

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:

The screenshot displays a DataCamp exercise page for "Dropping duplicates". The page includes instructions and a code editor. The instructions are as follows:

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

The code editor shows the following Python code:

```
1 # Drop duplicate store/type combinations
2 store_types = sales.drop_duplicates(subset=['store', 'type'])
3 print(store_types.head())
4
5 # Drop duplicate store/department combinations
6 store_depts = sales.drop_duplicates(subset=['store', 'department'])
7 print(store_depts.head())
8
9 # Subset the rows where is_holiday is True and drop duplicate dates
10 holiday_dates = sales[sales['is_holiday'] == True].drop_duplicates(subset=['date'])
11
12 # Print date col of holiday_dates
13 print(holiday_dates['date'].head())
```

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.