MAHATMA EDUCATION SOCIETY’S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(Autonomous)

NEW PANVEL

PROJECT REPORT ON

“DATA SCIENCE”

IN PARTIAL FULFILMENT OF

BACHELOR OF SCIENCE IN

INFORMATION TECHNOLOGY

SEMESTER IV – 2023-24

PROJECT GUIDE

Name : Pradhnya Maam

SUBMITTED BY : Vinish Ananta Mhatre

ROLL NO : 6242

Name - Vinish Ananta Mhatre

Class - BSc.IT - C (sy)

Roll no. - 6242

Subject - Data Science

Topic - Exploratory Data Analysis of Sales Data

Introduction

In today's data-driven world, understanding the underlying patterns and trends in sales data is crucial for businesses to make informed decisions and drive growth. Exploratory Data Analysis (EDA) serves as the foundational step in the data analysis process, providing insights into the structure and characteristics of the dataset.

Project Overview:

In this project, we conduct an exploratory analysis of sales data to gain a comprehensive understanding of various aspects such as regional distribution, item types, sales channels, and order priorities. The dataset includes information such as region, country, item type, sales channel, order priority, order date, order ID, ship date, units sold, unit price, unit cost, total revenue, total cost, and total profit.

Objectives:

Explore the distribution of sales across different regions and countries.

Analyze the types of items sold and their frequency.

Examine the distribution of sales channels and order priorities.

Investigate the trends and patterns in sales over time.

Identify correlations between numerical variables such as units sold, total revenue, total cost, and total profit.

Visualize the data using various plots and charts to facilitate interpretation.

Methodology:

Data Loading: Load the sales data from the provided CSV file into a pandas DataFrame.

Data Cleaning: Perform data cleaning steps such as handling missing values, converting data types, and removing duplicates.

Exploratory Data Analysis:

Distribution Analysis: Explore the distribution of sales across different regions, countries, item types, sales channels, and order priorities using bar charts,heatmap,scatter plots and pie charts.

Time Series Analysis: Analyze the trends and patterns in sales over time using line plots and time series decomposition techniques.

Correlation Analysis: Investigate correlations between numerical variables using correlation matrices and heatmaps.

Data Visualization: Visualize the findings using various plots such as histograms, box plots, scatter plots, and pair plots to gain insights into the dataset.

Conclusion: Summarize the key findings from the exploratory analysis and highlight actionable insights for stakeholders.

Conclusion:

Exploratory Data Analysis serves as a crucial step in uncovering hidden patterns and insights from sales data. By leveraging visualization techniques and statistical analysis, businesses can make data-driven decisions to optimize their sales strategies, improve operational efficiency, and drive overall growth.

- **Data Preparation and Pre-Processing**

```
[1] import matplotlib.pyplot as plt
    import seaborn as sns
    import pandas as pd
    import numpy as np

[2] df=pd.read_csv("/content/1000 Sales Records.csv")
```

- **Data Preview**

```
[5] df.describe()
```

	Order ID	Units Sold	Unit Price	Unit Cost	Total Revenue	Total Cost	Total Profit
count	1.000000e+03	1000.000000	1000.00000	1000.000000	1.000000e+03	1.000000e+03	1.000000e+03
mean	5.496813e+08	5053.988000	262.10684	184.965110	1.327322e+06	9.361192e+05	3.912026e+05
std	2.571334e+08	2901.375317	216.02106	175.289311	1.486515e+06	1.162571e+06	3.836402e+05
min	1.029280e+08	13.000000	9.33000	6.920000	2.043250e+03	1.416750e+03	5.326100e+02
25%	3.280740e+08	2420.250000	81.73000	56.670000	2.811919e+05	1.649319e+05	9.837612e+04
50%	5.566097e+08	5184.000000	154.06000	97.440000	7.549392e+05	4.647261e+05	2.772260e+05
75%	7.696945e+08	7536.750000	421.89000	263.330000	1.733503e+06	1.141750e+06	5.484568e+05
max	9.955298e+08	9998.000000	668.27000	524.960000	6.617210e+06	5.204978e+06	1.726181e+06

- **Data Cleaning and Feature Engineering**

```
[3] df.shape

(1000, 14)
```

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Region          1000 non-null   object
 1   Country         1000 non-null   object
 2   Item Type       1000 non-null   object
 3   Sales Channel   1000 non-null   object
 4   Order Priority  1000 non-null   object
 5   Order Date      1000 non-null   object
 6   Order ID        1000 non-null   int64
 7   Ship Date       1000 non-null   object
 8   Units Sold      1000 non-null   int64
 9   Unit Price      1000 non-null   float64
 10  Unit Cost       1000 non-null   float64
 11  Total Revenue   1000 non-null   float64
 12  Total Cost      1000 non-null   float64
 13  Total Profit    1000 non-null   float64
dtypes: float64(5), int64(2), object(7)
memory usage: 109.5+ KB
```

- **DisplaythecleanedandengineeredDataFrame**

```python
# Display the cleaned and engineered DataFrame
print(df.head())
```

```
                        Region Country Order Date   Order ID  Ship Date  \
0  Middle East and North Africa   Libya 2014-10-18  686800706 2014-10-31
1                 North America  Canada 2011-11-07  185941302 2011-12-08
2  Middle East and North Africa   Libya 2016-10-31  246222341 2016-12-09
3                          Asia   Japan 2010-04-10  161442649 2010-05-12
4            Sub-Saharan Africa    Chad 2011-08-16  645713555 2011-08-31

   Units Sold  Unit Price  Unit Cost  Total Revenue  Total Cost  ...  \
0        8446      437.20     263.33     3692591.20  2224085.18  ...
1        3018      154.06      90.93      464953.08   274426.74  ...
2        1517      255.28     159.42      387259.76   241840.14  ...
3        3322      205.70     117.11      683335.40   389039.42  ...
4        9845        9.33       6.92       91853.85    68127.40  ...

   Item Type_Office Supplies  Item Type_Personal Care  Item Type_Snacks  \
0                          0                        0                 0
1                          0                        0                 0
2                          0                        0                 0
3                          0                        0                 0
4                          0                        0                 0

   Item Type_Vegetables  Sales Channel_Offline  Sales Channel_Online  \
0                     0                      1                     0
1                     1                      0                     1
2                     0                      1                     0
3                     0                      1                     0
4                     0                      1                     0

   Order Priority_C  Order Priority_H  Order Priority_L  Order Priority_M
0                 0                 0                 0                 1
1                 0                 0                 0                 1
2                 1                 0                 0                 0
3                 1                 0                 0                 0
4                 0                 1                 0                 0

[5 rows x 33 columns]
```

- **Uniquevalueindata**

```
df.nunique()
```

```
Region                          7
Country                       185
Order Date                    841
Order ID                     1000
Ship Date                     835
Units Sold                    960
Unit Price                     12
Unit Cost                      12
Total Revenue                 999
Total Cost                    999
Total Profit                  999
Order Year                      8
Order Month                    12
Order Day                      31
Shipping Time                  51
Item Type_Baby Food             2
Item Type_Beverages             2
Item Type_Cereal                2
Item Type_Clothes               2
Item Type_Cosmetics             2
Item Type_Fruits                2
Item Type_Household             2
Item Type_Meat                  2
Item Type_Office Supplies       2
Item Type_Personal Care         2
Item Type_Snacks                2
Item Type_Vegetables            2
Sales Channel_Offline           2
Sales Channel_Online            2
Order Priority_C                2
Order Priority_H                2
Order Priority_L                2
Order Priority_M                2
dtype: int64
```

- **Checking for duplicate rows**

```
[16] # Checking for duplicate rows
duplicate_rows = df[df.duplicated()]
if not duplicate_rows.empty:
    print("Duplicate rows found!")
    print(duplicate_rows)
else:
    print("No duplicate rows found.")
```

```
No duplicate rows found.
```

- **Handling missing values**

```
[31] # 1. Handling missing values
     df.fillna(method='ffill', inplace=True)  # Forward fill missing values
     df.dropna(inplace=True)  # Drop remaining rows with missing values
```

- **Converting dates to datetime objects**

```
# 2. Converting dates to datetime objects
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
```

- **Creating new features**

```
[9]  # 3. Creating new features
     df['Order Year'] = df['Order Date'].dt.year
     df['Order Month'] = df['Order Date'].dt.month
     df['Order Day'] = df['Order Date'].dt.day
     df['Shipping Time'] = (df['Ship Date'] - df['Order Date']).dt.days
```

- **Encoding categorical variables**

```
[34] # 4. Encoding categorical variables
     df = pd.get_dummies(df, columns=['Item Type', 'Sales Channel', 'Order Priority'])
```

- **Display the cleaned and engineered DataFrame**

```python
# Display the cleaned and engineered DataFrame
print(df.head())
```

```
                        Region Country Order Date   Order ID  Ship Date  \
0  Middle East and North Africa   Libya 2014-10-18  686800706 2014-10-31
1                 North America  Canada 2011-11-07  185941302 2011-12-08
2  Middle East and North Africa   Libya 2016-10-31  246222341 2016-12-09
3                          Asia   Japan 2010-04-10  161442649 2010-05-12
4            Sub-Saharan Africa    Chad 2011-08-16  645713555 2011-08-31

   Units Sold  Unit Price  Unit Cost  Total Revenue  Total Cost  ...  \
0        8446      437.20     263.33     3692591.20  2224085.18  ...
1        3018      154.06      90.93      464953.08   274426.74  ...
2        1517      255.28     159.42      387259.76   241840.14  ...
3        3322      205.70     117.11      683335.40   389039.42  ...
4        9845        9.33       6.92       91853.85    68127.40  ...

   Item Type_Office Supplies  Item Type_Personal Care  Item Type_Snacks  \
0                          0                        0                 0
1                          0                        0                 0
2                          0                        0                 0
3                          0                        0                 0
4                          0                        0                 0

   Item Type_Vegetables  Sales Channel_Offline  Sales Channel_Online  \
0                     0                      1                     0
1                     1                      0                     1
2                     0                      1                     0
3                     0                      1                     0
4                     0                      1                     0

   Order Priority_C  Order Priority_H  Order Priority_L  Order Priority_M
0                 0                 0                 0                 1
1                 0                 0                 0                 1
2                 1                 0                 0                 0
3                 1                 0                 0                 0
4                 0                 1                 0                 0

[5 rows x 33 columns]
```

- **Summarystatistics**

```python
# Summary statistics
summary_stats = df.describe()
print(summary_stats)
```

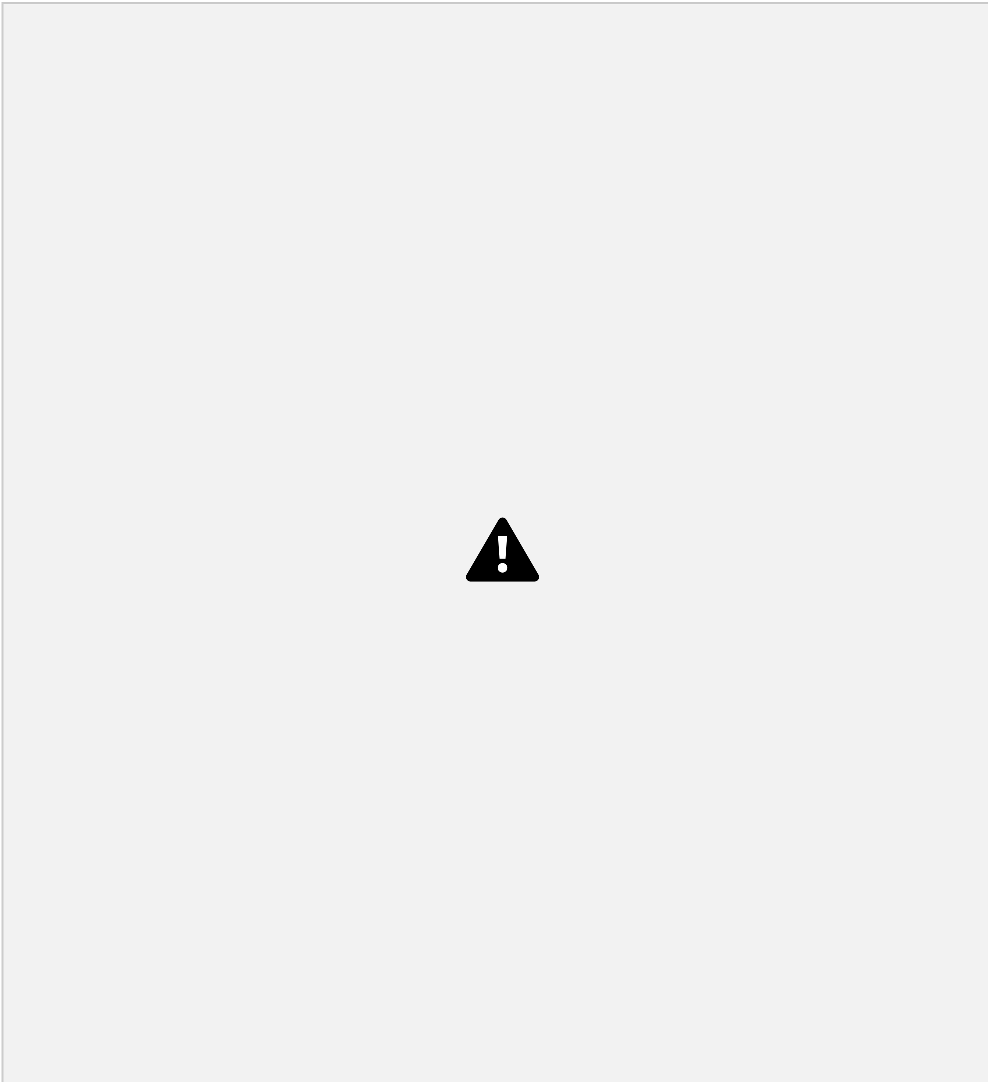
```
           Order ID   Units Sold  Unit Price   Unit Cost  Total Revenue  \
count  1.000000e+03  1000.000000  1000.00000  1000.000000   1.000000e+03
mean   5.496813e+08  5053.988000   262.10684   184.965110   1.327322e+06
std    2.571334e+08  2901.375317   216.02106   175.289311   1.486515e+06
min    1.029280e+08    13.000000     9.33000     6.920000   2.043250e+03
25%    3.280740e+08  2420.250000    81.73000    56.670000   2.811919e+05
50%    5.566097e+08  5184.000000   154.06000    97.440000   7.549392e+05
75%    7.696945e+08  7536.750000   421.89000   263.330000   1.733503e+06
max    9.955298e+08  9998.000000   668.27000   524.960000   6.617210e+06

         Total Cost  Total Profit   Order Year  Order Month    Order Day  ...  \
count  1.000000e+03  1.000000e+03  1000.000000  1000.000000  1000.000000  ...
mean   9.361192e+05  3.912026e+05  2013.234000     6.348000    15.797000  ...
std    1.162571e+06  3.836402e+05     2.164238     3.472889     8.729949  ...
min    1.416750e+03  5.326100e+02  2010.000000     1.000000     1.000000  ...
25%    1.649319e+05  9.837612e+04  2011.000000     3.000000     8.000000  ...
50%    4.647261e+05  2.772260e+05  2013.000000     6.000000    16.000000  ...
75%    1.141750e+06  5.484568e+05  2015.000000     9.000000    23.000000  ...
max    5.204978e+06  1.726181e+06  2017.000000    12.000000    31.000000  ...

       Item Type_Office Supplies  Item Type_Personal Care  Item Type_Snacks  \
count                1000.000000              1000.000000       1000.000000
mean                    0.089000                 0.087000          0.082000
std                     0.284886                 0.281976          0.274502
min                     0.000000                 0.000000          0.000000
25%                     0.000000                 0.000000          0.000000
50%                     0.000000                 0.000000          0.000000
75%                     0.000000                 0.000000          0.000000
max                     1.000000                 1.000000          1.000000

       Item Type_Vegetables  Sales Channel_Offline  Sales Channel_Online  \
count           1000.000000             1000.00000            1000.00000
mean               0.097000                0.52000               0.48000
std                0.296106                0.49985               0.49985
min                0.000000                0.00000               0.00000
25%                0.000000                0.00000               0.00000
50%                0.000000                1.00000               0.00000
75%                0.000000                1.00000               1.00000
max                1.000000                1.00000               1.00000

       Order Priority_C  Order Priority_H  Order Priority_L  Order Priority_M
count       1000.000000       1000.000000       1000.000000       1000.000000
mean           0.262000          0.228000          0.268000          0.242000
std            0.439943          0.419753          0.443139          0.428509
min            0.000000          0.000000          0.000000          0.000000
25%            0.000000          0.000000          0.000000          0.000000
50%            0.000000          0.000000          0.000000          0.000000
75%            1.000000          0.000000          1.000000          0.000000
max            1.000000          1.000000          1.000000          1.000000

[8 rows x 29 columns]
```

- **Uniquevaluesincategoricalcolumns**

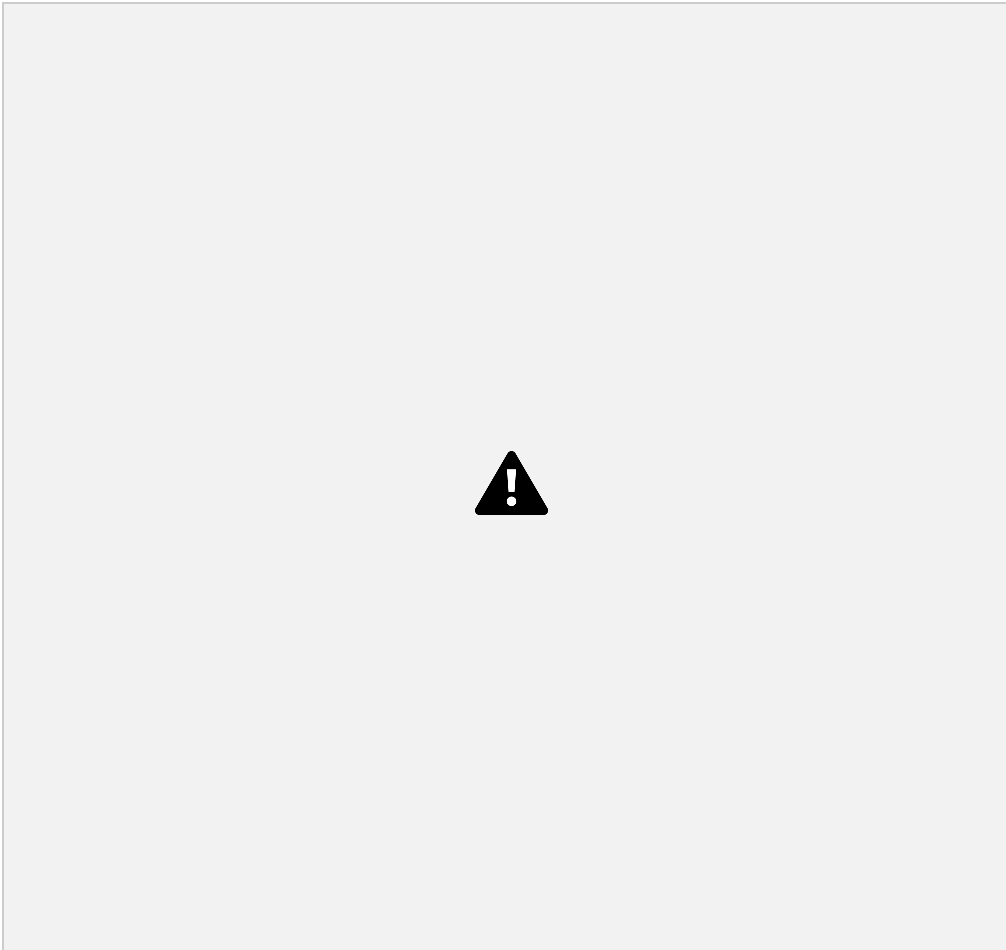
```python
# Unique values in categorical columns
unique_values = {}
for col in df.select_dtypes(include=['object']):
    unique_values[col] = df[col].unique()
print(unique_values)
```

```
{'Region': array(['Middle East and North Africa', 'North America', 'Asia',
       'Sub-Saharan Africa', 'Europe',
       'Central America and the Caribbean', 'Australia and Oceania'],
      dtype=object), 'Country': array(['Libya', 'Canada', 'Japan', 'Chad', 'Armenia', 'Eritrea',
       'Montenegro', 'Jamaica', 'Fiji', 'Togo', 'Greece', 'Sudan',
       'Maldives', 'Estonia', 'Greenland', 'Cape Verde', 'Senegal',
       'Federated States of Micronesia', 'Bulgaria', 'Algeria',
       'Mongolia', 'Grenada', 'Mauritius ', 'Morocco', 'Honduras',
       'Benin', 'Equatorial Guinea', 'Swaziland', 'Trinidad and Tobago',
       'Sweden', 'Belarus', 'Guinea-Bissau', 'Turkey',
       'Central African Republic', 'Laos', 'Israel', 'Bhutan', 'Vanuatu',
       'Burundi', 'Ukraine', 'Croatia', 'Madagascar', 'Malaysia',
       'Uzbekistan', 'Italy', 'Nepal', 'Portugal', 'Panama', 'Botswana',
       'Tanzania', 'Romania', 'Mali', 'Niger', 'Austria', 'India',
       'Luxembourg', 'Iceland', 'Qatar', 'South Sudan', 'United Kingdom',
       'Tunisia ', 'United States of America', 'Liberia', 'South Korea',
       'Kenya', 'Rwanda', 'Cuba', 'Czech Republic', 'Philippines',
       'El Salvador', 'Tonga', 'Democratic Republic of the Congo',
       'Afghanistan', 'Tuvalu', 'Gabon', 'East Timor', 'Jordan', 'Cyprus',
       'Malawi', 'United Arab Emirates', 'China', 'Somalia', 'Bangladesh',
       'Egypt', 'Vietnam', 'Marshall Islands', 'Taiwan', 'Ireland',
       'South Africa', 'Albania', 'Ghana', 'Saint Lucia', 'Macedonia',
       'Germany', 'Poland', 'Namibia', 'Zimbabwe', 'Norway', 'Oman',
       'Serbia', 'Brunei', 'Nicaragua', 'Lithuania',
       'Republic of the Congo', 'Cameroon', 'Moldova ', 'Bahrain',
       'Hungary', 'Iraq', 'Lesotho', 'Lebanon', 'Georgia', 'Ethiopia',
       'Mexico', 'Nigeria', 'Solomon Islands', 'Burkina Faso', 'Kiribati',
       'Comoros', 'Iran', 'Belize', 'Andorra', 'Slovakia',
       'Antigua and Barbuda ', 'Myanmar', 'Nauru', 'Finland',
       'Papua New Guinea', 'Mozambique', 'Spain', 'Belgium',
       "Cote d'Ivoire", 'Switzerland', 'Palau', 'Slovenia', 'Guinea',
       'Russia', 'Seychelles ', 'Costa Rica', 'Liechtenstein', 'Uganda',
       'Guatemala', 'Thailand', 'Denmark', 'Angola', 'North Korea',
       'Yemen', 'Dominican Republic', 'Vatican City', 'Djibouti', 'Malta',
       'The Bahamas', 'Tajikistan', 'Saudi Arabia', 'Mauritania',
       'New Zealand', 'Samoa ', 'Singapore', 'Pakistan',
       'Sao Tome and Principe', 'Turkmenistan', 'Monaco',
       'Saint Kitts and Nevis ', 'Cambodia', 'Kyrgyzstan', 'Indonesia',
       'Kazakhstan', 'Australia', 'Syria', 'Azerbaijan', 'Barbados',
       'Kuwait', 'San Marino', 'Netherlands', 'Kosovo', 'Latvia',
       'Bosnia and Herzegovina', 'Sri Lanka', 'Dominica', 'Haiti',
       'Saint Vincent and the Grenadines', 'Sierra Leone', 'Zambia',
       'France', 'The Gambia'], dtype=object)}
```

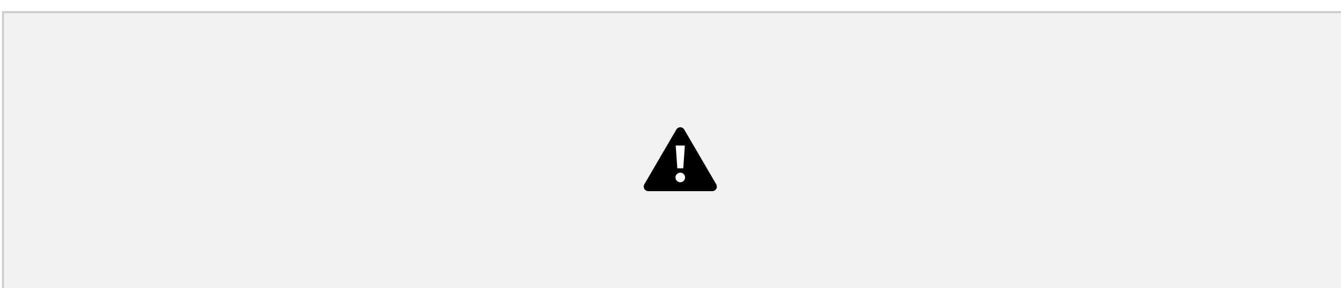
- **Correlation matrix**

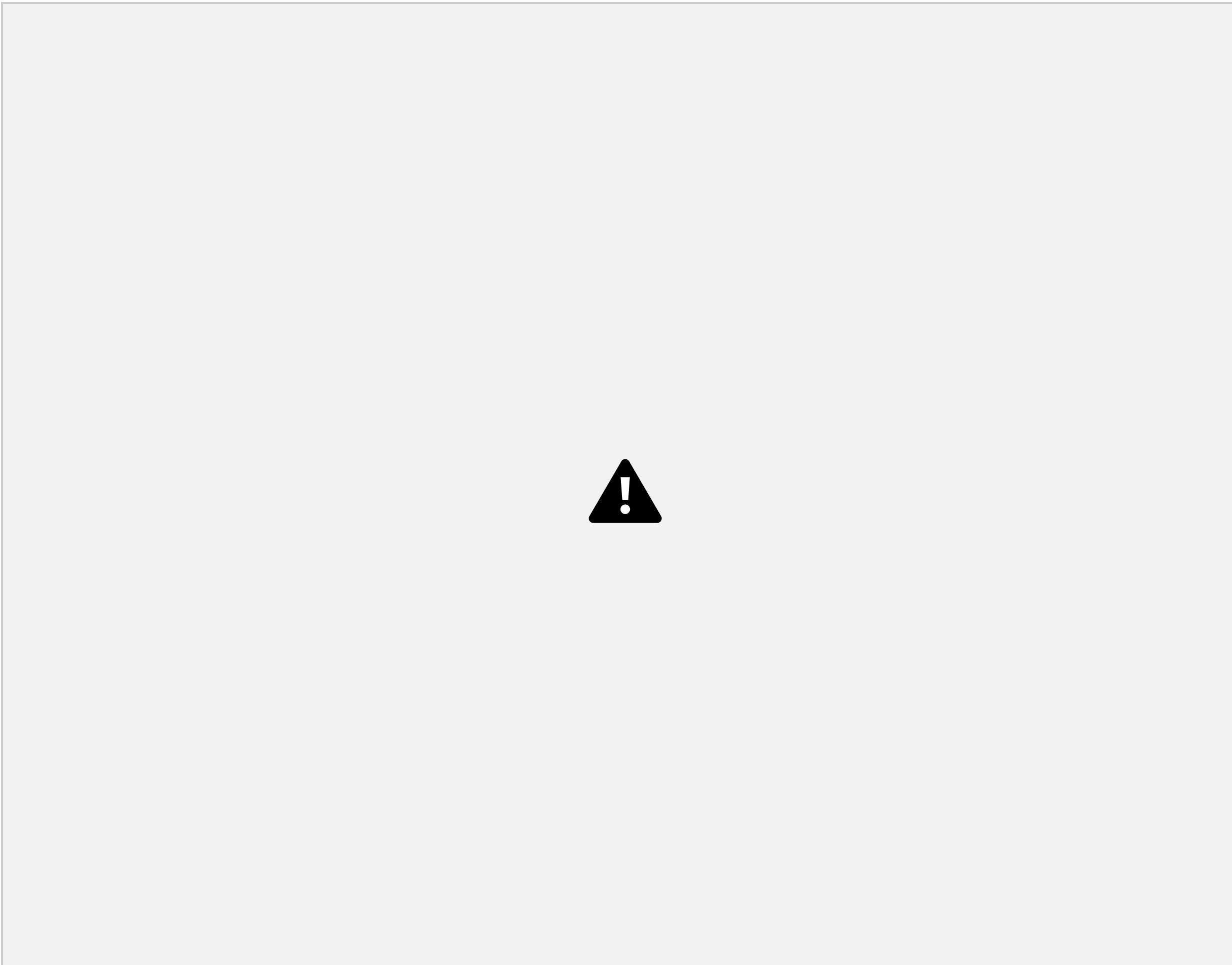


●Frequencydistributionofcategoricalvariables

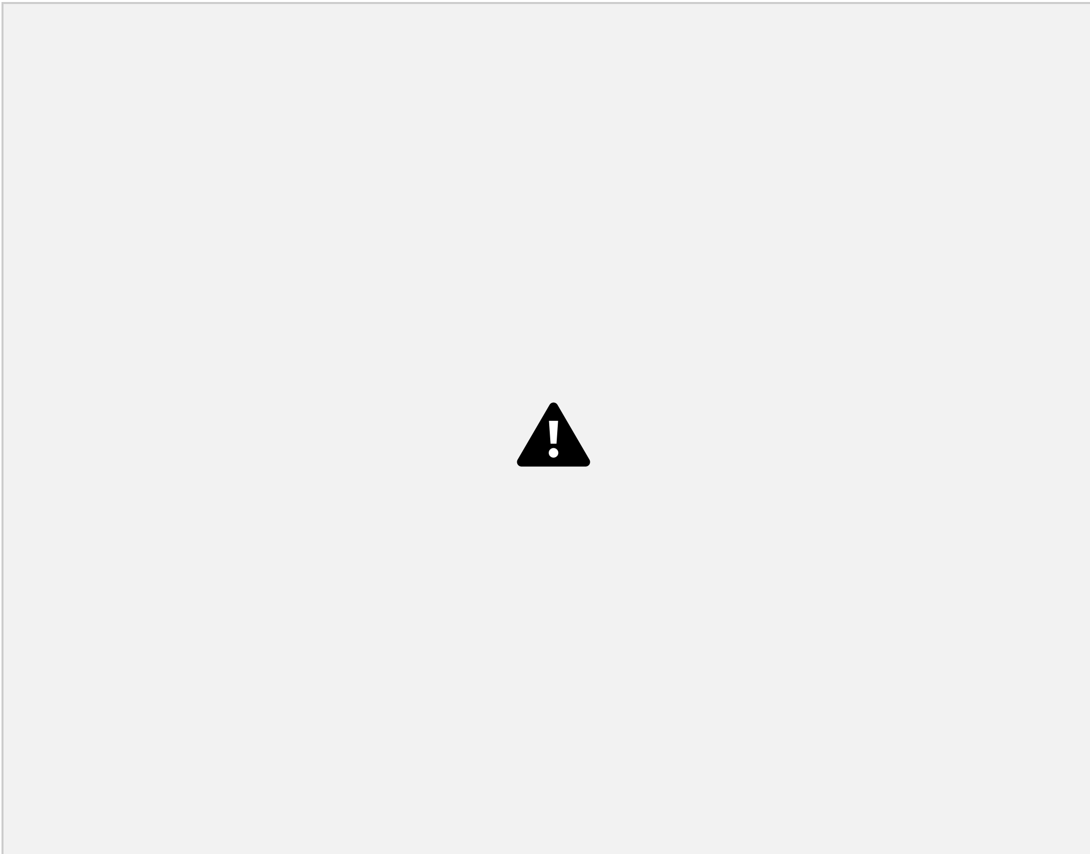


- **Histogram**

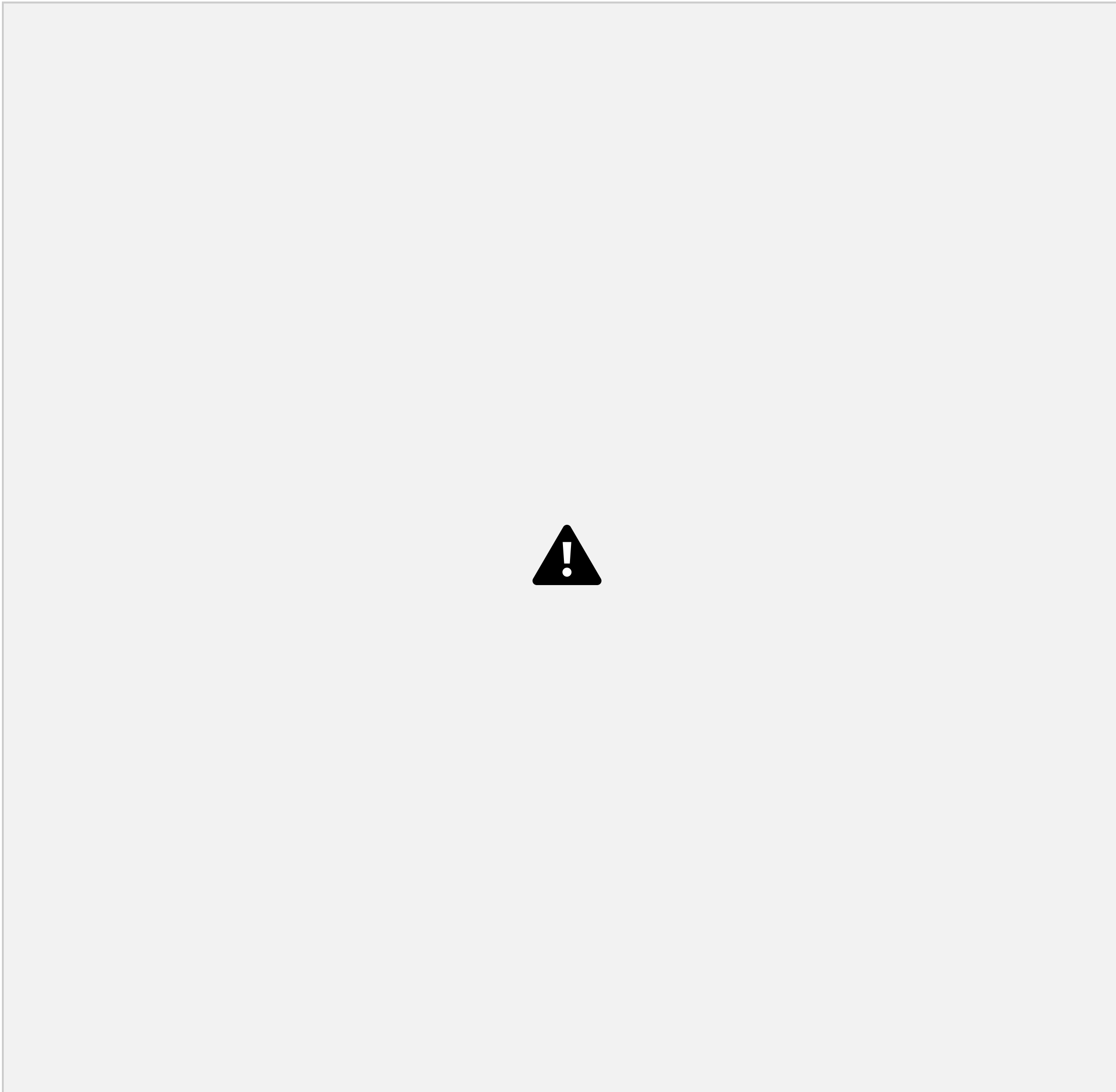




code block generates histograms for the given numerical columns in the DataFrame df. A histogram is a graphical representation of the distribution of data, typically used to identify patterns, distribution, and outliers in the data.

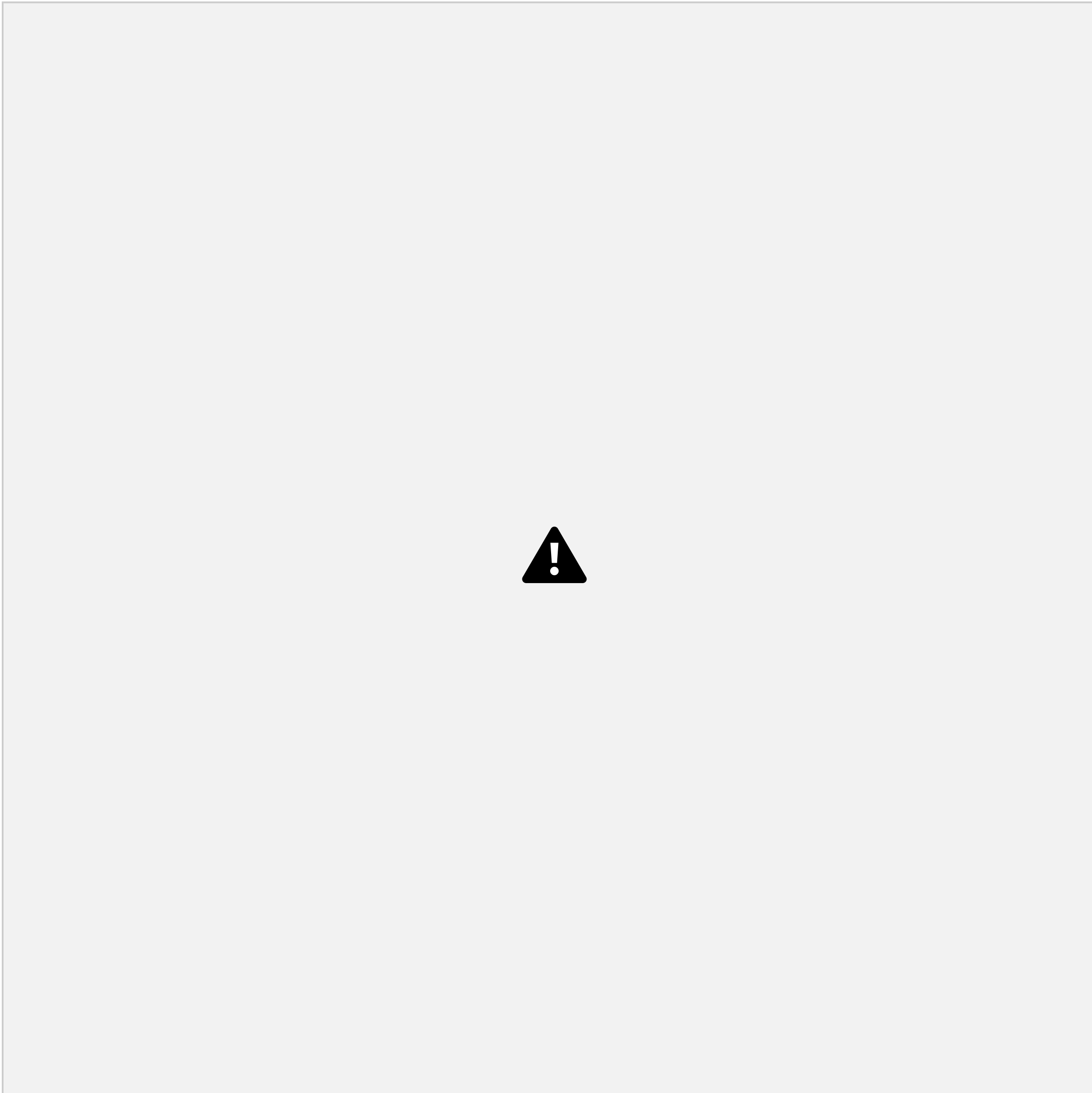


- **Scatter plot**



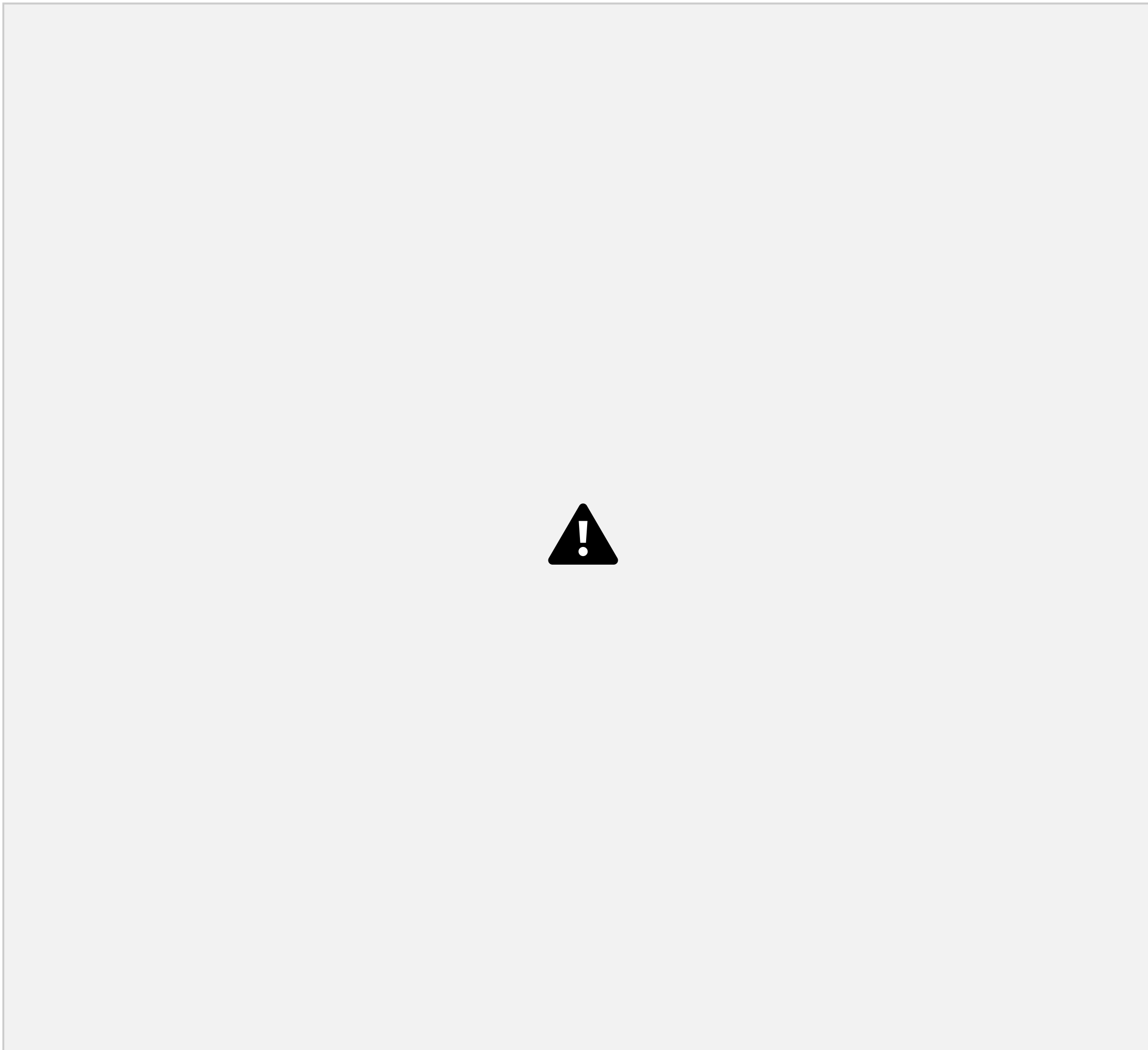
This code block generates a scatter plot to visualize the relationship between 'Total Revenue' and 'Total Profit' in the given DataFrame ‘df’.

- **Heatmap**

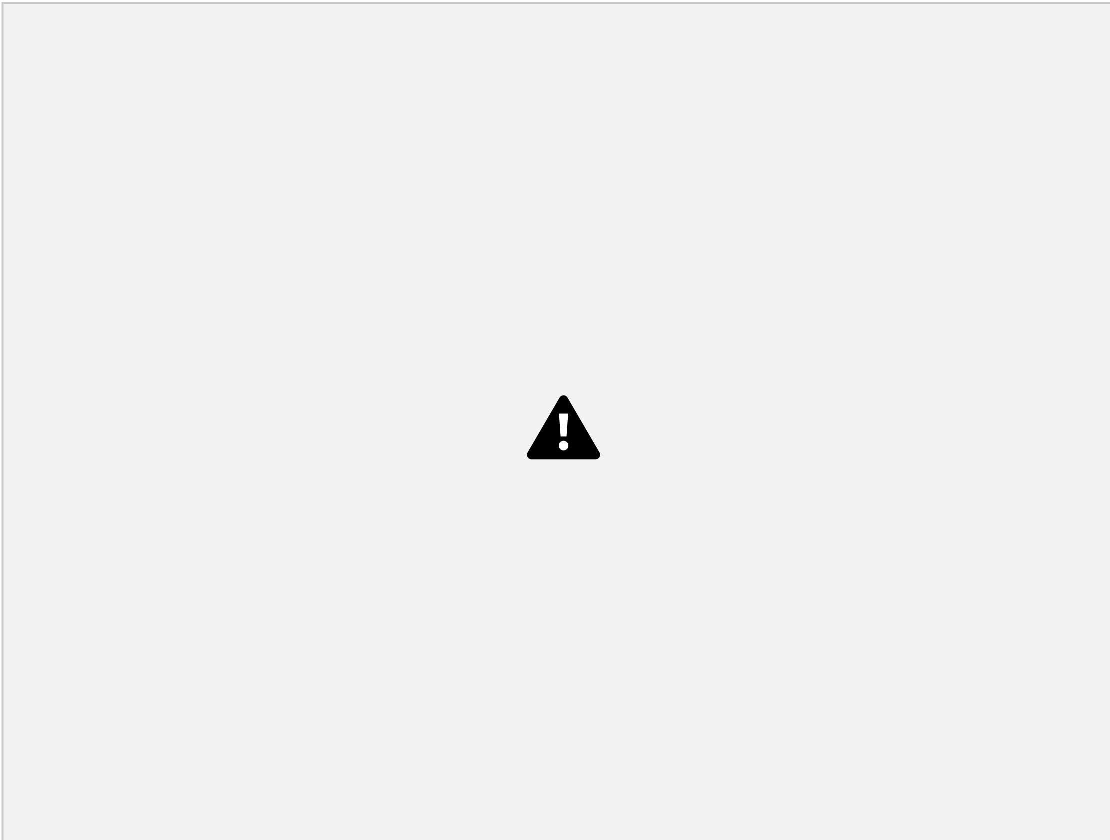


code block generates a heatmap to visualize the correlation matrix between various columns in the given DataFrame df using the seaborn library.

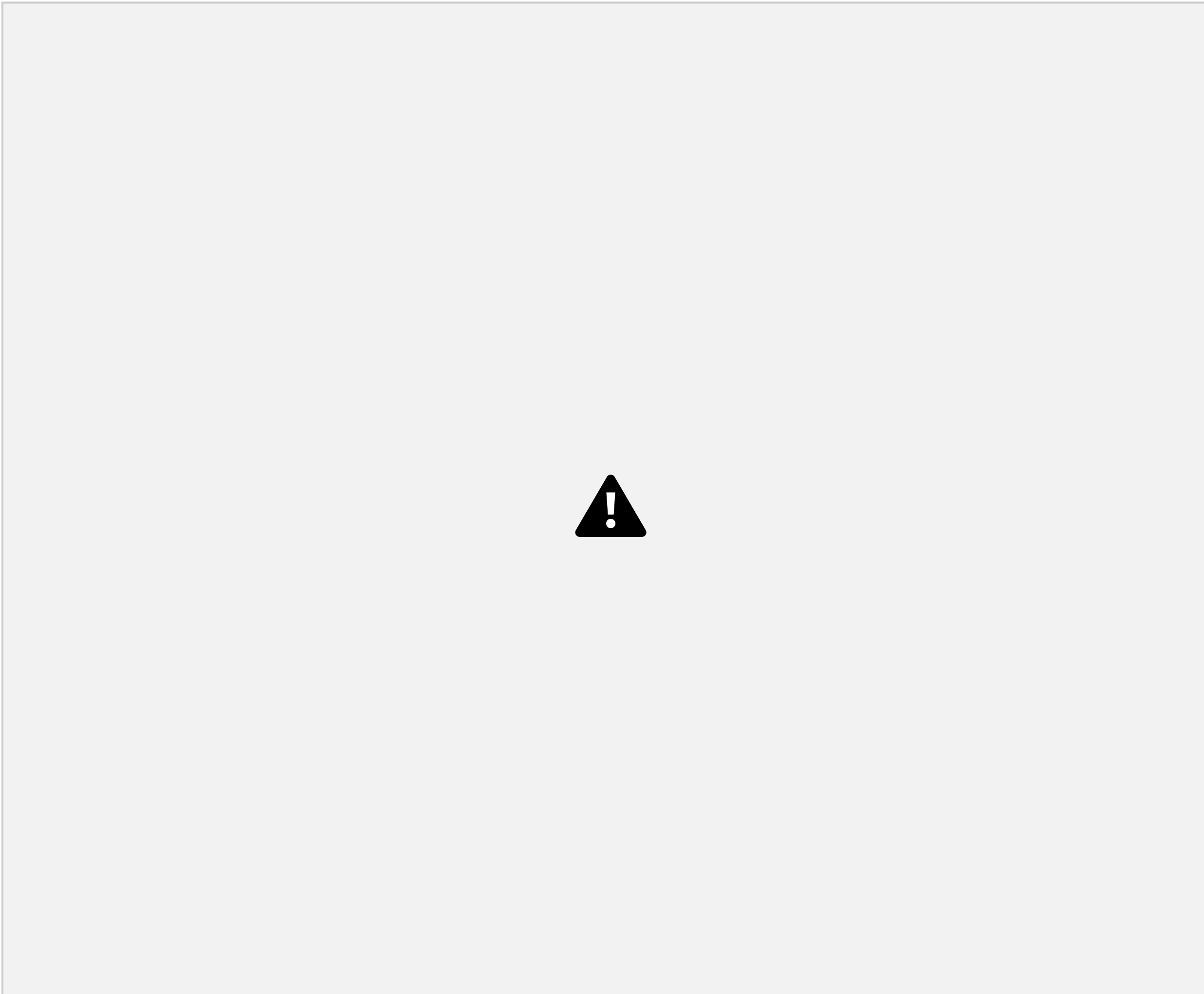
- ## Boxplot



code block generates a series of box plots for each numerical column in the given DataFrame ‘df’. The box plots help visualize the distribution of the values, including the range, quartiles, and potential outliers.



- **Pie chart for the distribution of Region**



code creates a pie chart that visualizes the distribution of unique values in the 'Region' column of the DataFrame ‘df’. The chart's size, labels, percentages, colors, and title are all set according to the specified options