1. Given a dataset of monthly sales figures for a year, visualize the sales using a line chart and bar chart using the matplotlib library.

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'] sales = [200, 220, 250, 275, 290, 320, 350, 370, 400, 420, 450, 480]

In [153…

```
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', '
sales = [200, 220, 250, 275, 290, 320, 350, 370, 400, 420, 450, 480]

# Creating a DataFrame

import pandas as pd

df = pd.DataFrame({
    'months' : months,
'sales' : sales

})
```

In [155…

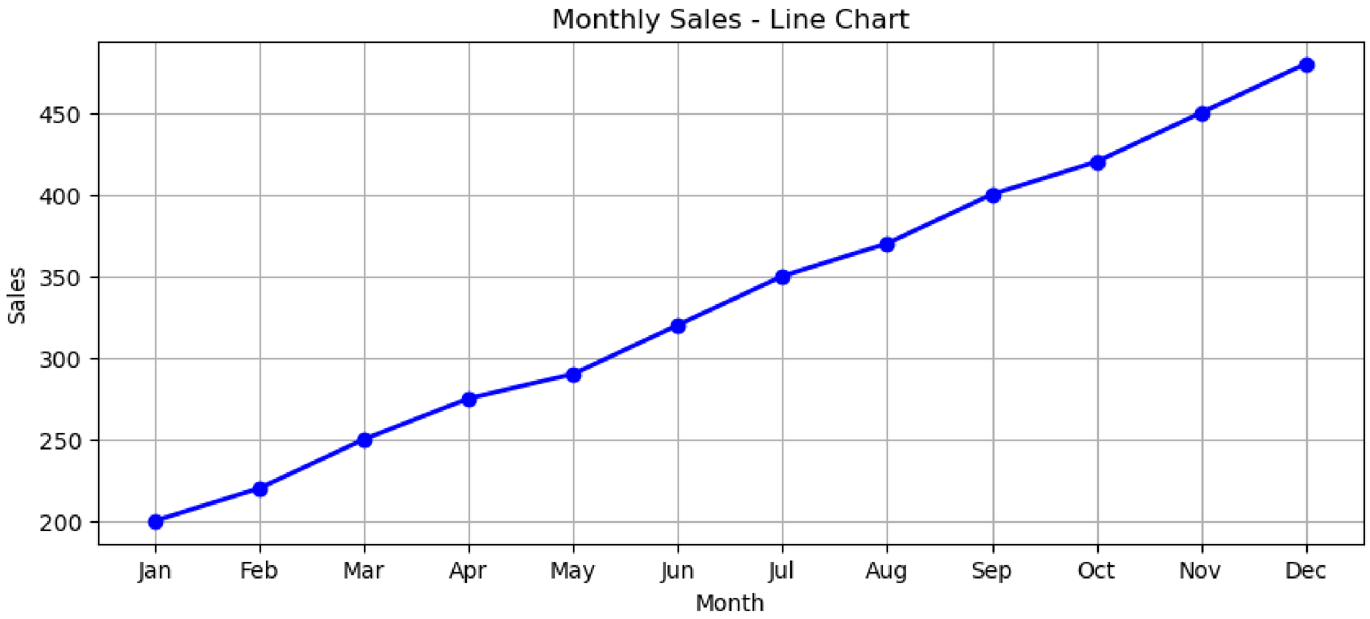
```
df.head()
```

Out[155]:

	months	sales
0	Jan	200
1	Feb	220
2	Mar	250
3	Apr	275
4	May	290

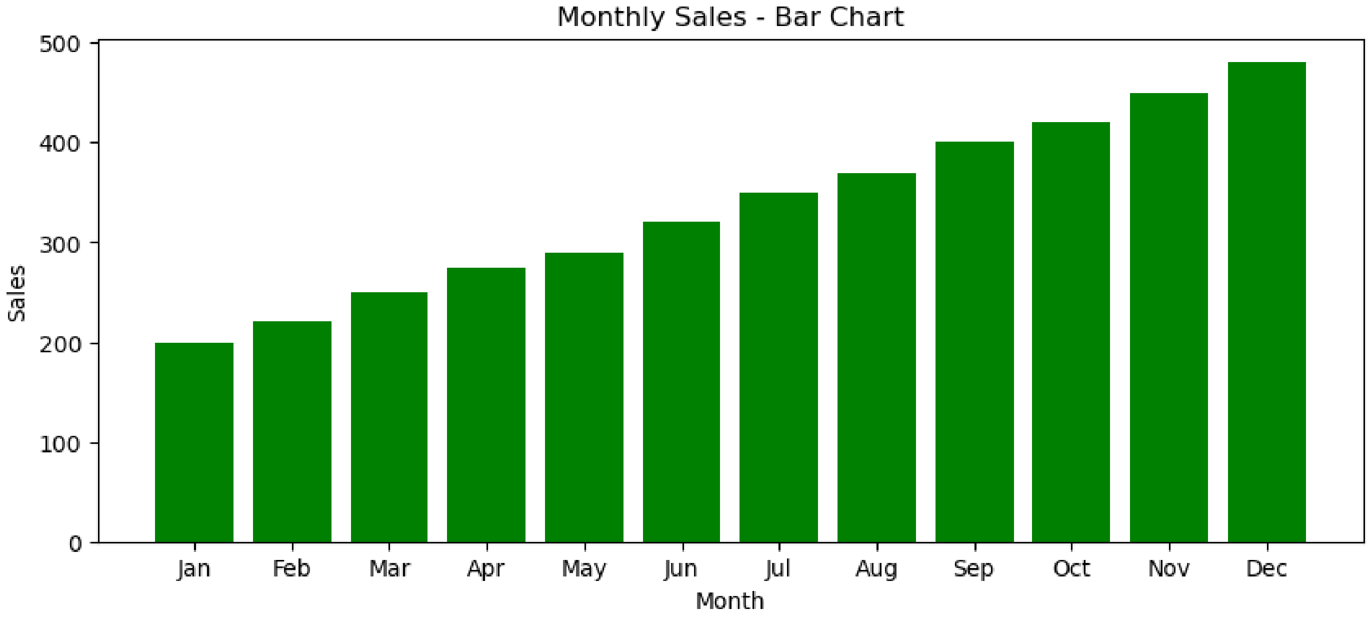
In [157…

```
import matplotlib.pyplot  as plt
# Step 2: Create the line chart
plt.figure(figsize=(10,4))
# Line chart
plt.plot(months, sales, marker='o', color='b', linestyle='-', linewidth=2)
plt.title('Monthly Sales - Line Chart')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.grid(True)
plt.show()
```



In [159…

```
import matplotlib.pyplot  as plt
# Step 2: Create the Bar chart
plt.figure(figsize=(10,4))
# Bar chart
plt.bar(months, sales, color='green')
plt.title('Monthly Sales - Bar Chart')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.show()
```



2. Given a dataframe df with a column age containing some missing values, write a code snippet to replace these missing values with the mean age.

In [162…

```
import pandas as pd

# Assuming df is your DataFrame with an 'age' column

df = pd.DataFrame({
    'name': ['Jo', 'Mark', 'Nav', 'Kiran', 'Sam'],
    'age': [None, 30, 32, None, 24]
})
```

In [164…

```
# Original df
print("Original Dataframe")
print("__________________")
print(df)
```

```
Original Dataframe
__________________
    name   age
0     Jo   NaN
1   Mark  30.0
2    Nav  32.0
3  Kiran   NaN
4    Sam  24.0
```

In [166…

```
# Step 1: Calculate the mean of the 'age' column (excluding missing values)
mean_age = df['age'].mean()

# Step 2: Replace missing values in 'age' column with the calculated mean
df['age'].fillna(mean_age, inplace=True)

# Optional: Display the DataFrame to verify missing values have been replaced
print("Df after updating missing values with mean")
print("_________________________________\n")
print(df)
```

```
Df after updating missing values with mean
_________________________________

    name        age
0     Jo  28.666667
1   Mark  30.000000
2    Nav  32.000000
3  Kiran  28.666667
4    Sam  24.000000
```

3. Given a list of numbers, compute the mean, median, and mode using the numpy and pandas libraries.

numbers = [4, 5, 6, 6, 6, 4, 5, 5, 7]

In [169…

```
import pandas as pd
import numpy as np
from scipy import stats
```

In [173…

```
numbers = [4, 5, 6, 6, 6, 4, 5, 5, 7]
# Convert the list to a Pandas Series
series = pd.Series(numbers)

# Step 1: Calculate the mean using numpy and pandas
mean = np.mean(numbers)
mean_pd= series.mean()

# Step 2: Calculate the median using numpy and pandas
median = np.median(numbers)
median_pd= series.median()

# Step 3: Calculate the mode using pandas and numpy
mode_pd = series.mode()[0]  # mode() returns a series, take the first element
#Convert the list to a numpy array
```

```
array = np.array(numbers)

# Calculate the mode using scipy.stats.mode
mode_result = stats.mode(array, axis=None)
mode = mode_result[0]

# Display the results
print(f"Mean using Numpy: {mean}")
print(f"Mean using Pandas:{mean_pd}")
print(f"Median using Numpy: {median}")
print(f"Median using pandas: {median_pd}")
print(f"Mode using Numpy: {mode}")
print(f"Mode using pandas: {mode_pd}")
```

```
Mean using Numpy: 5.333333333333333
Mean using Pandas:5.333333333333333
Median using Numpy: 5.0
Median using pandas: 5.0
Mode using Numpy: 5
Mode using pandas: 5
```

4.Perform a 1-sample t-test to check if the given sample of student heights is significantly different from a population mean height of 160 cm.

student_heights = [158, 162, 161, 159, 163, 160, 157, 164]

In [176…

```
import numpy as np
from scipy import stats

# Sample Students height data
student_heights = [158, 162, 161, 159, 163, 160, 157, 164]

# Hypothesized population mean
population_mean = 160

# Perform one-sample t-test
t_statistic, p_value = stats.ttest_1samp(student_heights, population_mean)

# Display results
print(f"t-statistic: {t_statistic}")
print(f"p-value: {p_value}")

# Interpretation of results
alpha = 0.05  # significance level
if p_value < alpha:
    print("We reject the null hypothesis: The sample mean is significantly different fro
else:
    print("We fail to reject the null hypothesis: There's no significant difference betw
```

```
t-statistic: 0.5773502691896258
p-value: 0.5817882345917442
We fail to reject the null hypothesis: There's no significant difference between the sam
ple mean and the hypothesized population mean.
```

5. Given the distribution of student grades, visualize the grade distribution using a pie chart in the matplotlib library.

grades = ['A', 'B', 'C', 'D', 'F']

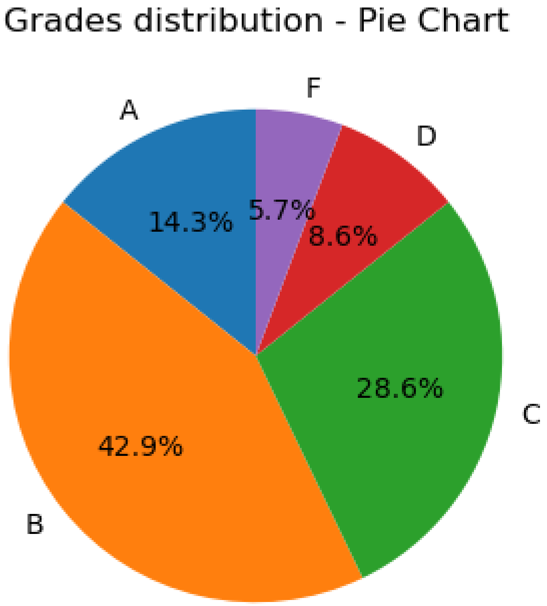
students = [5, 15, 10, 3, 2]

In [179…

```python
# Creating a DataFrame
import pandas as pd

grades = ['A', 'B', 'C', 'D', 'F']
students = [5, 15, 10, 3, 2]


df = pd.DataFrame({
   'grades' : grades,
'students' : students

})
```

In [181…

```python
import matplotlib.pyplot  as plt
# Step 1: Create the Pie chart
plt.figure(figsize=(10,4))
# Pie chart
plt.pie(df['students'], labels = df['grades'] , autopct='%1.1f%%', startangle=90)
plt.title('Grades distribution - Pie Chart')
plt.show()
```



6. Given two dataframes df1 and df2 with a common column key, write a code snippet to perform an inner join using Pandas.

How would you modify this to perform a left join instead

In [184…

```python
# Example dataframes with Enrollments and New grades data
df1 = pd.DataFrame({
    'student_id':[101,102,103,105],
    'course_id':[301,302,301,304],
    'semester':['2024 Spring', '2024 Spring', '2024 Fall','2024 Spring'],
})

df2 = pd.DataFrame({
    'student_id':[101,102,103,105,106],
    'course_id':[301,302, 301,304,300],
    'semester':['2024 Spring', '2024 Spring', '2024 Fall','2024 Spring','2024 Spring'],
} )
```

```
# Merge the datasets on the 'student_id' column with inner join
merged_df = pd.merge(df1, df2, on='student_id', how='inner')


# print

print("Inner join merged df")
print("__________________")
print(merged_df)
```

```
Inner join merged df

__________________
   student_id  course_id_x   semester_x  course_id_y   semester_y
0         101          301  2024 Spring          301  2024 Spring
1         102          302  2024 Spring          302  2024 Spring
2         103          301    2024 Fall          301    2024 Fall
3         105          304  2024 Spring          304  2024 Spring
```

In [186…

```
# Merge the datasets on the 'student_id' column with left join
left_merged_df = pd.merge(df1, df2, on='student_id', how='left')
print("Lef join merged df")
print("__________________")
print(left_merged_df)
```

```
Lef join merged df

__________________
   student_id  course_id_x   semester_x  course_id_y   semester_y
0         101          301  2024 Spring          301  2024 Spring
1         102          302  2024 Spring          302  2024 Spring
2         103          301    2024 Fall          301    2024 Fall
3         105          304  2024 Spring          304  2024 Spring
```

7. Consider a dataset df with a feature column feature_A. Write a Python code snippet to scale feature_A between 0 and 1 using the Min Max scaling technique without the help of external libraries. What would be the formula to reverse this scaling?

A Min-Max scaling is typically done via the following equation:

```
Xsc=X−X_min/X_max−X_min
```

In [190…

```
import pandas as pd

# Sample dataset with feature_A
df = pd.DataFrame({'feature_A': [10, 20, 30, 40, 50, 60, 70, 90]})
```

In [194…

```
# Calculating max of the column feature_A and assigning to X_max
X_max = df['feature_A'].max()

# Calculating min of the column feature_A and assigning to X_min
X_min = df['feature_A'].min()
```

In [196…

```
# Step 2: Apply Min-Max Scaling
df['feature_A_scaled'] = (df['feature_A'] - X_min) / (X_max - X_min)
```

In [198…

```
# Print df

print("Min Max scaling after applying formula")
print("_________________________________\n")
print(df)
```

```
Min Max scaling after applying formula
_________________________________

   feature_A  feature_A_scaled
0         10             0.000
1         20             0.125
2         30             0.250
3         40             0.375
4         50             0.500
5         60             0.625
6         70             0.750
7         90             1.000
```

Reverse min max scaler

```
X= Xscaled*(X_max-X_min) + X_min
```

In [201…

```
# Reverse the scaling to get original values
df['feature_A_original'] = df['feature_A_scaled'] * (X_max - X_min) + X_min
```

In [203…

```
print("Min Max after reversing the formula to original")

# Display the DataFrame with original feature after reverse scaling
print(df)
```

```
Min Max after reversing the formula to original
   feature_A  feature_A_scaled  feature_A_original
0         10             0.000                10.0
1         20             0.125                20.0
2         30             0.250                30.0
3         40             0.375                40.0
4         50             0.500                50.0
5         60             0.625                60.0
6         70             0.750                70.0
7         90             1.000                90.0
```

In [ ]: