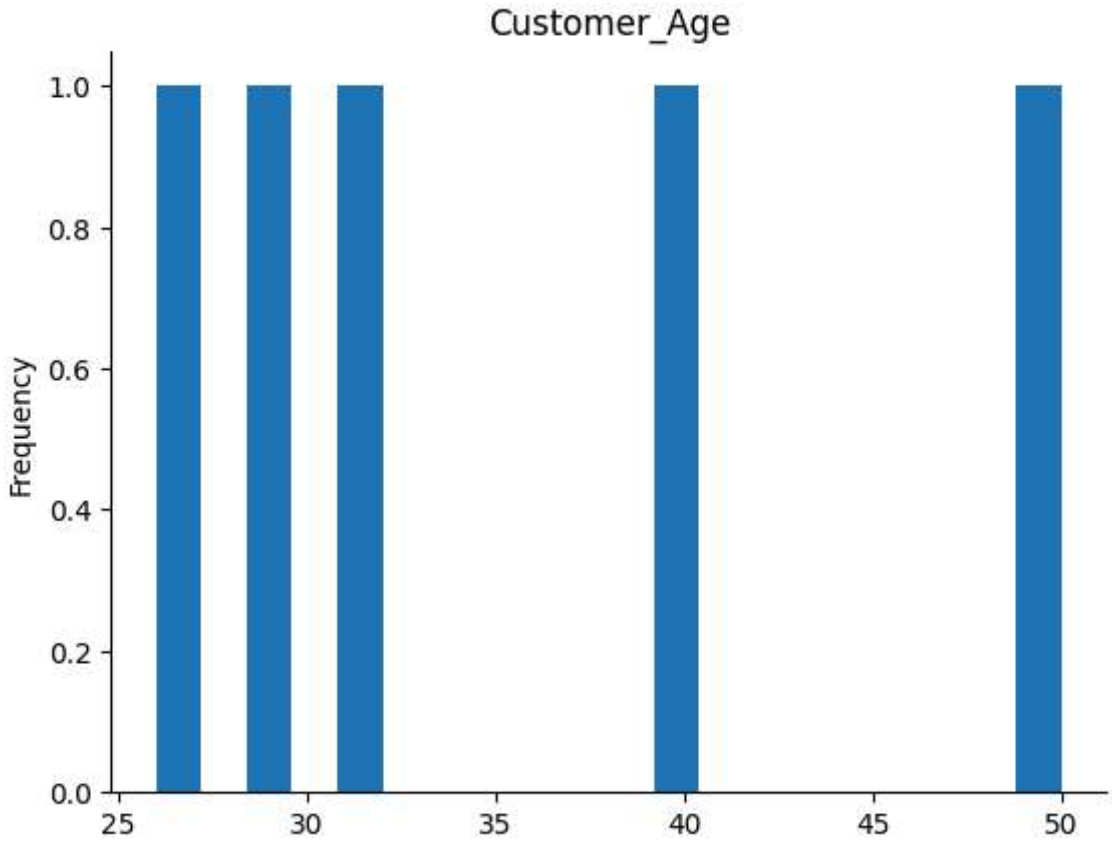
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.
```



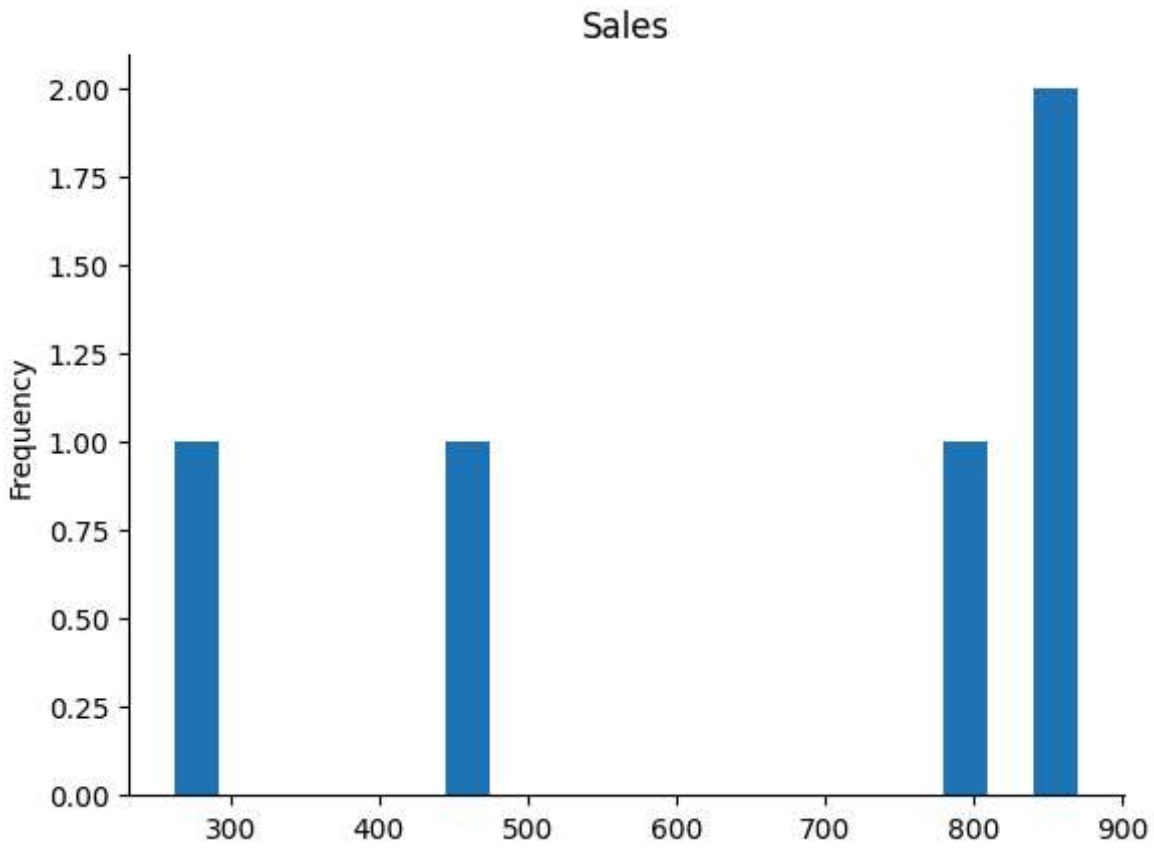
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effe
ct.
```



In [ ]:
```
from matplotlib import pyplot as plt
_df_1['Customer_Age'].plot(kind='hist', bins=20, title='Customer_Age')
plt.gca().spines[['top', 'right',]].set_visible(False)
```



In [ ]:
```
from matplotlib import pyplot as plt
_df_0['Sales'].plot(kind='hist', bins=20, title='Sales')
plt.gca().spines[['top', 'right',]].set_visible(False)
```



In [ ]:
```
df.describe(), df.dtypes
```

Out[ ]:
```
(             Sales  Customer_Age  Customer_Satisfaction
 count  2500.000000   2500.000000            2500.000000
 mean    553.288000     43.332800               3.025869
 std     260.101758     14.846758               1.156981
 min     100.000000     18.000000               1.005422
 25%     324.750000     31.000000               2.056014
 50%     552.500000     43.000000               3.049480
 75%     779.000000     56.000000               4.042481
 max     999.000000     69.000000               4.999006,
 Date                      object
 Product                   object
 Region                    object
 Sales                      int64
 Customer_Age               int64
 Customer_Gender           object
 Customer_Satisfaction    float64
 dtype: object)
```

In [ ]:
```
df['Date'].dtype
```

Out[ ]:
```
dtype('O')
```

In [ ]:
```
df['Date'] = pd.to_datetime(df['Date'])
```

In [ ]:
```
df['Date'].dtype
```

Out[ ]:
```
dtype('<M8[ns]')
```

In [ ]:
```
df['Month'] = df['Date'].dt.to_period('M')
```

In [ ]:
```
monthly_sales = df.groupby(['Month','Region'],observed=False)['Sales'].sum().sor
```

In [ ]:
```
monthly_sales
```

Out[ ]:

		Sales
Month	**Region**	
2024-05	**South**	8862
2026-04	**North**	8725
2027-11	**North**	8593
2024-02	**West**	8278
2026-05	**South**	8174
...	...	...
2028-11	**East**	878
2026-05	**East**	849
2024-09	**West**	699
2028-11	**West**	614
2024-11	**East**	347

331 rows × 1 columns

dtype: int64

In [ ]:
```
monthly_sales
```

Out[ ]:

Month	Sales
2028-04	20387
2025-02	20168
2024-05	19740
2022-12	19267
2028-08	19222
...	...
2027-10	14049
2026-09	13827
2027-02	13717
2022-04	13376
2028-11	3212

83 rows × 1 columns

dtype: int64

In [ ]:

```
def generate_advanced_data_summary(df):
    # Ensure 'Date' is in datetime format
    df['Date'] = pd.to_datetime(df['Date'])

    # Sales Analysis
    total_sales = df['Sales'].sum()
    avg_sale = df['Sales'].mean()
    median_sale = df['Sales'].median()
    sales_std = df['Sales'].std()

    # Time-based Analysis
    df['Month'] = df['Date'].dt.to_period('M')
    monthly_sales = df.groupby('Month', observed=False)['Sales'].sum().sort_valu
    best_month = monthly_sales.index[0]
    worst_month = monthly_sales.index[-1]

    # Product Analysis
    product_sales = df.groupby('Product', observed=False)['Sales'].agg(['sum', '
    top_product = product_sales['sum'].idxmax()
    most_sold_product = product_sales['count'].idxmax()

    # Regional Analysis
    region_sales = df.groupby('Region', observed=False)['Sales'].sum().sort_valu
    best_region = region_sales.index[0]
    worst_region = region_sales.index[-1]

    # Customer Analysis
    avg_satisfaction = df['Customer_Satisfaction'].mean()
    satisfaction_std = df['Customer_Satisfaction'].std()
```

```
    age_bins = [0, 25, 35, 45, 55, 100]
    age_labels = ['18-25', '26-35', '36-45', '46-55', '55+']
    df['Age_Group'] = pd.cut(df['Customer_Age'], bins=age_bins, labels=age_label
    age_group_sales = df.groupby('Age_Group', observed=False)['Sales'].mean().sc
    best_age_group = age_group_sales.index[0]

    # Gender Analysis
    gender_sales = df.groupby('Customer_Gender', observed=False)['Sales'].mean()

    summary = f"""
    Advanced Sales Data Summary:

    Overall Sales Metrics:
    - Total Sales: ${total_sales:,.2f}
    - Average Sale: ${avg_sale:.2f}
    - Median Sale: ${median_sale:.2f}
    - Sales Standard Deviation: ${sales_std:.2f}

    Time-based Analysis:
    - Best Performing Month: {best_month}
    - Worst Performing Month: {worst_month}

    Product Analysis:
    - Top Selling Product (by value): {top_product}
    - Most Frequently Sold Product: {most_sold_product}

    Regional Performance:
    - Best Performing Region: {best_region}
    - Worst Performing Region: {worst_region}

    Customer Insights:
    - Average Customer Satisfaction: {avg_satisfaction:.2f}/5
    - Customer Satisfaction Standard Deviation: {satisfaction_std:.2f}
    - Best Performing Age Group: {best_age_group}
    - Gender-based Average Sales: Male=${gender_sales['Male']:.2f}, Female=${gen


    Key Observations:
    1. The sales data shows significant variability with a standard deviation of
    2. The {best_age_group} age group shows the highest average sales.
    3. Regional performance varies significantly, with {best_region} outperformi
    4. The most valuable product ({top_product}) differs from the most frequentl
    """

    return summary
```

In [ ]:
```
advanced_summary = generate_advanced_data_summary(df)
```

In [ ]:
```
print(advanced_summary)
```

```
Advanced Sales Data Summary:

Overall Sales Metrics:
- Total Sales: $1,383,220.00
- Average Sale: $553.29
- Median Sale: $552.50
- Sales Standard Deviation: $260.10

Time-based Analysis:
- Best Performing Month: 2028-04
- Worst Performing Month: 2028-11

Product Analysis:
- Top Selling Product (by value): Widget A
- Most Frequently Sold Product: Widget A

Regional Performance:
- Best Performing Region: West
- Worst Performing Region: East

Customer Insights:
- Average Customer Satisfaction: 3.03/5
- Customer Satisfaction Standard Deviation: 1.16
- Best Performing Age Group: 18-25
- Gender-based Average Sales: Male=$547.56, Female=$558.96


Key Observations:
1. The sales data shows significant variability with a standard deviation of $260.10.
2. The 18-25 age group shows the highest average sales.
3. Regional performance varies significantly, with West outperforming East.
4. The most valuable product (Widget A) differs from the most frequently sold product (Widget A), suggesting potential for targeted marketing strategies.
```