Chocolate Sales

May 6, 2025

[2]:
```
import matplotlib.pyplot as plt
import pandas as pd

# Task 1
#  load the Chocolate Sales.csv
df = pd.read_csv('Chocolate Sales.csv')
```

[3]:
```
#  display first few rows
df.head()
```

[3]:

	Sales Person	Country	Product	Date	Amount	Boxes Shipped
0	Jehu Rudeforth	UK	Mint Chip Choco	04-Jan-22	$5,320	180
1	Van Tuxwell	India	85% Dark Bars	01-Aug-22	$7,896	94
2	Gigi Bohling	India	Peanut Butter Cubes	07-Jul-22	$4,501	91
3	Jan Morforth	Australia	Peanut Butter Cubes	27-Apr-22	$12,726	342
4	Jehu Rudeforth	UK	Peanut Butter Cubes	24-Feb-22	$13,685	184

[4]:
```
# Data types
df.dtypes
```

[4]:
```
Sales Person     object
Country          object
Product          object
Date             object
Amount           object
Boxes Shipped     int64
dtype: object
```

[5]:
```
# Check for missing values
df.isnull().sum()
```

[5]:
```
Sales Person     0
Country          0
Product          0
Date             0
Amount           0
Boxes Shipped    0
dtype: int64
```

[6]:
```
#  names of all columns in the dataset
df.columns
```

[6]:
```
Index(['Sales Person', 'Country', 'Product', 'Date', 'Amount',
       'Boxes Shipped'],
      dtype='object')
```

[7]:
```
# dataset shape(rows,Columns)
df.shape
```

[7]:
```
(1094, 6)
```

[8]:
```
# 2. summary

df.describe()
```

[8]:
```
       Boxes Shipped
count    1094.000000
mean      161.797989
std       121.544145
min         1.000000
25%        70.000000
50%       135.000000
75%       228.750000
max       709.000000
```

[9]:
```
df['Country'].value_counts
```

[9]:
```
<bound method IndexOpsMixin.value_counts of 0             UK
1          India
2          India
3      Australia
4             UK
          ...
1089   Australia
1090         USA
1091      Canada
1092       India
1093       India
```

```
Name: Country, Length: 1094, dtype: object>
```

[10]:
```
# clean dataset
df = df.dropna()    #drop missing rows
```

[11]:
```
#  TASK 2

df.describe()
```

[11]:
```
       Boxes Shipped
count    1094.000000
mean      161.797989
std       121.544145
min         1.000000
25%        70.000000
50%       135.000000
75%       228.750000
max       709.000000
```

[12]:
```
# 2. groupimg we start with the mean
df.groupby("Country")["Boxes Shipped"].mean()
```

[12]:
```
Country
Australia      159.253659
Canada         178.405714
India          160.163043
New Zealand    153.641618
UK             170.028090
USA            149.854749
Name: Boxes Shipped, dtype: float64
```

[13]:
```
df.groupby("Sales Person")["Boxes Shipped"].mean()
```

[13]:
```
Sales Person
Andria Kimpton         165.333333
Barr Faughny           148.046512
Beverie Moffet         184.280000
Brien Boise            152.867925
Camilla Castle         167.937500
Ches Bonnell           156.708333
Curtice Advani         153.782609
Dennison Crosswaite    178.918367
Dotty Strutley         190.361111
Gigi Bohling           134.106383
Gunar Cockshoot        155.279070
Husein Augar           153.921053
Jan Morforth           196.435897
```

```
Jehu Rudeforth          168.511628
Kaine Padly             161.177778
Karlen McCaffrey        205.489362
Kelci Walkden           161.148148
Madelene Upcott         161.755556
Mallorie Waber          145.853659
Marney O'Breen          178.733333
Oby Sorrel              175.673469
Rafaelita Blaksland     126.382353
Roddy Speechley         160.441860
Van Tuxwell             133.313725
Wilone O'Kielt          118.617647
Name: Boxes Shipped, dtype: float64
```

[14]:
```
df.groupby("Product")["Boxes Shipped"].mean()
```

[14]:
```
Product
50% Dark Bites           163.200000
70% Dark Bites           190.833333
85% Dark Bars            155.860000
99% Dark & Pure          165.857143
After Nines              165.140000
Almond Choco             140.333333
Baker's Choco Chips      170.682927
Caramel Stuffed Bars     202.720930
Choco Coated Almonds     165.743590
Drinking Coco            154.642857
Eclairs                  145.950000
Fruit & Nut Bars         154.760000
Manuka Honey Choco       172.911111
Milk Bars                170.000000
Mint Chip Choco          182.377778
Orange Choco             164.510638
Organic Choco Syrup      149.019231
Peanut Butter Cubes      169.469388
Raspberry Choco          148.229167
Smooth Sliky Salty       149.322034
Spicy Special Slims      160.833333
White Choc               142.068966
Name: Boxes Shipped, dtype: float64
```

[15]:
```
#  sum
df.groupby("Country")["Boxes Shipped"].sum()
```

[15]:
```
Country
Australia      32647
Canada         31221
```

```
India          29470
New Zealand    26580
UK             30265
USA            26824
Name: Boxes Shipped, dtype: int64
```

[16]:
```
df.groupby("Product")["Boxes Shipped"].sum()
```

[16]:
```
Product
50% Dark Bites          9792
70% Dark Bites          8015
85% Dark Bars           7793
99% Dark & Pure         8127
After Nines             8257
Almond Choco            6736
Baker's Choco Chips     6998
Caramel Stuffed Bars    8717
Choco Coated Almonds    6464
Drinking Coco           8660
Eclairs                 8757
Fruit & Nut Bars        7738
Manuka Honey Choco      7781
Milk Bars               8330
Mint Chip Choco         8207
Orange Choco            7732
Organic Choco Syrup     7749
Peanut Butter Cubes     8304
Raspberry Choco         7115
Smooth Sliky Salty      8810
Spicy Special Slims     8685
White Choc              8240
Name: Boxes Shipped, dtype: int64
```

[17]:
```
df.groupby("Sales Person")["Boxes Shipped"].sum()
```

[17]:
```
Sales Person
Andria Kimpton         6448
Barr Faughny           6366
Beverie Moffet         9214
Brien Boise            8102
Camilla Castle         5374
Ches Bonnell           7522
Curtice Advani         7074
Dennison Crosswaite    8767
Dotty Strutley         6853
Gigi Bohling           6303
Gunar Cockshoot        6677
```

```
Husein Augar           5849
Jan Morforth           7661
Jehu Rudeforth         7246
Kaine Padly            7253
Karlen McCaffrey       9658
Kelci Walkden          8702
Madelene Upcott        7279
Mallorie Waber         5980
Marney O'Breen         8043
Oby Sorrel             8608
Rafaelita Blaksland    4297
Roddy Speechley        6899
Van Tuxwell            6799
Wilone O'Kielt         4033
Name: Boxes Shipped, dtype: int64
```

[18]:
```
#  lastly the median
df.groupby("Country")["Boxes Shipped"].median()
```

[18]:
```
Country
Australia      119.0
Canada         151.0
India          136.0
New Zealand    129.0
UK             152.0
USA            131.0
Name: Boxes Shipped, dtype: float64
```

[19]:
```
df.groupby("Product")["Boxes Shipped"].median()
```

[19]:
```
Product
50% Dark Bites          126.5
70% Dark Bites          148.5
85% Dark Bars           107.0
99% Dark & Pure         154.0
After Nines             170.0
Almond Choco            125.0
Baker's Choco Chips     135.0
Caramel Stuffed Bars    149.0
Choco Coated Almonds    134.0
Drinking Coco           135.5
Eclairs                 110.5
Fruit & Nut Bars        140.5
Manuka Honey Choco      135.0
Milk Bars               126.0
Mint Chip Choco         180.0
Orange Choco            138.0
```

```
Organic Choco Syrup     145.0
Peanut Butter Cubes     159.0
Raspberry Choco         131.5
Smooth Sliky Salty      113.0
Spicy Special Slims     139.5
White Choc              135.0
Name: Boxes Shipped, dtype: float64
```

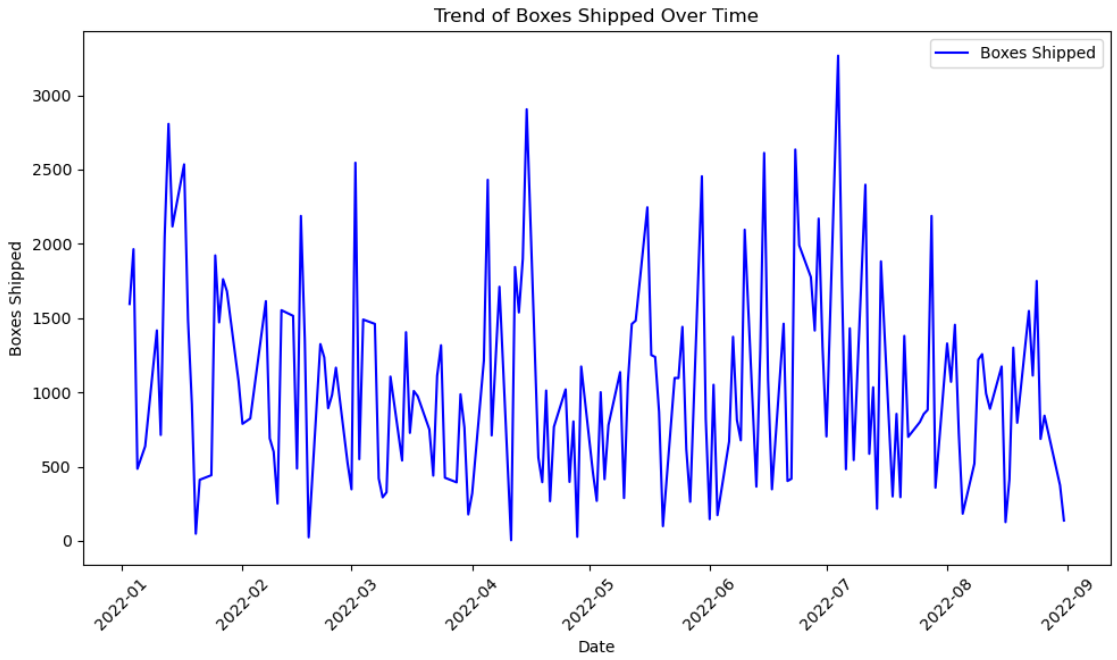
[21]:
```
df.groupby("Sales Person")["Boxes Shipped"].median()
```

[21]:
```
Sales Person
Andria Kimpton          134.0
Barr Faughny            117.0
Beverie Moffet          146.5
Brien Boise             133.0
Camilla Castle          157.0
Ches Bonnell            127.0
Curtice Advani          137.5
Dennison Crosswaite     140.0
Dotty Strutley          207.0
Gigi Bohling            116.0
Gunar Cockshoot         135.0
Husein Augar            131.5
Jan Morforth            178.0
Jehu Rudeforth          138.0
Kaine Padly             123.0
Karlen McCaffrey        177.0
Kelci Walkden           131.5
Madelene Upcott         138.0
Mallorie Waber          107.0
Marney O'Breen          157.0
Oby Sorrel              151.0
Rafaelita Blaksland     101.5
Roddy Speechley         140.0
Van Tuxwell             104.0
Wilone O'Kielt           98.0
Name: Boxes Shipped, dtype: float64
```

[22]:
```
# TASK 3 : VISUALIZATION

print(df.columns)
```

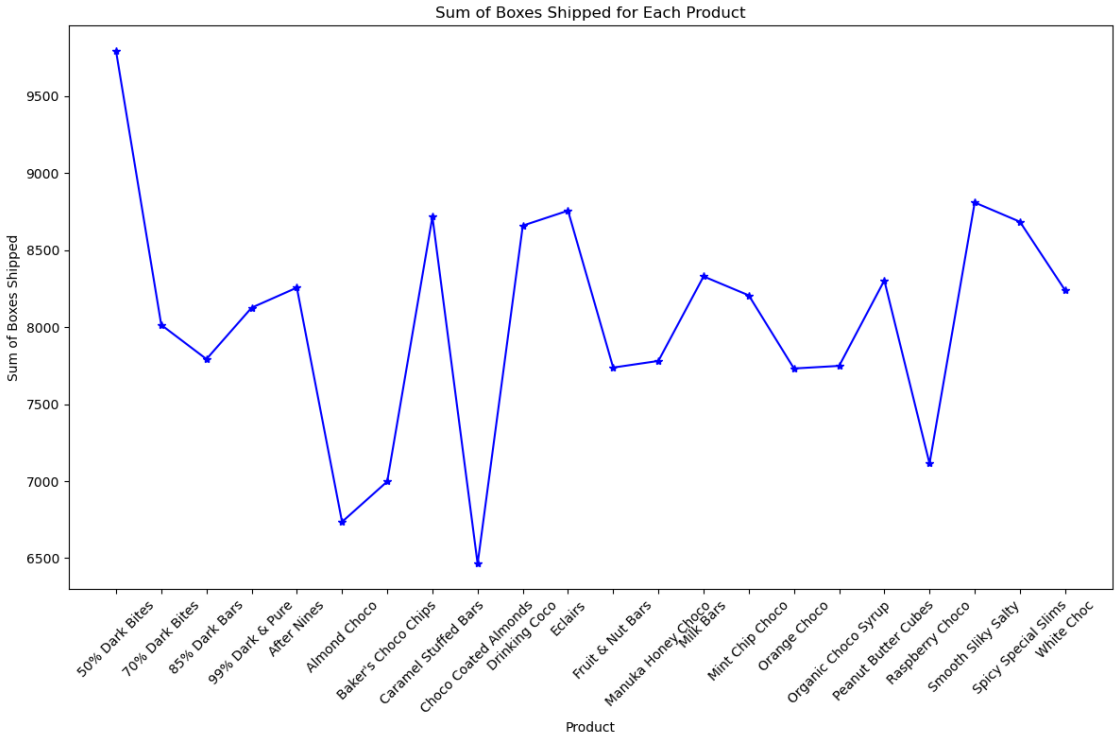
```
Index(['Sales Person', 'Country', 'Product', 'Date', 'Amount',
       'Boxes Shipped'],
      dtype='object')
```

[29]:
```
# 1. START WITH THE LINE GRAPH
# Convert 'Date' to datetime if it's not already
df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y')


# Group by date and sum the Boxes Shipped
df_date = df.groupby('Date')['Boxes Shipped'].sum()

# Plot the line chart
plt.figure(figsize=(10, 6))
plt.plot(df_date.index, df_date.values, label='Boxes Shipped', color='b')
plt.title('Trend of Boxes Shipped Over Time')
plt.xlabel('Date')
plt.ylabel('Boxes Shipped')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
```



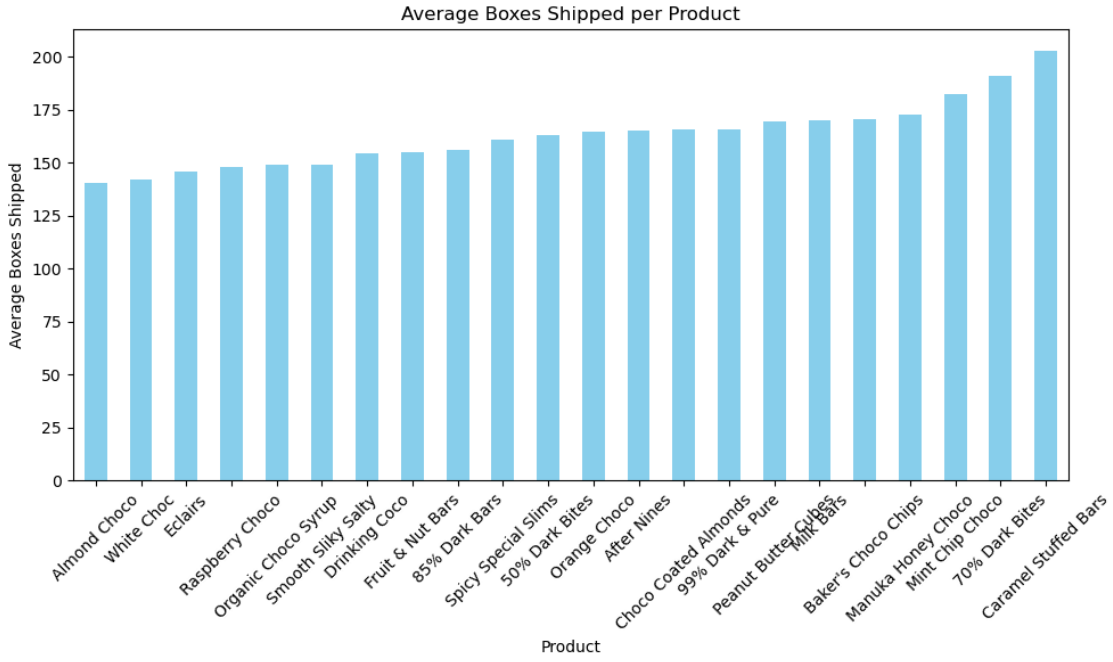
[31]:
```
# I have another line graph-- product vs amount

# Group by Product and calculate the sum of Boxes Shipped
df_product_sum = df.groupby('Product')['Boxes Shipped'].sum()

# Plot the line chart
plt.figure(figsize=(12, 8))  # Set  figure size
```

```
plt.plot(df_product_sum.index, df_product_sum.values, marker='*', color='b')  #␣
 ↪Line plot with markers
plt.title('Sum of Boxes Shipped for Each Product')  # Title
plt.xlabel('Product')  # X-axis label
plt.ylabel('Sum of Boxes Shipped')  # Y-axis label
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()  # Adjust layout to prevent overlap
plt.show()  # Display the plot
```

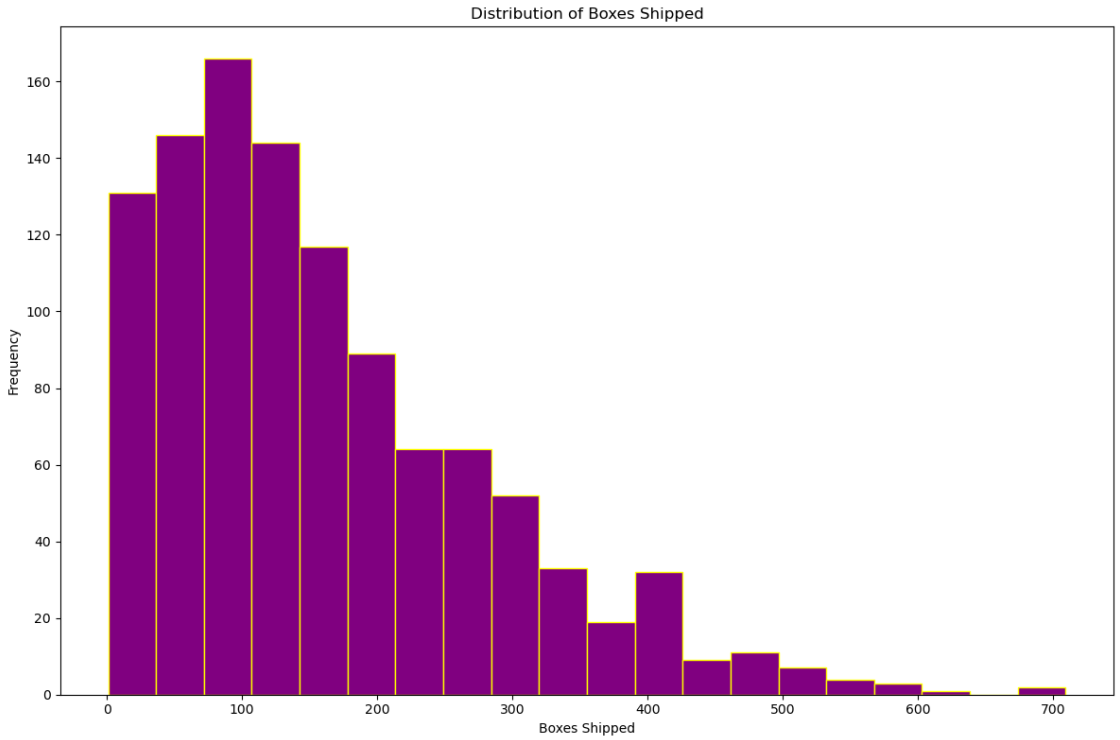


[32]:
```
# 2. BAR GRAPH -Bar Graph[Product vs Boxes Shipped]


# Group by product and compute the mean of Boxes Shipped
df_product = df.groupby('Product')['Boxes Shipped'].mean()

# Plot the bar chart
plt.figure(figsize=(10, 6))
df_product.sort_values().plot(kind='bar', color='skyblue')
plt.title('Average Boxes Shipped per Product')
plt.xlabel('Product')
plt.ylabel('Average Boxes Shipped')
plt.xticks(rotation=45)
plt.tight_layout()
```

```
plt.show()
```

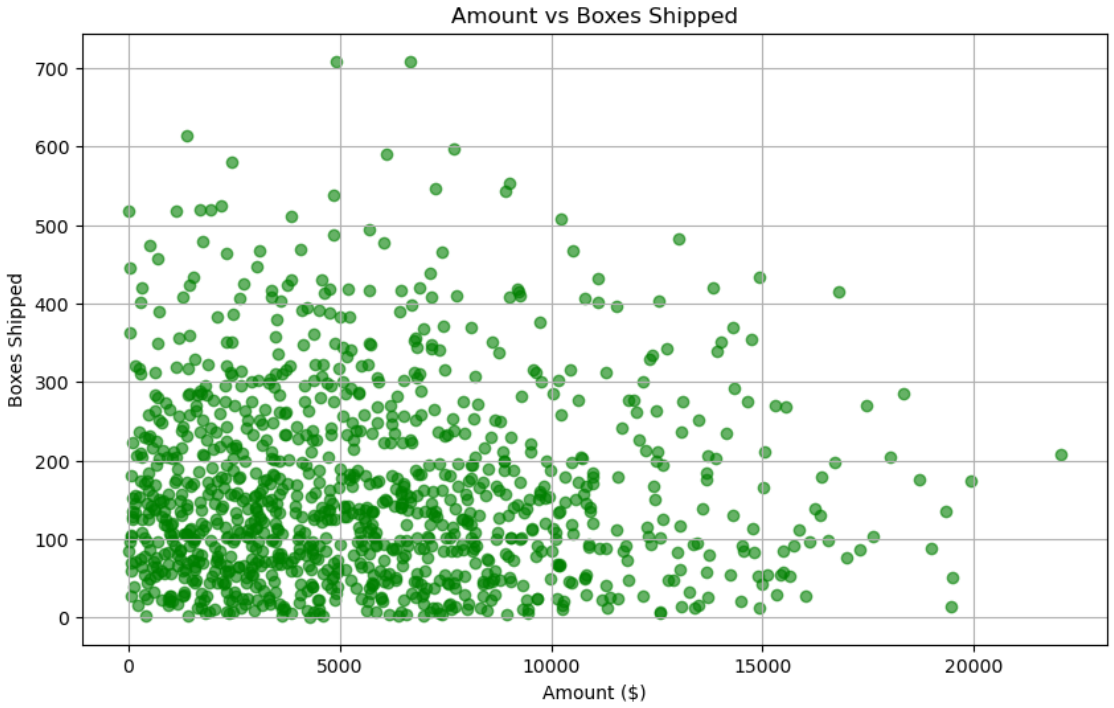


[34]:
```
3. # HISTOGRAM for boxes shipped

# Plot histogram of Boxes Shipped
plt.figure(figsize=(12, 8))
plt.hist(df['Boxes Shipped'], bins=20, color='purple', edgecolor='yellow')
plt.title('Distribution of Boxes Shipped')
plt.xlabel('Boxes Shipped')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



[41]:
```
# Removes $ signs, commas and spaces to convert to numeric
df['Amount'] = df['Amount'].str.replace('$', '', regex=False).str.replace(',',␣
↪'', regex=False).str.strip().astype(float)
```

[42]:
```
import matplotlib.pyplot as plt

# Create scatter plot
plt.figure(figsize=(10, 6))
plt.scatter(df['Amount'], df['Boxes Shipped'], color='green', alpha=0.6)

# Add titles and labels
plt.title('Amount vs Boxes Shipped')
plt.xlabel('Amount ($)')
plt.ylabel('Boxes Shipped')

# Show grid and plot
plt.grid(True)
plt.show()
```



[43]:
```
#  bonus Error Handling

import pandas as pd

try:
    # Attempt to read the CSV file
    df = pd.read_csv("Chocolate Sales.csv")
    print("CSV file loaded successfully.\n")

    # Try converting 'Date' to datetime with specific format
    try:
        df['Date'] = pd.to_datetime(df['Date'], format='%d/%m/%Y')
        print("Date column has been successfull.\n")
    except Exception as e:
        print("Warning: Couldn't convert 'Date' column to datetime with format␣
 ↪'%d/%m/%Y'.")
        print("Error:", e)

    # Check for missing values
    if df.isnull().values.any():
        print("Missing values found. Handling them...")
        df = df.dropna()  # Or use df.fillna(value)
        print("Missing values dropped.\n")
    else:
```

```
        print("No missing values found.\n")

except FileNotFoundError:
    print("Error: The file 'Chocolate Sales.csv' was not found. Please check␣
 ↪the filename or path.")
except pd.errors.EmptyDataError:
    print("Error: The file is empty.")
except pd.errors.ParserError:
    print("Error: There was an error parsing the file.")
except Exception as e:
    print("An unexpected error occurred:", e)
```

```
CSV file loaded successfully.

Warning: Couldn't convert 'Date' column to datetime with format '%d/%m/%Y'.
Error: time data "04-Jan-22" doesn't match format "%d/%m/%Y", at position 0. You
might want to try:
    - passing `format` if your strings have a consistent format;
    - passing `format='ISO8601'` if your strings are all ISO8601 but not
necessarily in exactly the same format;
    - passing `format='mixed'`, and the format will be inferred for each element
individually. You might want to use `dayfirst` alongside this.
No missing values found.
```

[ ]: