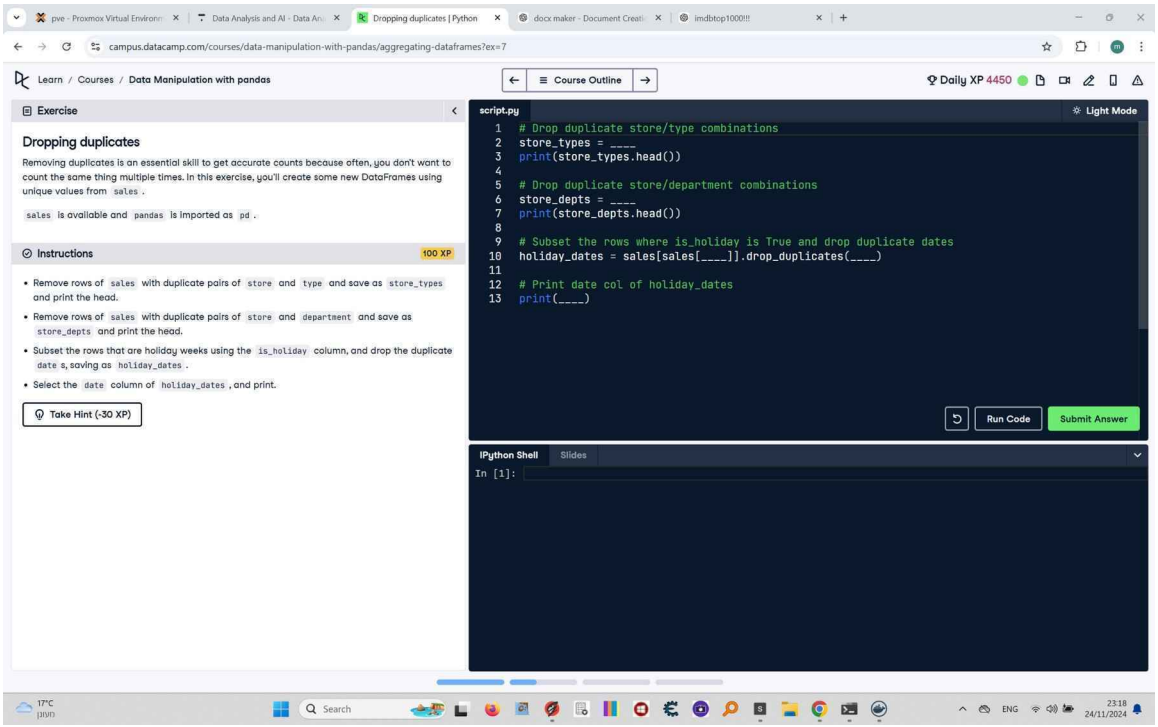
Dropping Duplicates (Updated Solution)

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

Uploaded Screenshot

Below is the updated screenshot of the task:



Question

1. Remove rows of `sales` with duplicate pairs of `store` and `type` and save as `store_types`, then print the head.
2. Remove rows of `sales` with duplicate pairs of `store` and `department` and save as `store_depts`, then print the head.
3. Subset the rows that are holiday weeks using the `is_holiday` column and drop duplicate `date`s, saving as `holiday_dates`.
4. Select the `date` column of `holiday_dates` and print.

Updated Answer

```
# Drop duplicate store/type combinations
store_types = sales.drop_duplicates(subset=['store', 'type'])
print(store_types.head())

# Drop duplicate store/department combinations
store_depts = sales.drop_duplicates(subset=['store', 'department'])
```

```
print(store_depts.head())

# Subset rows where is_holiday is True and drop duplicate dates
holiday_dates = sales[sales['is_holiday']].drop_duplicates(subset='date')

# Print date column of holiday_dates
print(holiday_dates['date'])
```

Code Explanation

Explanation of the code:

1. `sales.drop_duplicates(subset=['store', 'type'])`: Removes duplicate rows based on the combination of `store` and `type` columns.

2. `sales.drop_duplicates(subset=['store', 'department'])`: Removes duplicate rows based on the combination of `store` and `department` columns.

3. `sales[sales['is_holiday']]`: Filters the rows where the `is_holiday` column is `True`.

4. `.drop_duplicates(subset='date')`: Removes duplicate rows based on the `date` column within the filtered DataFrame.

5. `holiday_dates['date']`: Selects the `date` column from the `holiday_dates` DataFrame.