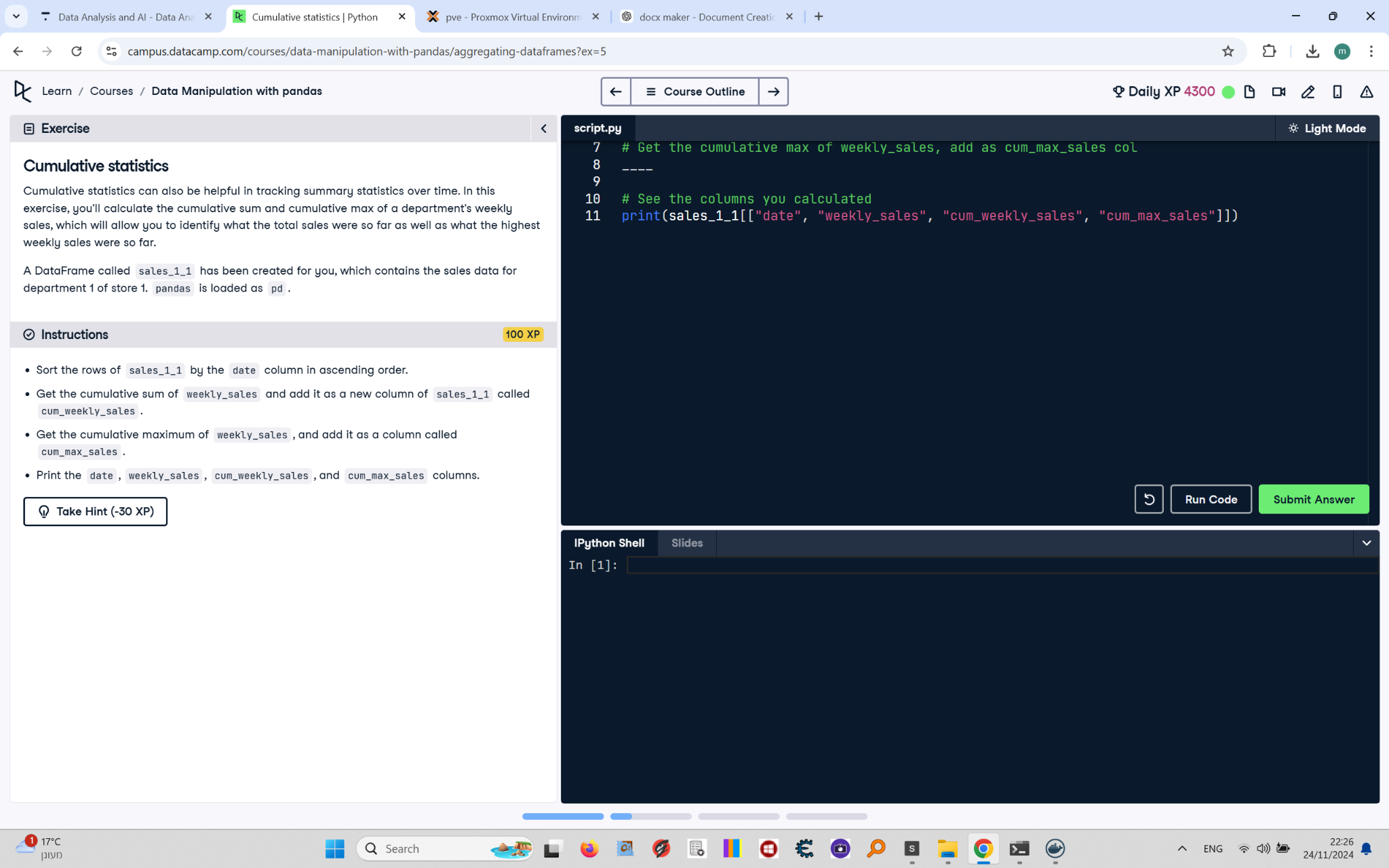
# Cumulative Statistics (Solution)

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:



## Question

1. Sort the rows of `sales\_1\_1` by the `date` column in ascending order.  
2. Get the cumulative sum of `weekly\_sales` and add it as a new column of `sales\_1\_1` called `cum\_weekly\_sales`.  
3. Get the cumulative maximum of `weekly\_sales` and add it as a column called `cum\_max\_sales`.  
4. Print the `date`, `weekly\_sales`, `cum\_weekly\_sales`, and `cum\_max\_sales` columns.

## Answer

# Sort rows by date  
sales\_1\_1 = sales\_1\_1.sort\_values('date')  
  
# Get the cumulative sum of weekly\_sales  
sales\_1\_1['cum\_weekly\_sales'] = sales\_1\_1['weekly\_sales'].cumsum()  
  
# Get the cumulative maximum of weekly\_sales  
sales\_1\_1['cum\_max\_sales'] = sales\_1\_1['weekly\_sales'].cummax()  
  
# Print the specified columns  
print(sales\_1\_1[['date', 'weekly\_sales', 'cum\_weekly\_sales', 'cum\_max\_sales']])

## Code Explanation

# Explanation of the code:

1. `sales\_1\_1.sort\_values('date')`: Sorts the `sales\_1\_1` DataFrame by the `date` column in ascending order.

2. `sales\_1\_1['cum\_weekly\_sales'] = sales\_1\_1['weekly\_sales'].cumsum()`: Calculates the cumulative sum of the `weekly\_sales` column and adds it as a new column called `cum\_weekly\_sales`.

3. `sales\_1\_1['cum\_max\_sales'] = sales\_1\_1['weekly\_sales'].cummax()`: Calculates the cumulative maximum of the `weekly\_sales` column and adds it as a new column called `cum\_max\_sales`.

4. `print(sales\_1\_1[['date', 'weekly\_sales', 'cum\_weekly\_sales', 'cum\_max\_sales']])`: Prints the `date`, `weekly\_sales`, `cum\_weekly\_sales`, and `cum\_max\_sales` columns from the `sales\_1\_1` DataFrame.