Test/Check Your Pandas Skills

Custom Aggregation with groupby()

- You have a DataFrame df with sales data: date, product, sales. Write code to group by product, aggregate total sales for - each product, and calculate the mean, median, and standard deviation of sales per day.

In [ ]:
```
import pandas as pd
df = pd.DataFrame({
    'date': pd.date_range('2023-01-01', periods=12, freq='D'),
    'product': ['A', 'A', 'B', 'B', 'C', 'A', 'C', 'C', 'B', 'A', 'B', 'C'],
    'sales': [5, 9, 3, 7, 6, 2, 10, 1, 4, 8, 6, 11]
})

# Your solution here------------------------------

```

Pivot Table with Complex Indexing

- Create a pivot table where country and city are indices, year is a column, and population values are aggregated using the - sum function. Make sure that missing values are replaced by 0.

In [ ]:
```
df = pd.DataFrame({
    'country': ['USA', 'USA', 'Canada', 'Canada', 'Mexico', 'Mexico'],
    'city': ['New York', 'San Francisco', 'Toronto', 'Vancouver', 'Mexico City', 'Cancun'],
    'year': [2020, 2021, 2020, 2021, 2020, 2021],
    'population': [8.6, 7.6, 2.7, 2.5, 21.7, 0.9]
})

# Your solution here------------------------------

```

Filtering Based on Complex Conditions

- Filter the DataFrame to get rows where sales > 150 and profit > 50, but also include rows where region is 'East' regardless of sales or profit.

In [ ]:
```
df = pd.DataFrame({
    'region': ['North', 'South', 'East', 'West', 'North', 'East'],
    'sales': [100, 200, 150, 300, 250, 120],
    'profit': [30, 50, 40, 60, 70, 20]
})
# Your solution here------------------------------


```

Merging DataFrames with Multiple Keys

- Perform a merge to find the GDP information for cities that appear in both df1 and df2. Also, return cities from df1 even if they don't exist in df2, filling missing GDPs with NaN

In [ ]:
```
df1 = pd.DataFrame({
    'state': ['CA', 'TX', 'NY'],
    'city': ['San Francisco', 'Dallas', 'New York'],
    'population': [884, 1340, 862]
})

df2 = pd.DataFrame({
    'state': ['CA', 'TX', 'NY', 'TX'],
    'city': ['Los Angeles', 'Austin', 'Buffalo', 'Dallas'],
    'gdp': [1043, 477, 176, 512]
})

# Your solution here------------------------------
```

Apply Function on Multiple Columns

- Write a custom function that sums the squares of the values across columns A, B, and C row-wise, and apply it to get a new column 'sum_squares'.

In [ ]:
```
df = pd.DataFrame({
    'A': [1, 2, 3, 4],
    'B': [5, 6, 7, 8],
    'C': [9, 10, 11, 12]
})
# Your solution here------------------------------


```

Hierarchical Indexing and Slicing

- Write code to slice the DataFrame and get sales data for all cities in the North region.

In [ ]:

```
df = pd.DataFrame({
    'region': ['North', 'North', 'South', 'South'],
    'city': ['Boston', 'Boston', 'Miami', 'Miami'],
    'year': [2020, 2021, 2020, 2021],
    'sales': [100, 150, 200, 180]
}).set_index(['region', 'city', 'year'])

# Your solution here------------------------------
```