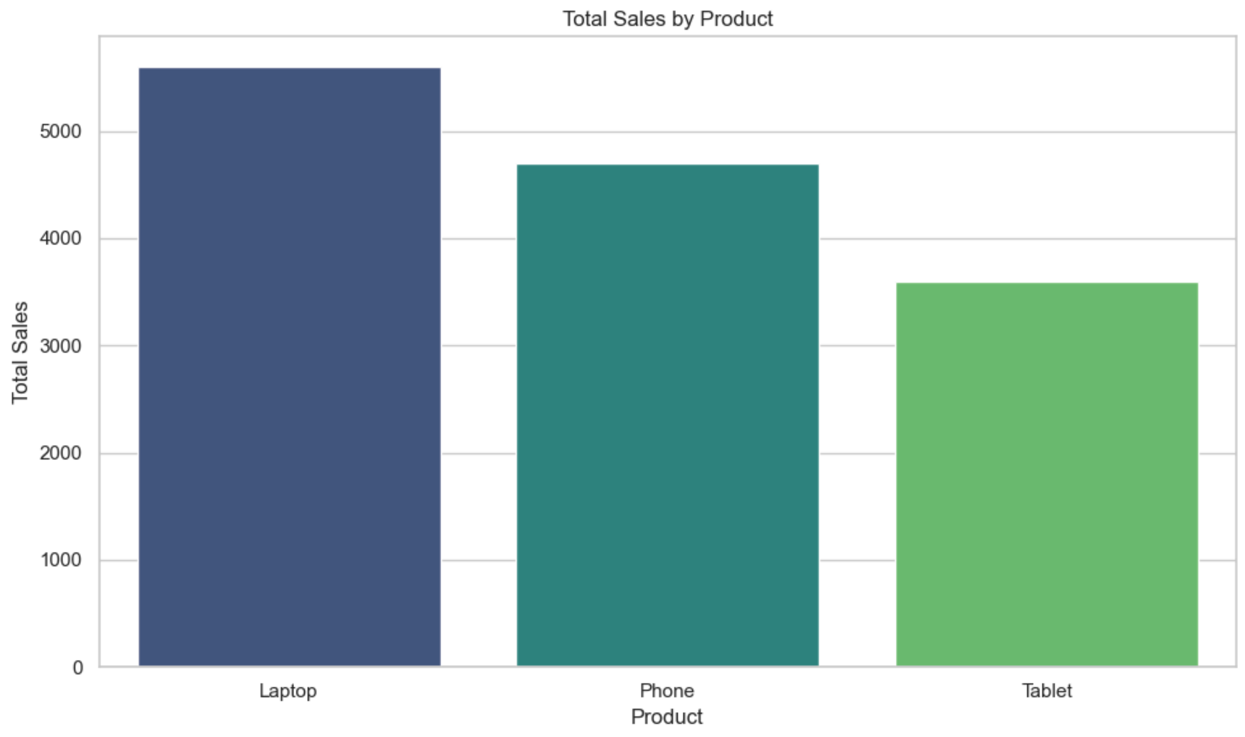
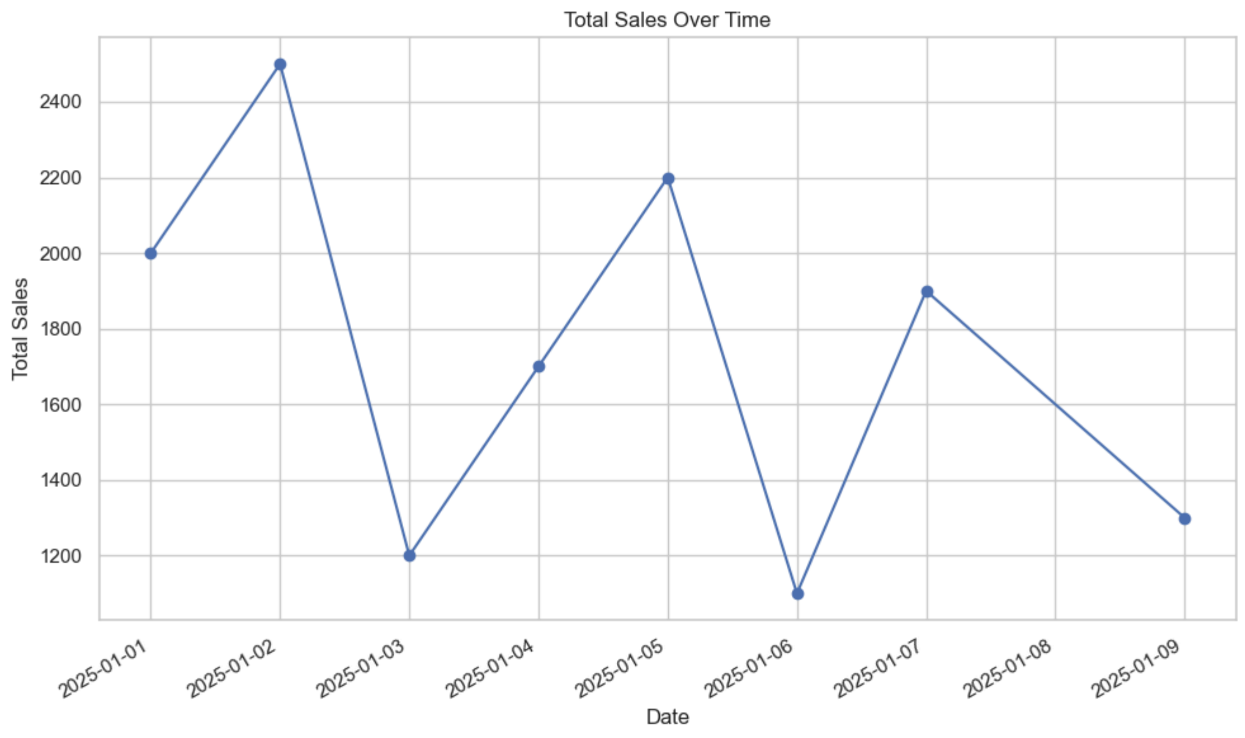
240701160 GURU PRASAD V FDS WEEK 06

```
1) A retail company has provided a CSV file (sales_data.csv) containing sales
transaction details
with the following columns: Date, Product, Quantity, Sales, and Region. As a Data
Analyst,
perform an Exploratory Data Analysis (EDA) using Python (Pandas, NumPy, Matplotlib, and
Seaborn) with the following tasks:
1. Load the dataset into a Pandas DataFrame and display the first few records.
2. Identify and handle missing values (replace missing Sales values with the mean, and
drop
rows with missing Product, Quantity, or Region).
3. Generate summary statistics of the numerical columns.
4. Group the data by Product and compute the total Sales and Quantity sold.
5. Create a bar chart showing the total sales for each product.
6. Convert the Date column to datetime and plot a line chart of total sales over time.
7. Construct a pivot table to analyze sales by Region and Product.
8. Compute the correlation matrix of numerical variables and visualize it using a
heatmap.
Discuss the insights obtained from each step.
```

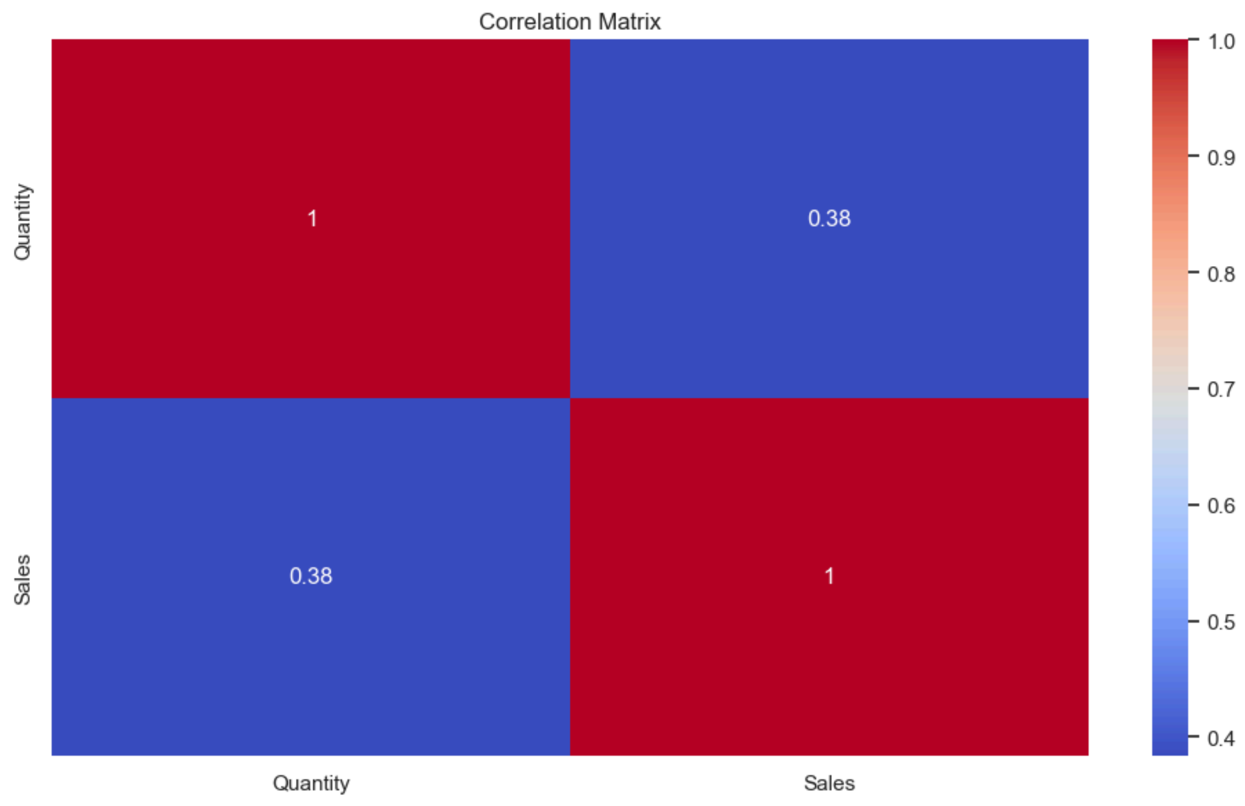
In [6]:
```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Optional: make plots look better
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (10, 6)

# Step 1: Create a sample dataset
data = {
    "Date": [
        "2025-01-01", "2025-01-02", "2025-01-03", "2025-01-04", "2025-01-05",
        "2025-01-06", "2025-01-07", "2025-01-08", "2025-01-09", "2025-01-10"
    ],
    "Product": [
        "Laptop", "Phone", "Tablet", "Laptop", "Phone",
        "Tablet", "Laptop", "Phone", "Tablet", None
    ],
    "Quantity": [2, 5, 3, 1, 4, 2, 2, 3, 4, 1],
    "Sales": [2000, 2500, 1200, None, 2200, 1100, 1900, 2100, 1300, 1000],
    "Region": ["North", "South", "East", "West", "North", "South", "East", None, "West",
}

# Create DataFrame
df = pd.DataFrame(data)

# Save to CSV
df.to_csv("sales_data.csv", index=False)
print("'sales_data.csv' created.")

# Step 2: Load the dataset
df = pd.read_csv("sales_data.csv")
print("\nFirst 5 Records:")
print(df.head())

# Step 3: Handle missing values
print("\nHandling missing values...")
# Replace missing 'Sales' with mean
df['Sales'] = df['Sales'].fillna(df['Sales'].mean())

# Drop rows with missing 'Product', 'Quantity', or 'Region'
df.dropna(subset=['Product', 'Quantity', 'Region'], inplace=True)

# Show remaining missing values
print("Remaining missing values:\n", df.isnull().sum())

# Step 4: Summary statistics
print("\nSummary Statistics:")
print(df.describe())

# Step 5: Group by Product
product_group = df.groupby('Product').agg({'Sales': 'sum', 'Quantity': 'sum'}).sort_value
print("\nTotal Sales and Quantity by Product:")
print(product_group)

# Step 6: Bar chart of total sales by product
plt.figure()
sns.barplot(x=product_group.index, y=product_group['Sales'], palette="viridis")
plt.title("Total Sales by Product")
plt.xlabel("Product")
plt.ylabel("Total Sales")
plt.tight_layout()
plt.show()

# Step 7: Line chart of total sales over time
df['Date'] = pd.to_datetime(df['Date'])
daily_sales = df.groupby('Date')['Sales'].sum()
plt.figure()
daily_sales.plot(marker='o')
plt.title("Total Sales Over Time")
plt.xlabel("Date")
plt.ylabel("Total Sales")
plt.tight_layout()
plt.show()

# Step 8: Pivot table of sales by Region and Product
pivot = pd.pivot_table(df, values='Sales', index='Region', columns='Product', aggfunc='su
print("\nPivot Table (Sales by Region and Product):")
print(pivot)

# Step 9: Correlation heatmap
corr = df[['Quantity', 'Sales']].corr()
plt.figure()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.tight_layout()
plt.show()

print("\nEDA Completed.")
```

```
'sales_data.csv' created.

First 5 Records:
         Date Product  Quantity   Sales Region
0  2025-01-01  Laptop         2  2000.0  North
1  2025-01-02   Phone         5  2500.0  South
2  2025-01-03  Tablet         3  1200.0   East
3  2025-01-04  Laptop         1     NaN   West
4  2025-01-05   Phone         4  2200.0  North

Handling missing values...
Remaining missing values:
 Date        0
Product     0
Quantity    0
Sales       0
Region      0
dtype: int64

Summary Statistics:
       Quantity        Sales
count  8.000000     8.000000
mean   2.875000  1737.500000
std    1.356203   504.090411
min    1.000000  1100.000000
25%    2.000000  1275.000000
50%    2.500000  1800.000000
75%    4.000000  2050.000000
max    5.000000  2500.000000

Total Sales and Quantity by Product:
          Sales  Quantity
Product
Laptop   5600.0         5
Phone    4700.0         9
Tablet   3600.0         9
```





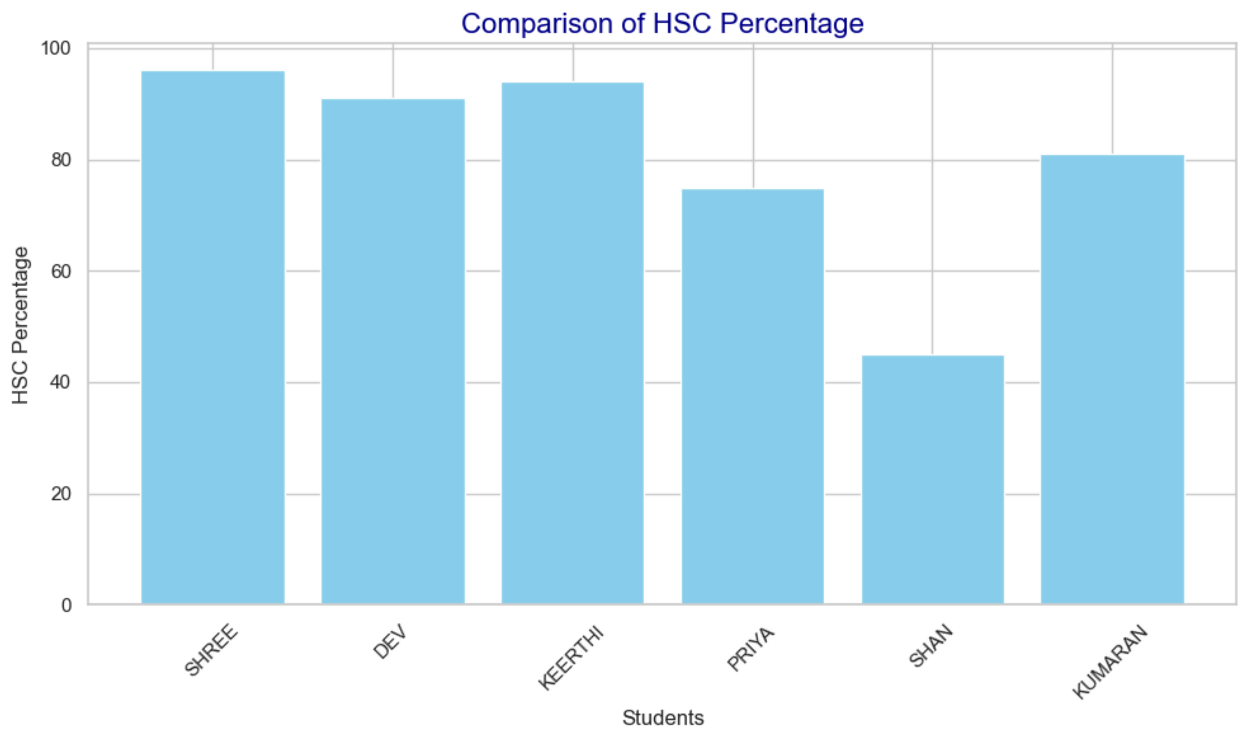
```
Pivot Table (Sales by Region and Product):
Product  Laptop  Phone  Tablet
Region
East       1900      0    1200
North      2000   2200       0
South         0   2500    1100
West       1700      0    1300
```



```
EDA Completed.
```

```
2) A dataset contains the names of six students (SHREE, DEV, KEERTHI, PRIYA, SHAN,
KUMARAN)
along with their Higher Secondary Certificate (HSC) exam percentages: 96, 91, 94, 75,
45, 81.
Using Matplotlib and NumPy in Python:
3) Plot a bar chart comparing the HSC percentages of the students.
4) Set appropriate labels for the x-axis and y-axis.
5) Rotate the x-axis tick labels for better readability.
6) Add a title ("Comparison of HSC Percentage") with customized font size and color.
7) Display the chart using show().
Also, interpret the chart to identify the student with the highest and lowest
percentages.
```

In [7]:
```python
import matplotlib.pyplot as plt
import numpy as np

# Student names and their HSC percentages
students = ['SHREE', 'DEV', 'KEERTHI', 'PRIYA', 'SHAN', 'KUMARAN']
percentages = [96, 91, 94, 75, 45, 81]

# Create an array for the x-axis positions
x_pos = np.arange(len(students))

# Plot bar chart
plt.bar(x_pos, percentages, color='skyblue')

# Set x-axis and y-axis labels
plt.xlabel('Students')
plt.ylabel('HSC Percentage')

# Set x-axis ticks and rotate for better readability
plt.xticks(x_pos, students, rotation=45)

# Add title with customized font size and color
plt.title('Comparison of HSC Percentage', fontsize=16, color='darkblue')

# Show the plot
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()

# Interpretation
max_percent = max(percentages)
min_percent = min(percentages)
max_student = students[percentages.index(max_percent)]
min_student = students[percentages.index(min_percent)]

print(f"The highest percentage is {max_percent}% scored by {max_student}.")
print(f"The lowest percentage is {min_percent}% scored by {min_student}.")
```



```
The highest percentage is 96% scored by SHREE.
The lowest percentage is 45% scored by SHAN.
```

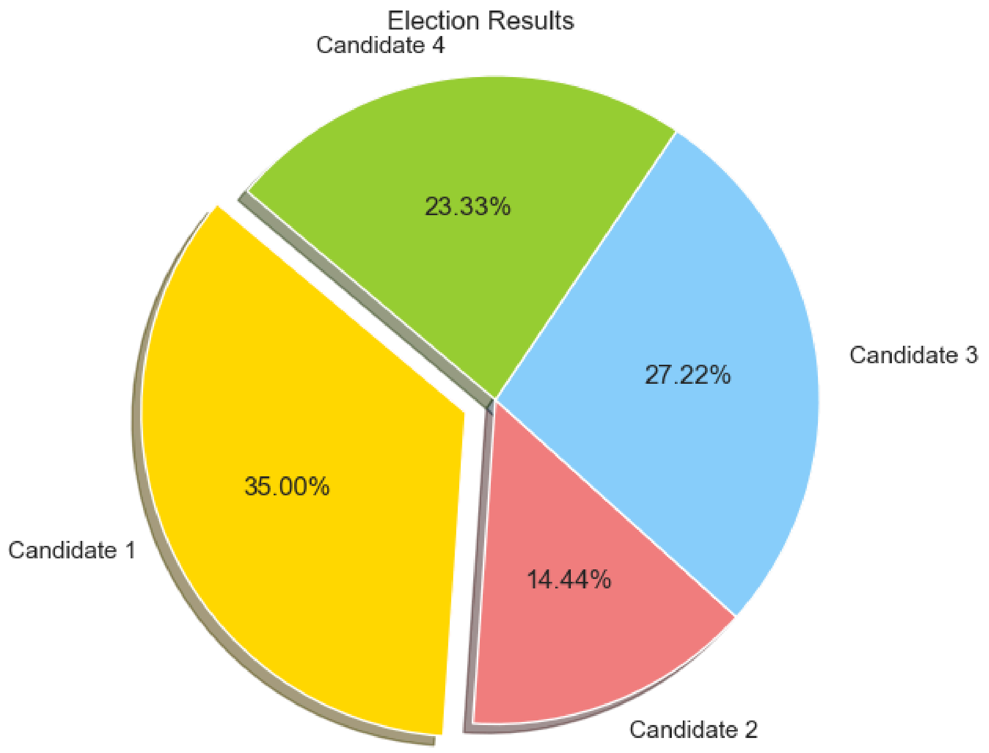
3) The following table shows the number of votes received by four candidates in a college election:

Candidate	Votes
Candidate 1	315
Candidate 2	130
Candidate 3	245
Candidate 4	210

Using Matplotlib in Python, write a program to:

1. Plot a pie chart to represent the election results.
2. Highlight Candidate 1 using the explode parameter.
3. Use different colors for each candidate.
4. Display percentage values on the chart up to two decimal places.
5. Add a suitable title ("Election Results").

In [8]:
```python
import matplotlib.pyplot as plt

# Data
candidates = ['Candidate 1', 'Candidate 2', 'Candidate 3', 'Candidate 4']
votes = [315, 130, 245, 210]

# Explode Candidate 1 (index 0)
explode = (0.1, 0, 0, 0)

# Colors for each candidate
colors = ['gold', 'lightcoral', 'lightskyblue', 'yellowgreen']

# Plot pie chart
plt.pie(votes, labels=candidates, explode=explode, colors=colors,
        autopct='%1.2f%%', shadow=True, startangle=140)

# Title
plt.title('Election Results')

# Equal aspect ratio ensures pie is drawn as a circle
plt.axis('equal')

# Show plot
plt.show()
```



4) To Count the frequency of occurrence of a word in a body of text is often needed during text
processing.
Description: Import the word_tokenize function and gutenberg.

In [12]:
```python
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import gutenberg

# Download the gutenberg corpus and punkt tokenizer if not already installed
nltk.download('gutenberg')
nltk.download('punkt')

# Load the text from the gutenberg corpus
sample = gutenberg.raw("austen-emma.txt")  # You can change to any other available text

# Tokenize the sample text
tokens = word_tokenize(sample)

# Create a list of the first 50 tokens
wlist = tokens[:50]

# Calculate the frequency of each word in the list
wordfreq = [wlist.count(w) for w in wlist]

# Print the word-frequency pairs
print("Pairs\n", list(zip(wlist, wordfreq)))
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Lenovo\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!

Pairs
 [('[', 1), ('Emma', 2), ('by', 1), ('Jane', 1), ('Austen', 1), ('1816', 1), (']', 1),
('VOLUME', 1), ('I', 2), ('CHAPTER', 1), ('I', 2), ('Emma', 2), ('Woodhouse', 1), (',',
5), ('handsome', 1), (',', 5), ('clever', 1), (',', 5), ('and', 3), ('rich', 1), (',',
5), ('with', 2), ('a', 1), ('comfortable', 1), ('home', 1), ('and', 3), ('happy', 1),
('disposition', 1), (',', 5), ('seemed', 1), ('to', 1), ('unite', 1), ('some', 1), ('o
f', 2), ('the', 2), ('best', 1), ('blessings', 1), ('of', 2), ('existence', 1), (';',
1), ('and', 3), ('had', 1), ('lived', 1), ('nearly', 1), ('twenty-one', 1), ('years',
1), ('in', 1), ('the', 2), ('world', 1), ('with', 2)]
```