Question 1:

A retail store wants to identify customers who make frequent purchases. Given the dataset below, write a code to:

1. Group customers by their IDs.
2. Calculate the total number of purchases per customer.
3. Identify the top 3 frequent customers.

Dataset:

```
data = {'Customer_ID': [101, 102, 103, 101, 104, 102, 101, 105, 102, 103],
        'Purchase_Amount': [200, 150, 180, 220, 300, 200, 100, 400, 250,
300]}
```

Expected Output:

```
Total Purchases per Customer:
   Customer_ID  Purchase_Amount
0          101              520
1          102              600
2          103              480
3          104              300
4          105              400

Top 3 Frequent Customers:
   Customer_ID  Purchase_Amount
1          102              600
0          101              520
2          103              480
```

Answer

```
import pandas as pd

data = {
    'Customer_ID'    : [101, 102, 103, 101, 104, 102, 101, 105, 102, 103],
    'Purchase_Amount': [200, 150, 180, 220, 300, 200, 100, 400, 250, 300]
}

df = pd.DataFrame(data)

# Total Purchases per Customer
total_purchases = df.groupby('Customer_ID')['Purchase_Amount'].sum().reset_index()

print("Total Purchases per Customer:")
print(total_purchases)
```

```
# Getting "Top 3" Frequent Customers
top_customers = total_purchases.sort_values(by='Purchase_Amount',
ascending=False).head(3).reset_index(drop=True)

print("\nTop 3 Frequent Customers:")
print(top_customers)
```

Output:

```
Total Purchases per Customer:
   Customer_ID  Purchase_Amount
0          101              520
1          102              600
2          103              480
3          104              300
4          105              400

Top 3 Frequent Customers:
   Customer_ID  Purchase_Amount
0          102              600
1          101              520
2          103              480
```

Question 2:

A company tracks the daily sales of a product over a month. You are tasked with identifying any abnormal sales data using the IQR (Interquartile Range) method.

Dataset (Daily Sales in Units):

```
data = {'Day': range(1, 31),
        'Sales': [25, 30, 28, 45, 55, 60, 22, 80, 95, 120,
                  33, 29, 27, 35, 40, 50, 85, 110, 105, 92,
                  30, 34, 31, 33, 36, 42, 44, 48, 90, 200]}
```

Tasks:

1. Calculate the Q1 (25th percentile) and Q3 (75th percentile).

2. Determine the IQR.

3. Identify the Lower Bound and Upper Bound.

4. Detect and display the outliers.

5. Replace the Outliers with the Median Value.

Expected Output:

Q1: 31.5, Q3: 83.75, IQR: 52.25

Lower Bound: -46.875, Upper Bound: 162.125

Outliers Detected:

	Day	Sales
29	30	200

Data After Replacing Outliers with Median:

	Day	Sales
0	1	25.0
1	2	30.0
2	3	28.0
3	4	45.0
4	5	55.0

5	6	60.0
6	7	22.0
7	8	80.0
8	9	95.0
9	10	120.0
10	11	33.0
11	12	29.0
12	13	27.0
13	14	35.0
14	15	40.0
15	16	50.0
16	17	85.0
17	18	110.0
18	19	105.0
19	20	92.0
20	21	30.0
21	22	34.0
22	23	31.0
23	24	33.0
24	25	36.0
25	26	42.0
26	27	44.0
27	28	48.0
28	29	90.0
29	30	43.0

Answer

```
import pandas as pd
import numpy as np

data = {'Day': range(1, 31),
        'Sales': [25, 30, 28, 45, 55, 60, 22, 80, 95, 120,
                  33, 29, 27, 35, 40, 50, 85, 110, 105, 92,
                  30, 34, 31, 33, 36, 42, 44, 48, 90, 200]}
df = pd.DataFrame(data)

# Calculate Q1 and Q3
q1 = np.percentile(df['Sales'], 25)#OR q1 = df.describe()["Sales"]["25%"]

q3 = np.percentile(df['Sales'], 75)#OR q3 = df.describe()["Sales"]["75%"]

# Calculate IQR
iqr = q3 - q1

# Calculate lower and upper bounds for outliers
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr

# Detect outliers
outliers = df[(df['Sales'] < lower_bound) | (df['Sales'] > upper_bound)]

# Calculate median to replace outliers
median = np.median(df['Sales'])    #or df.describe()["Sales"].median

# Replace outliers with median
# Locate the particular from the "Sales" dataset where it is greater than
the upper_bound value.  And assign this row with the "median" value:
df.loc[df['Sales'] > upper_bound, 'Sales'] = median

print(f"Q1: {q1}, Q3: {q3}, IQR: {iqr}")
print(f"Lower Bound: {lower_bound}, Upper Bound: {upper_bound}")
print("Outliers Detected:")
print(outliers)
print("Data After Replacing Outliers with Median:")
print(df)
```

Question 3:

A pharmaceutical company is testing the effectiveness of a new drug to reduce blood pressure. Two groups of patients were selected:

Group 1 (Treatment): Received the drug

Group 2 (Control): Received a placebo

The company wants to check if there is a significant difference in the blood pressure levels between the two groups using an Independent T-Test.

Dataset:

```
data = {'Group': ['Treatment']*10 + ['Control']*10,

        'Blood_Pressure': [120, 115, 118, 123, 122, 119, 124, 117, 116, 121,

                           130, 135, 140, 138, 142, 136, 139, 134, 137,
141]}
```

Tasks:

1. Perform an Independent T-Test.
2. State the null and alternative hypotheses.
3. Calculate the p-value.
4. Conclude whether the drug has a significant effect.

Expected Output:

T-Statistic: -11.870553692962726

P-Value: 6.008066605173374e-10

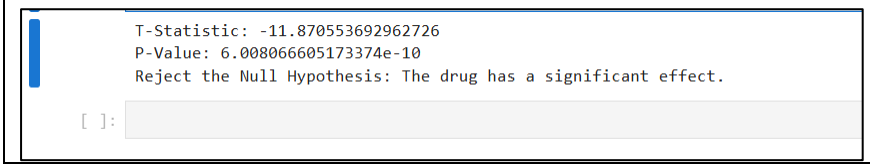
Reject the Null Hypothesis: The drug has a significant effect.

Answer

```
import pandas as pd
from scipy.stats import ttest_ind

data = {'Group': ['Treatment']*10 + ['Control']*10,
        'Blood_Pressure': [120, 115, 118, 123, 122, 119, 124, 117, 116, 121,
                           130, 135, 140, 138, 142, 136, 139, 134, 137, 141]}

# Create DataFrame
df = pd.DataFrame(data)

# Separate groups
```

```
group1 = df[df['Group'] == 'Treatment']['Blood_Pressure']
group2 = df[df['Group'] == 'Control']['Blood_Pressure']

# Null Hypothesis: No difference in blood pressure between groups
# Alternative Hypothesis: There is a difference

# Perform Independent T-Test
stat, p_value = ttest_ind(group1, group2)

print(f"T-Statistic: {stat}")
print(f"P-Value: {p_value}")

if p_value < 0.05:
    print("Reject the Null Hypothesis: The drug has a significant effect.")
else:
    print("Fail to Reject the Null Hypothesis: No significant effect.")
```

Output:



Question 4:

GlobalMart is a large retailer conducting advertisement campaigns in different regions. The company spends money on two types of advertisements: **TV Ads and Social Media Ads**

They want to analyze how these ads influence their sales.
Your task is to calculate the **Covariance** and **Correlation** to determine which type of ad has a stronger impact on sales.

Tasks :

1. Calculate the **Covariance** between ad budgets and sales to measure the direction of the relationship.
2. Calculate the **Correlation** to measure the strength of the relationship.
3. Determine which type of ad is more effective for increasing sales.

Dataset

Region	TV_Ad_Budget	Social_Media_Budget	Sales
North	200	150	20
South	300	250	35
East	400	300	50
West	500	450	60
Central	600	500	80

Expected Output:

Covariance (TV vs Sales): 3625.0

Covariance (Social Media vs Sales): 3225.0

Correlation (TV vs Sales): 0.9958640886279954

Correlation (Social Media vs Sales): 0.9724846021568381

TV Ads have a stronger impact on Sales.

Answer

```
import numpy as np
import pandas as pd

data = {
    'TV_Ad': [200, 250, 300, 350, 400],
    'Social_Media_Ad': [150, 190, 230, 280, 310],
    'Sales': [2200, 2700, 3200, 3700, 4200]
}
```

```
df = pd.DataFrame(data)

# Calculate Covariance between TV Ads and Sales
cov_tv_sales      = np.cov(df['TV_Ad'], df['Sales'])[0, 1]

# Calculate Covariance between Social Media Ads and Sales
cov_social_sales  = np.cov(df['Social_Media_Ad'], df['Sales'])[0, 1]

# Calculate Correlation between TV Ads and Sales
corr_tv_sales     = np.corrcoef(df['TV_Ad'], df['Sales'])[0, 1]

# Calculate Correlation between Social Media Ads and Sales
corr_social_sales = np.corrcoef(df['Social_Media_Ad'], df['Sales'])[0, 1]

print(f"Covariance (TV vs Sales): {cov_tv_sales}")
print(f"Covariance (Social Media vs Sales): {cov_social_sales}")
print(f"Correlation (TV vs Sales): {corr_tv_sales}")
print(f"Correlation (Social Media vs Sales): {corr_social_sales}")

if corr_tv_sales > corr_social_sales:
    print("TV Ads have a stronger impact on Sales.")
else:
    print("Social Media Ads have a stronger impact on Sales.")
```

Output

```
Covariance (TV vs Sales): 62500.0
Covariance (Social Media vs Sales): 51250.0
Correlation (TV vs Sales): 0.9999999999999999
Correlation (Social Media vs Sales): 0.9979243861980327
TV Ads have a stronger impact on Sales.
```

Question 5:

A company tracks the delivery time (in minutes) for its online orders. You are given the delivery times for 50 orders.

- Calculate the mean and standard deviation of the delivery times.
- Plot the Probability Density Function (PDF) to visualize the distribution.

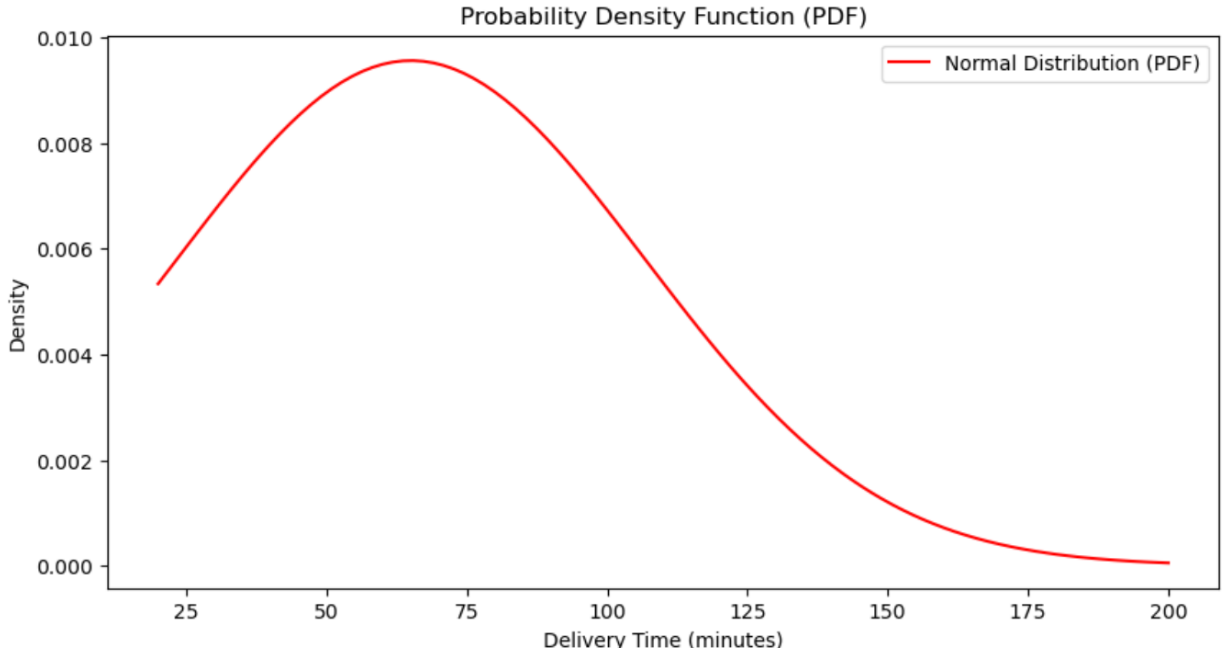
Dataset (Delivery Times in Minutes):

```
[25, 30, 28, 45, 55, 60, 22, 80, 95, 120, 33, 29, 27, 35, 40, 50, 85,
110, 105, 92, 30, 34, 31, 33, 36, 42, 44, 48, 90, 200, 20, 25, 27, 32,
38, 41, 47, 58, 62, 77, 80, 84, 90, 110, 123, 145, 150, 160]
```

You can try implementing this using libraries like `numpy`, `matplotlib`, and `scipy`

Expected Output:

```
Mean Delivery Time: 65.0625
Standard Deviation of Delivery Time: 41.718504212759115
```



Answer

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

data = [25, 30, 28, 45, 55, 60, 22, 80, 95, 120,
```

```
        33, 29, 27, 35, 40, 50, 85, 110, 105, 92,
        30, 34, 31, 33, 36, 42, 44, 48, 90, 200,
        20, 25, 27, 32, 38, 41, 47, 58, 62, 77,
        80, 84, 90, 110, 123, 145, 150, 160]

# Calculate mean and standard deviation
mean_delivery = np.mean(data)
std_delivery  = np.std(data)

print(f"Mean Delivery Time: {mean_delivery}")
print(f"Standard Deviation of Delivery Time: {std_delivery}")

# Get Plot Probability Density Function (PDF)
x = np.linspace(min(data), max(data), 100)
pdf = norm.pdf(x, mean_delivery, std_delivery)

plt.plot(x, pdf, color='blue', label='PDF')
plt.hist(data, bins=20, density=True, alpha=0.5, color='gray')
plt.title('Delivery Time Distribution')
plt.xlabel('Delivery Time (minutes)')
plt.ylabel('Probability Density')
plt.legend()
plt.show()
```

Output

```
Mean Delivery Time: 65.0625
Standard Deviation of Delivery Time: 41.718504212759115
```


```
[ ]:
```