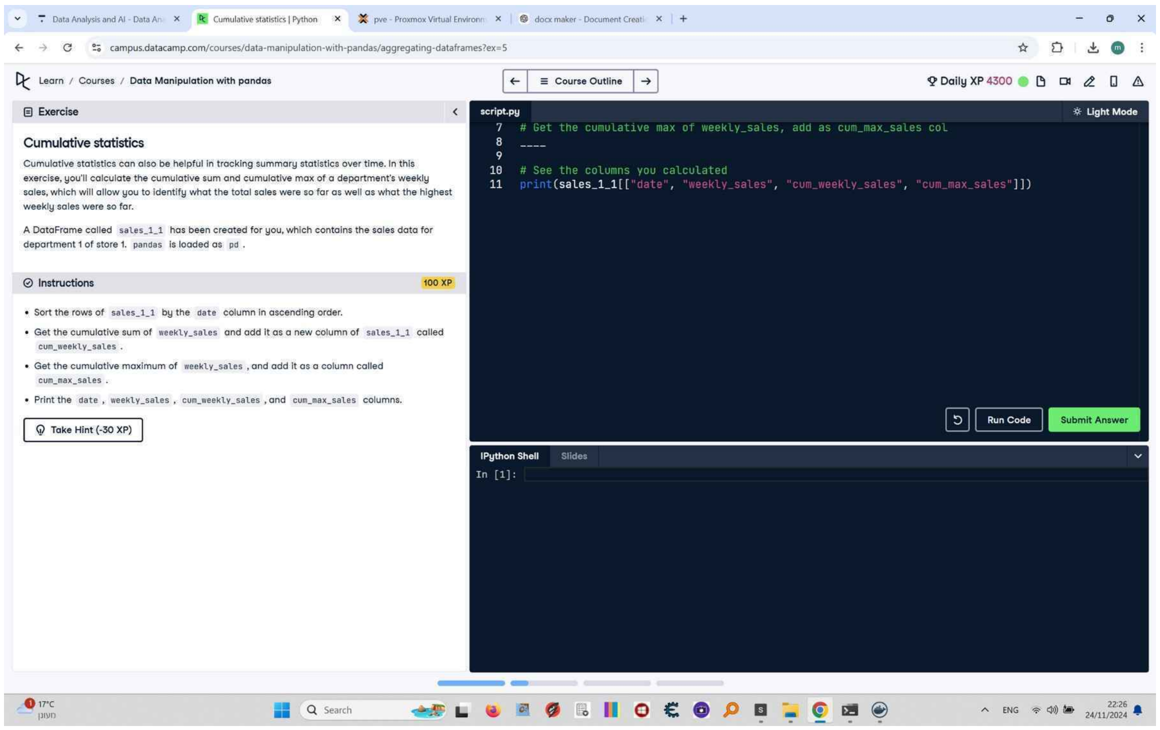
Cumulative Statistics (Solution)

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:



Data Analysis and AI - Data An
Cumulative statistics | Python
pve - Proxmox Virtual Environm
docx maker - Document Creati
campus.datacamp.com/courses/data-manipulation-with-pandas/aggregating-dataframes?ex=5
Learn / Courses / Data Manipulation with pandas
Course Outline
Daily XP 4300
Exercise
script.py
Light Mode
100 XP
Take Hint (-30 XP)
Run Code
Submit Answer
IPython Shell
Slides

Cumulative statistics

Cumulative statistics can also be helpful in tracking summary statistics over time. In this exercise, you'll calculate the cumulative sum and cumulative max of a department's weekly sales, which will allow you to identify what the total sales were so far as well as what the highest weekly sales were so far.

A DataFrame called `sales_1_1` has been created for you, which contains the sales data for department 1 of store 1. `pandas` is loaded as `pd`.

Instructions

- Sort the rows of `sales_1_1` by the `date` column in ascending order.
- Get the cumulative sum of `weekly_sales` and add it as a new column of `sales_1_1` called `cum_weekly_sales`.
- Get the cumulative maximum of `weekly_sales`, and add it as a column called `cum_max_sales`.
- Print the `date`, `weekly_sales`, `cum_weekly_sales`, and `cum_max_sales` columns.

```python
# Get the cumulative max of weekly_sales, add as cum_max_sales col
----

# See the columns you calculated
print(sales_1_1[["date", "weekly_sales", "cum_weekly_sales", "cum_max_sales"]])
```

```
In [1]:
```

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.