Exercise 1: Data Structures in Pandas

import pandas as pd

# Load the housing data from the CSV file into a DataFrame

housing\_data = pd.read\_csv('housing\_data.csv')

# Create a new DataFrame with houses having more than 4 bedrooms and price > $1 million

filtered\_data = housing\_data[(housing\_data['bedrooms'] > 4) & (housing\_data['price'] > 1000000)]

# Print the first 10 rows of the filtered DataFrame

print(filtered\_data.head(10))

Exercise 2: Series Operations

# Calculate and print the 90th percentile of house prices

percentile\_90 = housing\_data['price'].quantile(0.9)

print("90th Percentile of House Prices:", percentile\_90)

# Calculate and print the variance in square footage of living space among houses with 3 bathrooms

variance\_sqft\_living = housing\_data[housing\_data['bathrooms'] == 3]['sqft\_living'].var()

print("Variance in Sqft Living (3 Bathrooms):", variance\_sqft\_living)

# Count houses with waterfront views and more than 4 bedrooms

count\_waterfront\_4bedrooms = len(housing\_data[(housing\_data['waterfront'] == 1) & (housing\_data['bedrooms'] > 4)])

print("Count of Houses with Waterfront and > 4 Bedrooms:", count\_waterfront\_4bedrooms)

Exercise 3: Exploratory Data Analysis (EDA) with Pandas

# Calculate advanced statistics for each unique zipcode

zipcodes = housing\_data['zipcode'].unique()

eda\_results = []

for zipcode in zipcodes:

zipcode\_data = housing\_data[housing\_data['zipcode'] == zipcode]

mean\_price = zipcode\_data['price'].mean()

median\_sqft\_living = zipcode\_data['sqft\_living'].median()

std\_price = zipcode\_data['price'].std()

total\_houses = len(zipcode\_data)

eda\_results.append({

'Zipcode': zipcode,

'Mean Price': mean\_price,

'Median Sqft Living': median\_sqft\_living,

'Std Price': std\_price,

'Total Houses': total\_houses

})

# Create a DataFrame from the results and display it

eda\_df = pd.DataFrame(eda\_results)

print(eda\_df)

Exercise 4: Indexing with Pandas

# Select the top 5 most expensive houses and specific columns

top\_expensive\_houses = housing\_data.nlargest(5, 'price')[['price', 'bedrooms', 'bathrooms', 'zipcode']]

print("Top 5 Most Expensive Houses:")

print(top\_expensive\_houses)

# Create a DataFrame with houses meeting specific criteria

filtered\_houses = housing\_data[(housing\_data['bathrooms'] >= 3) & (housing\_data['view'] >= 3) & (housing\_data['condition'] == 5)]

print("Houses with >= 3 Bathrooms, View >= 3, and Condition 5:")

print(filtered\_houses.head(10))

Exercise 5: Aggregating Data and GroupBy

# Group the data by 'grade' and calculate advanced statistics

grouped\_data = housing\_data.groupby('grade').agg({

'price': 'mean',

'sqft\_living': 'median',

'bedrooms': 'max',

'bathrooms': 'min'

}).reset\_index()

# Rename the columns for clarity

grouped\_data.rename(columns={

'price': 'Mean Price',

'sqft\_living': 'Median Sqft Living',

'bedrooms': 'Max Bedrooms',

'bathrooms': 'Min Bathrooms'

}, inplace=True)

# Sort the DataFrame by 'grade'

grouped\_data.sort\_values(by='grade', inplace=True)

print("Advanced Statistics Grouped by Grade:")

print(grouped\_data)

Exercise 6: Custom Aggregations

# Define a custom aggregation function

def avg\_price\_per\_sqft(df):

return (df['price'] / df['sqft\_living']).mean()

# Group the data by 'zipcode' and apply the custom aggregation

custom\_agg\_result = housing\_data.groupby('zipcode').apply(avg\_price\_per\_sqft).reset\_index()

custom\_agg\_result.columns = ['Zipcode', 'Avg Price per Sqft']

# Sort the results by average price per sqft

custom\_agg\_result.sort\_values(by='Avg Price per Sqft', ascending=False, inplace=True)

print("Average Price per Sqft by Zipcode:")

print(custom\_agg\_result)