```
import pandas as pd
import numpy as np

homelessness = pd.read_csv('Class03_Data Manipulation Data
Sets/homelessness.csv', index_col=0)

homelessness.describe()
print(homelessness.shape)
display(homelessness.head())
homelessness.describe()
homelessness.shape
homelessness.values
homelessness.columns
homelessness.index

homelessness_ind = homelessness.sort_values(by='individuals',
ascending=True)
display(homelessness_ind.head())

homelessness_fam = homelessness.sort_values(by='family_members',
ascending=False)
display(homelessness_fam.head())

homelessness_reg_fam = homelessness.sort_values(by=['region',
'family_members'], ascending=[True, False])
display(homelessness_reg_fam.head())

state_fam = homelessness[['state', 'family_members']]
display(state_fam.head())

ind_gt_10k = homelessness[homelessness['individuals'] > 10000]
display(ind_gt_10k.head())
print('\nSorted Version')
ind_gt_10k = homelessness[homelessness['individuals'] >
10000].sort_values('individuals', ascending=True)

display(ind_gt_10k.head())

mountain_reg = homelessness[homelessness['region'] == 'Mountain']
display(mountain_reg.head(10))

fam_It_1k_pac = homelessness[(homelessness['family_members'] < 1000) &
(homelessness['region'] == 'Pacific')]
display(fam_It_1k_pac.head())

homeless_pop = homelessness[homelessness['state_pop'] < 2000000]
display(homeless_pop.head())

south_mid_atlantic = homelessness[homelessness['region'].isin(['South
Atlantic', 'Mid-Atlantic'])]
display(south_mid_atlantic.head(12))
```

```
mojave_homelessness =
homelessness[homelessness['state'].isin(['California', 'Nevada',
'Arizona', 'Utah'])]
display(mojave_homelessness.head())

homelessness['total'] = homelessness['individuals'] +
homelessness['family_members']
display(homelessness.head())

homelessness['p_individuals'] = homelessness['individuals'] /
homelessness['total']
display(homelessness.head())

homelessness['indiv_per_10k'] = homelessness['individuals'] /
homelessness['state_pop'] * 10000
high_homelessness = homelessness[homelessness['indiv_per_10k'] > 20]
high_homelessness_srt = high_homelessness.sort_values('indiv_per_10k',
ascending=False)
display(high_homelessness_srt[['state', 'indiv_per_10k']])

sales = pd.read_csv("Class03_Data Manipulation Data
Sets/sales_subset.csv",index_col=0)
# Print the head of the sales DataFrame
print(sales.head())
print('----------------------------------------------------------------
-------------')
# Print the info about the sales DataFrame
print(sales.info())
print('----------------------------------------------------------------
-------------')
# Print the mean of weekly_sales
print(sales['weekly_sales'].mean())
print('----------------------------------------------------------------
-------------')
# Print the median of weekly_sales
print(sales['weekly_sales'].median())
print('----------------------------------------------------------------
-------------')
# Print the maximum of the date column
print(sales['date'].max())
print('----------------------------------------------------------------
-------------')
# Print the minimum of the date column
print(sales['date'].min())

sales_1_1 = sales[(sales['department'] == 1) & (sales['store'] == 1)]
# Sort sales_1_1 by date
sales_1_1 = sales_1_1.sort_values('date', ascending = True)
# Get the cumulative sum of weekly_sales, add as cum_weekly_sales col
sales_1_1['cum_weekly_sales'] = sales['weekly_sales'].cumsum()
```

```
# Get the cumulative max of weekly_sales, add as cum_max_sales col
sales_1_1['cum_max_sales'] = sales['weekly_sales'].cummax()
# See the columns you calculated
print(sales_1_1[["date", "weekly_sales", "cum_weekly_sales",
"cum_max_sales"]])

store_type = sales.drop_duplicates(subset=['store', 'type'])
store_type.head()

store_depts = sales.drop_duplicates(subset=['store', 'department'])
store_depts.head()

holiday_dates = sales.drop_duplicates(subset='date')
[sales['is_holiday'] == True]
print(holiday_dates['date'])

# Count the number of stores of each store type
store_type_counts = store_type["type"].value_counts()
print(store_type_counts)
print('----------------------------------------------------------------
-------------')
# Count the proportion of stores of each store type
store_type_proportion =
store_type["type"].value_counts(normalize=True)
print(store_type_proportion)
print('----------------------------------------------------------------
-------------')
# Count the number of different departments, sorting the counts in
descending order
dept_counts =
store_depts["department"].value_counts().sort_values(ascending=False)
print(dept_counts)
print('----------------------------------------------------------------
-------------')
# Count the proportion of different departments, sorting the
proportions in descending order
dept_proportion =
store_depts["department"].value_counts(normalize=True).sort_values(asc
ending=False)
print(dept_proportion)

# Calc total weekly sales
sales_all = sales["weekly_sales"].sum()
# Subset for type A stores, calc total weekly sales
sales_A = sales[sales["type"] == "A"]["weekly_sales"].sum()
# Subset for type B stores, calc total weekly sales
sales_B = sales[sales["type"] == "B"]["weekly_sales"].sum()
# Subset for type C stores, calc total weekly sales
sales_C = sales[sales["type"] == "C"]["weekly_sales"].sum()
# Get proportion for each type
```

```
sales_propn_by_type = [sales_A, sales_B, sales_C] / sales_all
print(sales_propn_by_type)

# For each store type, aggregate weekly_sales to calculate min, max,
mean, and median
sales_stats = sales.groupby('type')['weekly_sales'].agg([min, max,
np.mean, np.median])
# Print the aggregated sales statistics
print("Sales Statistics by Store Type:")
print(sales_stats)

# For each store type, aggregate unemployment and fuel_price_usd_per_l
to calculate min, max, mean, and median
unemp_fuel_stats = sales.groupby('type')[['unemployment',
'fuel_price_usd_per_l']].agg([min, max, np.mean, np.median])
# Print the aggregated unemployment and fuel price statistics
print("\nUnemployment and Fuel Price Statistics by Store Type:")
print(unemp_fuel_stats)

temperatures = pd.read_csv("Class03_Data Manipulation Data
Sets/temperatures.csv", index_col=0)

# Print the head of the temperatures DataFrame
print(temperatures)

# Set the index of temperatures to city
temperatures_ind = temperatures.set_index('city')

# Look at temperatures_ind
print(temperatures_ind)

# Reset the temperatures_ind index, keeping its contents
print(temperatures_ind.reset_index())

# Reset the temperatures_ind index, dropping its contents
print(temperatures_ind.reset_index(drop = True))

# Make a list of cities to subset on
cities = ["Moscow", "Saint Petersburg"]

# Subset temperatures using square brackets
print(temperatures[temperatures['city'].isin(cities)])

# Subset temperatures_ind using .loc[]
print(temperatures_ind.loc[cities])

# Import matplotlib.pyplot with alias plt
import matplotlib.pyplot as plt

# Load the data
avocados = pd.read_pickle("Class03_Data Manipulation Data
```

```
Sets/avoplotto.pkl")

# Look at the first few rows of the data
print("First few rows of the dataset:")
print(avocados.head())

# Get the total number of avocados sold for each size
nb_sold_by_size = avocados.groupby('size')['nb_sold'].sum()

# Create a bar plot of the number of avocados sold by size
print("\nNumber of avocados sold by size:")
print(nb_sold_by_size)
nb_sold_by_size.plot(kind='bar', title="Number of Avocados Sold by
Size")

# Display the plot
plt.xlabel("Size")
plt.ylabel("Number of Avocados Sold")
plt.tight_layout()
plt.show()

# Get the total number of avocados sold on each date
nb_sold_by_date = avocados.groupby('date')['nb_sold'].sum()
# Create a line plot of the number of avocados sold by date
nb_sold_by_date.plot(kind='line', rot = 45)
# Show the plot
plt.show()

# Scatter plot of avg_price vs. nb_sold with title
avocados.plot(x='nb_sold', y='avg_price', kind = 'scatter', title =
'Number of avocados sold vs. average price')
# Show the plot
plt.show()

# Histogram of conventional avg_price
avocados[avocados['type'] == 'conventional']['avg_price'].plot(kind =
'hist')
# Histogram of organic avg_price
avocados[avocados['type'] == 'organic']['avg_price'].plot(kind =
'hist')
# Add a legend
plt.legend(['conventional','organic'])
# Show the plot
plt.show()

# Histogram of conventional avg_price
avocados[avocados['type'] == 'conventional']['avg_price'].plot(kind =
'hist')
# Histogram of organic avg_price
avocados[avocados['type'] == 'organic']['avg_price'].plot(kind =
'hist', alpha = 0.6)
```

```
# Add a legend
plt.legend(['conventional','organic'])
# Show the plot
plt.show()

avocados_2016 = pd.read_csv("Class03_Data Manipulation Data
Sets/avocados_2016.csv", index_col = 0)

# Check individual values for missing values
print(avocados_2016.isna())
# Check each column for missing values
print(avocados_2016.isna().any())
# Bar plot of missing values by variable
avocados_2016.isna().sum().plot(kind = 'bar')
# Show plot
plt.show()
# Remove rows with missing values
avocados_complete = avocados_2016.dropna()
# Check if any columns contain missing values
print(avocados_complete.isna().any())

gdp_data = pd.read_csv("Class03_Data Manipulation Data
Sets/WorldBank_GDP.csv", index_col = 0)

# Which country's GDP is growing during the Year 2010 and Year 2018?

# Filter data for the years 2010 and 2018
gdp_2010_2018 = gdp_data[gdp_data['Year'].isin([2010, 2018])]

# Pivot the data to make countries as rows and years as columns
gdp_pivot = gdp_2010_2018.pivot(columns='Year', values='GDP')

# Drop rows with missing values for either year
gdp_pivot = gdp_pivot.dropna()

# Calculate GDP growth from 2010 to 2018
gdp_pivot['Growth'] = gdp_pivot[2018] - gdp_pivot[2010]

# Filter countries with positive growth
growing_countries = gdp_pivot[gdp_pivot['Growth'] > 0]

# Plot the GDP growth for these countries
growing_countries['Growth'].sort_values(ascending=False).plot(kind='ba
r')
plt.title('Countries with Positive GDP Growth (2010 to 2018)')
plt.ylabel('GDP Growth (US$)')
plt.xlabel('Country')
plt.show()

avg_temp_c = temperatures.groupby('country')
['avg_temp_c'].mean().reset_index()
```

```
print(avg_temp_c)
print(avg_temp_c.max())
temp_country2030 = avg_temp_c[(avg_temp_c['avg_temp_c'] >= 20) &
(avg_temp_c['avg_temp_c'] <= 30)][['country', 'avg_temp_c']]
print(temp_country2030)
print(temp_country2030.count())

temp_thailand = temperatures[temperatures['country'] == 'Thailand']
temp_thailand

temp_thailand_20052010 =
temp_thailand[temp_thailand['date'].between('2005-01-01', '2010-01-
01')]
display(temp_thailand_20052010)

# Average Temperature during that Period
avg_temp_thailand_200520010 =
temp_thailand_20052010['avg_temp_c'].mean()
print("The avg. temp of Thailand during 2005-2010 is ",
avg_temp_thailand_200520010, "Celsius.")
```