```
import pandas as pd

wards = pd.read_csv('Wards_Offices.csv')
print(wards.head())
print(wards.shape)

census = pd.read_csv('Wards_Census.csv')
print(census.head())
print(census.shape)

wards_census = wards.merge(census, on='ward', suffixes=('_ward',
'_cen'))
print(wards_census.head(4))
print(wards_census.shape)

wards_census.columns

taxi_owners = pd.read_pickle('taxi_owners.p')
taxi_vehicles = pd.read_pickle('taxi_vehicles.p')
print(taxi_owners.head())
print(taxi_owners.shape)
print(taxi_vehicles.head())
print(taxi_vehicles.shape)

taxi_own_veh = taxi_owners.merge(taxi_vehicles, on='vid',
suffixes=('_own', '_veh'))
print(taxi_own_veh.shape)
print(taxi_own_veh.columns)

print(taxi_own_veh['fuel_type'].value_counts())

print(taxi_own_veh.value_counts('fuel_type'))

wards_altered = pd.read_csv('Wards_Offices_Altered.csv')
print(wards_altered.shape)

wards_census_altered = wards_altered.merge(census, on='ward',
suffixes=('_ward', '_cen'))
print(wards_census_altered.columns)
print(wards_census_altered.shape)

licenses = pd.read_csv('Business_Licenses.csv')
print(licenses.shape)
print(licenses.head())

ward_licenses = wards.merge(licenses, on='ward', suffixes=('_ward',
'_lic'))
print(ward_licenses.head())

print(wards.shape)
print(ward_licenses.shape)
```

```python
biz_owners = pd.read_pickle('business_owners.p')
licenses = pd.read_pickle('licenses.p')

print(biz_owners.columns)
print(biz_owners.shape)
print(licenses.columns)
print(licenses.shape)

licenses_owners = licenses.merge(biz_owners, on='account')

counted_df = licenses_owners.groupby('title').agg({'account': 'count'})

sorted_df = counted_df.sort_values('account', ascending=False)

print(sorted_df.head())

print(licenses.head())
print("-----------------------------------------")
print(wards.head())

# Step 1: Load the data
cal = pd.read_pickle('cta_calendar.p')
ridership = pd.read_pickle('cta_ridership.p')
stations = pd.read_pickle('stations.p')

# Step 1: Merge ridership with cal on year, month, day
# ridership_cal = ridership.merge(cal, on=['year', 'month', 'day'], how='left')
ridership_cal = ridership.merge(cal)

# Step 2: Merge with stations on station_id
# merged_df = ridership_cal.merge(stations, on='station_id', how='left')
merged_df = ridership.merge(cal, on=['year', 'month', 'day']) \
                        .merge(stations, on='station_id')

# Step 3: Filter criteria: Wilson station, weekdays in July
filter_criteria = (
    (merged_df['station_name'] == 'Wilson') &
    (merged_df['day_type'] == 'Weekday') &
    (merged_df['month'] == 7)
)

# Step 4: Sum the rides column
total_rides = merged_df.loc[filter_criteria, 'rides'].sum()

print("Total rides provided at Wilson station on weekdays in July:", total_rides)
```

```python
# Step 1: Load the data
licenses = pd.read_pickle('licenses.p')
wards = pd.read_pickle('ward.p')
zip_demo = pd.read_pickle('zip_demo.p')
print(licenses.columns)
print(wards.columns)
print(zip_demo.columns)

# Step 1: merge the three tables with KEY columns
licenses_zip_ward = licenses.merge(zip_demo, on='zip') \
                        .merge(wards, on='ward')

# Step 2: Group by 'alderman' and calculate the median income
median_income_by_alderman =
licenses_zip_ward.groupby('alderman').agg({'income': 'median'})

# Display the result
print(median_income_by_alderman)

movies = pd.read_csv('tmdb_movies.csv')
print(movies.head())
print(movies.shape)

tagLines = pd.read_csv('tmdb_taglines.csv')
print(tagLines.head())
print(tagLines.shape)

movies_tagLines = movies.merge(tagLines, on='id', how='left')
print(movies_tagLines.head())

# Step 1: Load the data
movies = pd.read_pickle('movies.p')
financials = pd.read_pickle('financials.p')

print(movies.columns)
print(financials.columns)

# Step 2: Merge the tables with a left join on 'id'
movies_financials = movies.merge(financials, on='id', how='left')

# Step 3: Count rows with null values in the 'budget' column
missing_budget_count = movies_financials['budget'].isnull().sum()

# Step 4: Display the result
print("Number of movies with missing budget data:",
missing_budget_count)

# Step 1: Load the data
toy_story = pd.read_csv('toy_story.csv')  # Toy Story movie data
taglines = pd.read_pickle('taglines.p')   # Taglines data
```

```python
# Step 2: Perform a left join
toystory_tag_left = toy_story.merge(taglines, on='id', how='left')

# Print results for the left join
print("Left Join Result:")
print(toystory_tag_left)
print(toystory_tag_left.shape)

# Step 3: Perform an inner join
toystory_tag_inner = toy_story.merge(taglines, on='id')

# Print results for the inner join
print("\nInner Join Result:")
print(toystory_tag_inner)
print(toystory_tag_inner.shape)

# Step 4: Observe the differences
print("\nNumber of rows (Left Join):", len(toystory_tag_left))
print("Number of rows (Inner Join):", len(toystory_tag_inner))

movie_to_genres = pd.read_csv('tdmb_movie_to_genres.csv')
tv_genre = movie_to_genres[movie_to_genres['genre'] == 'TV Movie']
print(tv_genre)

tv_movies = movies.merge(tv_genre, how='right', left_on='id',
right_on='movie_id')
print(tv_movies.head())

# Load the movies and genres data
movies = pd.read_pickle('movies.p')
movie_to_genres = pd.read_csv('tdmb_movie_to_genres.csv')

# Subset 'Science Fiction' movies
m = movie_to_genres['genre'] == 'Science Fiction'
scifi_movies = movie_to_genres[m]

# Subset 'Action' movies
m = movie_to_genres['genre'] == 'Action'
action_movies = movie_to_genres[m]

# Merge action_movies and scifi_movies with a right join
action_scifi = action_movies.merge(scifi_movies, on='movie_id',
how='right', suffixes=('_act', '_sci'))

# Print the resulting DataFrame to check
print(action_scifi.head())

# Subset rows where genre_act is null (only sci-fi movies)
scifi_only = action_scifi[action_scifi['genre_act'].isnull()]
```

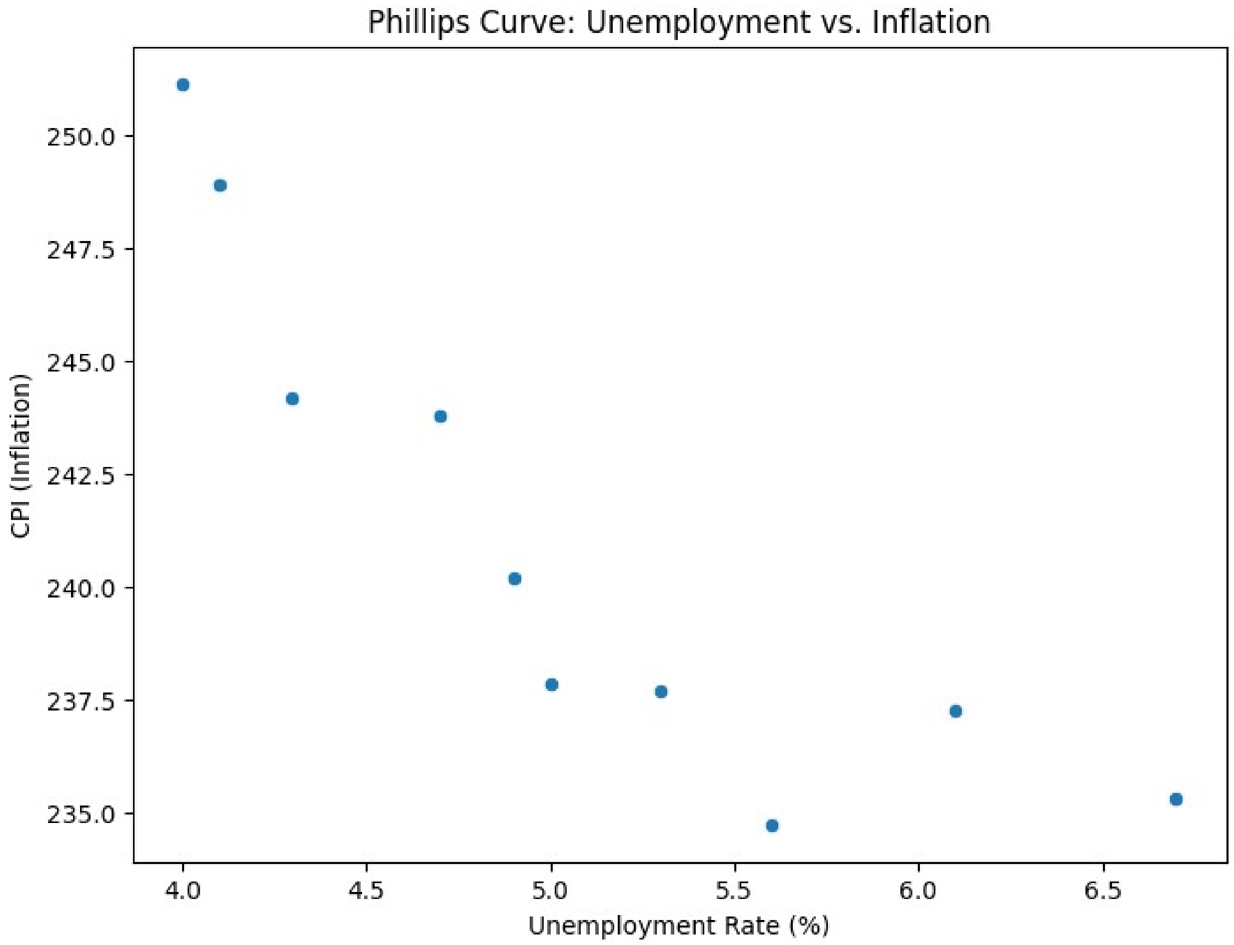
```
# Print to verify the result
print(scifi_only.head())

# Merge movies with scifi_only using an inner join
unique_scifi_movies = movies.merge(scifi_only, left_on='id',
right_on='movie_id', how='inner')

# Print the final result to see the movie names
print(unique_scifi_movies.head())

import matplotlib.pyplot as plt

# Load the data
pop_movies = pd.read_csv('pop_movies.csv')
movie_to_genres = pd.read_csv('tdmb_movie_to_genres.csv')

# Perform right join of movie_to_genres and pop_movies
genres_movies = pd.merge(movie_to_genres, pop_movies,
left_on='movie_id', right_on='id', how='right')

# Group by genre and count the number of movies
# genre_counts = genres_movies.groupby('genre')['id'].count()
genre_counts = genres_movies.groupby('genre').agg({'id':'count'})

# Create a bar plot
genre_counts.plot(kind='bar', label='id')
plt.title('Number of Popular Movies by Genre')
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.legend()
plt.tight_layout()
plt.show()

# Print the genre counts for reference
print(genre_counts)

sequels = pd.read_pickle('sequels.p')

original_sequels = sequels.merge(sequels, left_on='sequel',
right_on='id', suffixes=('_org', '_seq'))

print(original_sequels.head())

print(original_sequels[['title_org','title_seq']].head())

# Load the crews data
crews = pd.read_pickle('crews.p')

# Merge the crews table to itself using an inner join
crews_self_merged = crews.merge(crews, on='id', how='inner',
suffixes=('_dir', '_crew'))
```

```
# Create a boolean filter to select rows with 'Director' in the left
table and not 'Director' in the right table
boolean_filter = ((crews_self_merged['job_dir'] == 'Director') &
(crews_self_merged['job_crew'] != 'Director'))

# Apply the filter to the merged table
direct_crews = crews_self_merged[boolean_filter]

# Print the first few rows
print(direct_crews.head())

# Load the data
tracks_master = pd.read_csv('tracks_master.csv')
tracks_ride = pd.read_csv('tracks_ride.csv')
tracks_st = pd.read_csv('tracks_st.csv')

# 1. Concatenate the tables in order, setting sort to True
all_tracks_sorted = pd.concat([tracks_master, tracks_ride, tracks_st],
sort=True)
print("Concatenated tables in order, sorted:")
print(all_tracks_sorted.head())
print()

# 2. Concatenate the tables, resetting the index
all_tracks_reset = pd.concat([tracks_master, tracks_ride, tracks_st],
ignore_index=True)
print("Concatenated tables with reset index:")
print(all_tracks_reset.head())
print()

# 3. Concatenate the tables, keeping only common columns
all_tracks_common = pd.concat([tracks_master, tracks_ride, tracks_st],
axis=0, join='inner')
print("Concatenated tables with only common columns:")
print(all_tracks_common.head())
print()

# Load the data
inv_jul = pd.read_csv('inv_jul.csv')
inv_aug = pd.read_csv('inv_aug.csv')
inv_sep = pd.read_csv('inv_sep.csv')

# Concatenate the tables vertically with keys
avg_inv_by_month = pd.concat([inv_jul, inv_aug, inv_sep],
keys=['7Jul', '8Aug', '9Sep'])

# Calculate the average of the 'total' column, grouped by the keys
avg_invoice_total = avg_inv_by_month.groupby(level=0)['total'].mean()

# Create a bar chart
```

```python
avg_invoice_total.plot(kind='bar')
plt.title('Average Monthly Invoice Total')
plt.xlabel('Month')
plt.ylabel('Average Invoice Total')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Step 1: Merge gdp and sp500 using merge_ordered()
gdp = pd.read_csv('GDP.csv')
sp500 = pd.read_csv('S&P500.csv')

gdp_sp500 = pd.merge_ordered(gdp, sp500, left_on='year',
right_on='date', how='left')

print("gdp_sp500 DataFrame:")
print(gdp_sp500)

# Check the returns for 2018
print("\nReturns for 2018:", gdp_sp500.loc[gdp_sp500['year'] == 2018,
'returns'])

# Step 2: Merge gdp and sp500 using merge_ordered() with forward fill
gdp_sp500 = pd.merge_ordered(gdp, sp500, left_on='year',
right_on='date', how='left', fill_method='ffill')

# Step 3: Create gdp_returns DataFrame and compute correlation
gdp_returns = gdp_sp500[['gdp', 'returns']]
correlation_matrix = gdp_returns.corr()

print("\nCorrelation Matrix:")
print(correlation_matrix)

# Load the data
unemployment = pd.read_csv('unemployment.csv')
inflation = pd.read_csv('inflation.csv')

# Print the first few rows of both dataframes to inspect
print(unemployment.head())
print("----------------------------------")
print(inflation.head())

# Merge using merge_ordered with an inner join on the 'date' column
inflation_unemploy = pd.merge_ordered(inflation, unemployment,
on='date', how='inner')

# Print the resulting merged dataframe
print("----------------------------------")
print(inflation_unemploy.head())
```

```
        date  unemployment_rate
0  1/6/2013                7.5
1  1/1/2014                6.7
2  1/6/2014                6.1
3  1/1/2015                5.6
4  1/6/2015                5.3
----------------------------------
        date      cpi     seriesid                  data_type
0  1/1/2014  235.288  CUSR0000SA0  SEASONALLY ADJUSTED INDEX
1  1/2/2014  235.547  CUSR0000SA0  SEASONALLY ADJUSTED INDEX
2  1/3/2014  236.028  CUSR0000SA0  SEASONALLY ADJUSTED INDEX
3  1/4/2014  236.468  CUSR0000SA0  SEASONALLY ADJUSTED INDEX
4  1/5/2014  236.918  CUSR0000SA0  SEASONALLY ADJUSTED INDEX
----------------------------------
        date      cpi     seriesid                  data_type  \
0  1/1/2014  235.288  CUSR0000SA0  SEASONALLY ADJUSTED INDEX
1  1/1/2015  234.718  CUSR0000SA0  SEASONALLY ADJUSTED INDEX
2  1/1/2016  237.833  CUSR0000SA0  SEASONALLY ADJUSTED INDEX
3  1/1/2017  243.780  CUSR0000SA0  SEASONALLY ADJUSTED INDEX
4  1/1/2018  248.884  CUSR0000SA0  SEASONALLY ADJUSTED INDEX

   unemployment_rate
0                6.7
1                5.6
2                5.0
3                4.7
4                4.1
```

```python
import seaborn as sns

# Create a scatter plot
plt.figure(figsize=(8, 6))
sns.scatterplot(data=inflation_unemploy, x='unemployment_rate',
y='cpi')

# Set the title and labels
plt.title('Phillips Curve: Unemployment vs. Inflation')
plt.xlabel('Unemployment Rate (%)')
plt.ylabel('CPI (Inflation)')

# Show the plot
plt.show()

# Calculate the correlation between unemployment_rate and cpi
correlation = inflation_unemploy[['unemployment_rate', 'cpi']].corr()

# Print the correlation result
print(correlation)
```



```
                   unemployment_rate       cpi
unemployment_rate           1.000000 -0.868388
cpi                        -0.868388  1.000000
```