

Efficient Summaries with Custom Functions (Final Version)

This document includes the question, the solution, and a breakdown of the code provided in the screenshot.

Uploaded Screenshot

Below is the screenshot of the task:

The screenshot shows a web browser window displaying a DataCamp exercise titled "Efficient summaries". The exercise is part of the "Data Manipulation with pandas" course. The left sidebar contains the exercise title, a brief description of the task, and instructions. The main content area shows a code editor with a Python script. The script defines a custom function `iqr` and uses it to calculate the IQR of the `temperature_c` column and the median of `fuel_price_usd_per_l` and `unemployment`.

Exercise: Efficient summaries

While pandas and NumPy have tons of functions, sometimes, you may need a different function to summarize your data.

The `.agg()` method allows you to apply your own custom functions to a DataFrame, as well as apply functions to more than one column of a DataFrame at once, making your aggregations super-efficient. For example,

```
df['column'].agg(function)
```

In the custom function for this exercise, "IQR" is short for inter-quartile range, which is the 75th percentile minus the 25th percentile. It's an alternative to standard deviation that is helpful if your data contains outliers.

`sales` is available and `pandas` is loaded as `pd`.

Instructions 3/3 30 XP

- ✓ Use the custom `iqr` function defined for you along with `.agg()` to print the IQR of the `temperature_c` column of `sales`.
- ✓ Update the column selection to use the custom `iqr` function with `.agg()` to print the IQR of `temperature_c`, `fuel_price_usd_per_l`, and `unemployment`, in that order.
- ✗ Update the aggregation functions called by `.agg()`: include `iqr` and `np.median` in that order.

[Take Hint \(-9 XP\)](#)

```
script.py
1 # Import NumPy and create custom IQR function
2 import numpy as np
3 def iqr(column):
4     return column.quantile(0.75) - column.quantile(0.25)
5
6 # Update to print IQR and median of temperature_c, fuel_price_usd_per_l, & unemployment
7 print(sales[['temperature_c', 'fuel_price_usd_per_l', 'unemployment']].agg(iqr))
```

Python Shell Slides

```
<script.py> output:
temperature_c      16.583
fuel_price_usd_per_l  0.073
unemployment       0.565
dtype: float64

In [1]:
```

Question

1. Use the custom `iqr` function defined for you along with `.agg()` to print the IQR of the `temperature_c` column of `sales`.
2. Update the column selection to use the custom `iqr` function with `.agg()` to print the IQR of `temperature_c`, `fuel_price_usd_per_l`, and `unemployment`, in that order.
3. Update the aggregation functions called by `.agg()`: include `iqr` and `np.median` in that order.

Answer

```
# Import NumPy
import numpy as np
```

```
# A custom IQR function
def iqr(column):
```

```
return column.quantile(0.75) - column.quantile(0.25)

# Print IQR and median of temperature_c, fuel_price_usd_per_l, and
unemployment
print(sales[['temperature_c', 'fuel_price_usd_per_l',
'unemployment']].agg([iqr, np.median]))
```

Code Explanation

Explanation of the code:

1. ``import numpy as np``: Imports the NumPy library, which is necessary to use the ``np.median`` function.
2. ``def iqr(column):``: Defines a custom function ``iqr`` to calculate the interquartile range (IQR) of a column by subtracting the 25th percentile from the 75th percentile.
3. ``sales[['temperature_c', 'fuel_price_usd_per_l', 'unemployment']].agg([iqr, np.median])``: Applies both the ``iqr`` and ``np.median`` aggregation functions to the selected columns (``temperature_c``, ``fuel_price_usd_per_l``, and ``unemployment``) in the ``sales`` DataFrame and prints the results in the specified order.