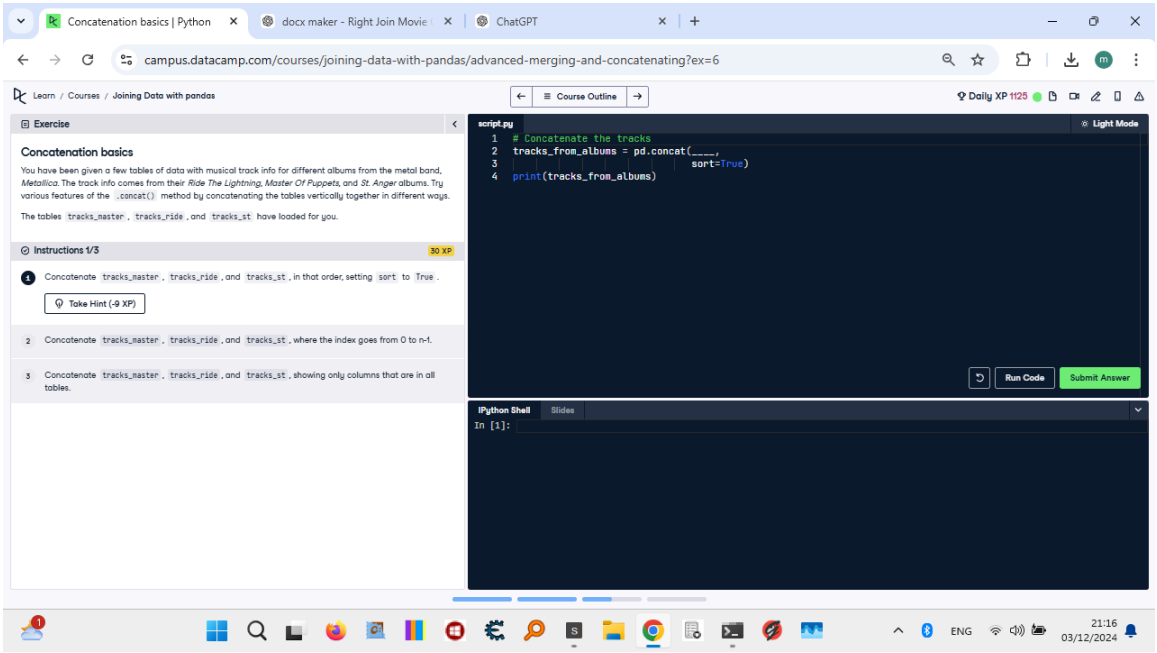
Concatenation Basics in Pandas - Final Solution



Screenshot showing the exercise context for concatenating tables in pandas.

Code Answer for Task 1:

```
# Concatenate the tracks tables, with sorting
tracks_from_albums = pd.concat([tracks_master, tracks_ride, tracks_st],
sort=True)

# Print the resulting concatenated DataFrame
print(tracks_from_albums)
```

Updated Code Answer for Task 2:

```
# Concatenate the tracks so the index goes from 0 to n-1
tracks_from_albums = pd.concat([tracks_master, tracks_ride, tracks_st],
                               ignore_index=True,
                               sort=True)

# Print the resulting concatenated DataFrame
print(tracks_from_albums)
```

Updated Code Answer for Task 3:

```
# Concatenate the tracks, show only column names that are in all tables
tracks_from_albums = pd.concat([tracks_master, tracks_ride, tracks_st],
                               join='inner',
                               sort=True)

# Print the resulting concatenated DataFrame
print(tracks_from_albums)
```

Explanation:

1. Task 1 uses the `pd.concat()` function to vertically concatenate the three DataFrames: 'tracks_master', 'tracks_ride', and 'tracks_st'. The `sort=True` parameter ensures that columns are sorted alphabetically.

2. Task 2 uses the `ignore_index=True` parameter in `pd.concat()` to reset the index in the concatenated DataFrame, resulting in a continuous numerical index starting from 0.

3. Task 3 uses the `join='inner'` parameter in `pd.concat()` to retain only the columns that are common across all three DataFrames, ensuring a consistent structure. The `sort=True` parameter further sorts the columns alphabetically.