In [1]:
```python
# problem 2
# import and load data
import pandas as pd
import numpy as np
```

In [5]:
```python
# original data
customer_data =[
    {'id': 'C001', 'name': 'John Doe', 'email': 'JOHN.DOE@email.com', 'age': 30, 'country': 'USA'},
    {'id': 'C002', 'name': 'Jane Smith ', 'email': 'jane.smith@email.co', 'age': '25', 'country': 'canada'},
    {'id': 'C003', 'name': ' Peter Jones', 'email': None, 'age': 45, 'country': 'UK'},
    {'id': 'C004', 'name': 'Alice', 'email': 'alice@email.com', 'age': 'thirty-two', 'country': '  USA'},
    {'id': 'C005', 'name': 'Mike Brown', 'email': 'mike.brown@email.com', 'age': 35, 'country': 'USA'},
    {'id': 'C006', 'name': 'Linda', 'email': 'linda@email.com', 'age': 28, 'country': 'usa'},
]
```

In [7]:
```python
df = pd.DataFrame(customer_data)
df
```

Out[7]:

	id	name	email	age	country
0	C001	John Doe	JOHN.DOE@email.com	30	USA
1	C002	Jane Smith	jane.smith@email.co	25	canada
2	C003	Peter Jones	None	45	UK
3	C004	Alice	alice@email.com	thirty-two	USA
4	C005	Mike Brown	mike.brown@email.com	35	USA
5	C006	Linda	linda@email.com	28	usa

In [8]:
```python
# clean name and county
df['name'] = df['name'].str.strip().str.title()
df['country'] = df['country'].str.strip().str.title()
```

In [9]:
```python
# standarsize country names
country_map= {'Usa': 'USA',
    'United States': 'USA',
    'Canada': 'Canada',
    'Uk': 'UK'}
df['country'] = df['country'].replace( country_map)
df
```

Out[9]:

	id	name	email	age	country
0	C001	John Doe	JOHN.DOE@email.com	30	USA
1	C002	Jane Smith	jane.smith@email.co	25	Canada
2	C003	Peter Jones	None	45	UK
3	C004	Alice	alice@email.com	thirty-two	USA
4	C005	Mike Brown	mike.brown@email.com	35	USA
5	C006	Linda	linda@email.com	28	USA

In [11]:
```python
# handle missing emails
df['email']=df['email'].fillna('No Email Provided')
df
```

Out[11]:

	id	name	email	age	country
0	C001	John Doe	JOHN.DOE@email.com	30	USA
1	C002	Jane Smith	jane.smith@email.co	25	Canada
2	C003	Peter Jones	No Email Provided	45	UK
3	C004	Alice	alice@email.com	thirty-two	USA
4	C005	Mike Brown	mike.brown@email.com	35	USA
5	C006	Linda	linda@email.com	28	USA

In [12]:
```python
# fix age column
def convert_age(x):
    try:
        return int(x)
    except:
        return np.nan
```

```
df['age']= df['age'].apply(convert_age)
# fill invalid ages with average
mean_age = df['age'].mean(skipna=True)
df['age']=df['age'].fillna(round(mean_age)).astype(int)
df
```

Out[12]:

	id	name	email	age	country
0	C001	John Doe	JOHN.DOE@email.com	30	USA
1	C002	Jane Smith	jane.smith@email.co	25	Canada
2	C003	Peter Jones	No Email Provided	45	UK
3	C004	Alice	alice@email.com	33	USA
4	C005	Mike Brown	mike.brown@email.com	35	USA
5	C006	Linda	linda@email.com	28	USA

In [13]:
```
# Extract domain
df['domain'] = df['email'].apply(
    lambda x: x.split('@')[-1] if '@' in x else 'No Domain'
)
df
```

Out[13]:

	id	name	email	age	country	domain
0	C001	John Doe	JOHN.DOE@email.com	30	USA	email.com
1	C002	Jane Smith	jane.smith@email.co	25	Canada	email.co
2	C003	Peter Jones	No Email Provided	45	UK	No Domain
3	C004	Alice	alice@email.com	33	USA	email.com
4	C005	Mike Brown	mike.brown@email.com	35	USA	email.com
5	C006	Linda	linda@email.com	28	USA	email.com

In [14]:
```
# Export cleaned DataFrame to CSV
df.to_csv("cleaned_customer_data.csv", index=False)

print("✅ Cleaned data has been saved as 'cleaned_customer_data.csv'")
```

```
✅ Cleaned data has been saved as 'cleaned_customer_data.csv'
```

Problem 3. You are given a list of integers that represent daily stock prices. Your task is to find the maximum profit that can be made by buying and selling the stock. You can only complete at most one transaction (i.e., buy one and sell one share of the stock). Example: • prices = [7, 1, 5, 3, 6, 4] • Output: 5 (Buy on day 2 at price 1 and sell on day 5 at price 6) Write a Python function max_profit(prices) that takes the list of prices and returns the maximum profit.

In [22]:
```
def max_profit(prices):
    if not prices or len(prices) <2:
        return 0
    min_price= prices[0]
    max_profit= 0

    for price in prices[1:]:
        profit= price - min_price
        max_profit = max(max_profit, profit)
        min_price = min(min_price, price)
    return max_profit
```

In [24]:
```
prices = [7, 1, 5, 3, 6, 4]
result = max_profit(prices)
result
```

Out[24]:
```
5
```

In [25]:
```
print(max_profit([7, 6, 4, 3, 1]))   # Output: 0 (no profit possible)
print(max_profit([2, 4, 1]))         # Output: 2 (buy at 2, sell at 4)
print(max_profit([3, 3, 3, 3]))      # Output: 0 (no change)
```

```
0
2
0
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js