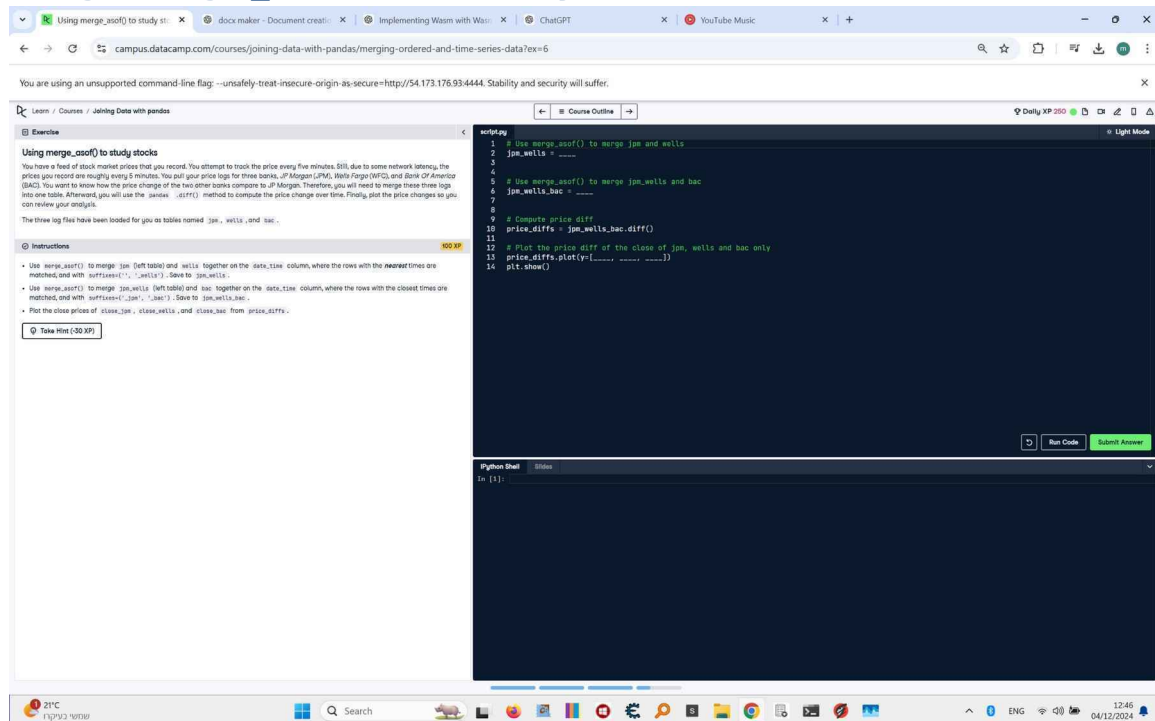


Using merge_asof() to study stocks - Corrected



You are using an unsupported command-line flag: --unsafely-treat-insecure-origin-as-secure=http://54.173.176.93:4444. Stability and security will suffer.

Learn / Courses / Joining Data with pandas

Exercise

Using merge_asof() to study stocks

You have a list of stock market prices that you record. You attempt to track the price every five minutes. Still, due to some network latency, the prices you record are roughly every 5 minutes. You put your price logs for three banks, JP Morgan (JPM), Wells Fargo (WFC), and Bank of America (BAC). You want to know how the price change of the two other banks compare to JP Morgan. Therefore, you will need to merge these three logs into one table. Afterward, you will use the `.diff()` method to compute the price change over time. Finally, plot the price changes to gain some review your analysis.

The three log files have been loaded for you as tables named `jpm`, `wells`, and `bac`.

Instructions

- Use `merge_asof()` to merge `jpm` (left table) and `wells` together on the `date_time` column, where the rows with the nearest times are matched, and with suffixes `('_jpm', '_wells')`. Save to `jpm_wells`.
- Use `merge_asof()` to merge `jpm_wells` (left table) and `bac` together on the `date_time` column, where the rows with the closest times are matched, and with suffixes `('_jpm', '_bac')`. Save to `jpm_wells_bac`.
- Plot the close prices of `close_jpm`, `close_wells`, and `close_bac` from `jpm_wells_bac`.

Take Hint (30 XP)

```
1 # Use merge_asof() to merge jpm and wells
2 jpm_wells = pd.merge_asof(
3     jpm, wells,
4     on='date_time',
5     direction='nearest',
6     suffixes=('_jpm', '_wells')
7 )
8
9 # Compute price diff
10 price_diffs = jpm_wells_bac.diff()
11
12 # Plot the price diff of the close of jpm, wells and bac only
13 price_diffs.plot()
14 plt.show()
```

Python Shell

In [1]:

Question:

Use `merge_asof()` to merge `jpm` and `wells` together on the `date_time` column, where the rows with the nearest times are matched, and with suffixes `('_jpm', '_wells')`. Save to `jpm_wells`. Then use `merge_asof()` to merge `jpm_wells` and `bac` together on the `date_time` column, where the rows with the closest times are matched, and with suffixes `('_jpm', '_bac')`. Save to `jpm_wells_bac`. Finally, compute the price change using the `.diff()` method and plot the close prices of `jpm`, `wells`, and `bac` from the resulting dataframe.

Answer:

```
# Use merge_asof() to merge jpm and wells with direction='nearest'
```

```
jpm_wells = pd.merge_asof(
    jpm, wells,
    on='date_time',
    suffixes=('_', '_wells'),
    direction='nearest'
)
```

```
# Use merge_asof() to merge jpm_wells and bac with direction='nearest'
```

```
jpm_wells_bac = pd.merge_asof(
    jpm_wells, bac,
    on='date_time',
```

```

    suffixes=('_jpm', '_bac'),
    direction='nearest'
)

# Compute price differences
price_diffs = jpm_wells_bac.diff()

# Plot the price differences of close_jpm, close_wells, and close_bac
price_diffs.plot(
    y=['close_jpm', 'close_wells', 'close_bac']
)
plt.show()

```

Code Explanation:

1. `jpm_wells = pd.merge_asof(..., direction='nearest')`:
This line merges the `jpm` and `wells` dataframes on the `'date_time'` column using the `'merge_asof'` function with the `direction` parameter set to `'nearest'`. This ensures that rows with the nearest timestamp are matched, and suffixes are applied to differentiate columns.
2. `jpm_wells_bac = pd.merge_asof(..., direction='nearest')`:
This line merges the `jpm_wells` dataframe with the `bac` dataframe on the `'date_time'` column using the `'merge_asof'` function with `direction='nearest'`. This combines the three dataframes by aligning the nearest timestamps.
3. `price_diffs = jpm_wells_bac.diff()`:
This line computes the difference between consecutive rows for each column in the dataframe, calculating the changes in close prices over time.
4. `price_diffs.plot(...)`:
This line creates a plot of the close prices of `jpm`, `wells`, and `bac`. The `'y'` parameter specifies the columns to plot, and `plt.show()` displays the visualization.