```
# DataFrames Exercises
# https://codeshare.io/w9DXLB
# SaiPrabath Chowdary S



Exercise 1: Creating DataFrame from Scratch
1. Create a DataFrame with the following columns: `"Product"`, `"Category"`,
`"Price"`, and `"Quantity"`. Use the following data:
   - Product: `['Laptop', 'Mouse', 'Monitor', 'Keyboard', 'Phone']`
   - Category: `['Electronics', 'Accessories', 'Electronics', 'Accessories',
'Electronics']`
   - Price: `[80000, 1500, 20000, 3000, 40000]`
   - Quantity: `[10, 100, 50, 75, 30]`
2. Print the DataFrame.

import pandas as pd
data = {
    "Product": ['Laptop', 'Mouse', 'Monitor', 'Keyboard', 'Phone'],
    "Category": ['Electronics', 'Accessories', 'Electronics', 'Accessories',
'Electronics'],
    "Price": [80000, 1500, 20000, 3000, 40000],
    "Quantity": [10, 100, 50, 75, 30]
}

df = pd.DataFrame(data)
print(df)



Exercise 2: Basic DataFrame Operations **
1. Display the first 3 rows of the DataFrame.
2. Display the column names and index of the DataFrame.
3. Display a summary of statistics (mean, min, max, etc.) for the numeric columns
in the DataFrame.



print(df.head(3))

print(df.columns)
print(df.index)

print(df.describe())
```

```
Exercise 3: Selecting Data
1. Select and display the `"Product"` and `"Price"` columns.
2. Select rows where the `"Category"` is `"Electronics"` and print them.
print(df[['Product', 'Price']])

electronics_df = df[df["Category"] == "Electronics"]
print(electronics_df)

Exercise 4: Filtering Data
1. Filter the DataFrame to display only the products with a price greater than
`10,000`.
2. Filter the DataFrame to show only products that belong to the `"Accessories"`
category and have a quantity greater than `50`.

filtered_df = df[df["Price"] > 10000]
print(filtered_df)

filtered_df = df[(df["Category"] == "Accessories") & (df["Quantity"] > 50)]
print(filtered_df)

Exercise 5: Adding and Removing Columns
1. Add a new column `"Total Value"` which is calculated by multiplying `"Price"`
and `"Quantity"`.
2. Drop the `"Category"` column from the DataFrame and print the updated
DataFrame.

df["Total Value"] = df["Price"] * df["Quantity"]
print(df)

df_dropCol = df.drop(columns=["Category"])
print(df_dropCol)

Exercise 6: Sorting Data
1. Sort the DataFrame by `"Price"` in descending order.
2. Sort the DataFrame by `"Quantity"` in ascending order, then by `"Price"` in
descending order (multi-level sorting).

df_sorted_price = df.sort_values(by="Price", ascending=False)
print(df_sorted_price)

df_sorted_multi = df.sort_values(by=["Quantity", "Price"], ascending=[True,
False])
print(df_sorted_multi)
```

```
Exercise 7: Grouping Data
1. Group the DataFrame by `"Category"` and calculate the total quantity for each
category.
2. Group by `"Category"` and calculate the average price for each category.

grouped_df = df.groupby("Category")["Quantity"].sum()
print(grouped_df)

grouped_df = df.groupby("Category")["Price"].mean()
print(grouped_df)


Exercise 8: Handling Missing Data
1. Introduce some missing values in the `"Price"` column by assigning `None` to
two rows.
2. Fill the missing values with the mean price of the available products.
3. Drop any rows where the `"Quantity"` is less than `50`.
'''
df_copy = df
df_copy.loc[2, "Price"] = None
df_copy.loc[4, "Price"] = None

df_copy["Price"] = df_copy["Price"].fillna(df_copy["Price"].mean()) #
df_copy.update(df_copy["Price"].fillna(df_copy["Price"].mean()))
print(df_copy)

df_copy = df_copy[df_copy["Quantity"] >= 50]
print(df_copy)


Exercise 9: Apply Custom Functions
1. Apply a custom function to the `"Price"` column that increases all prices by
5%.
2. Create a new column `"Discounted Price"` that reduces the original price by
10%.


def IncPrice(price):
    return price*1.05

df["Price"] = df["Price"].apply(IncPrice)
print(df)
```

```
df["Discounted Price"] = df["Price"] * 0.9
print(df)


Exercise 10: Merging DataFrames
1. Create another DataFrame with columns `"Product"` and `"Supplier"`, and merge
it with the original DataFrame based on the `"Product"` column.

df2 = pd.DataFrame(
    {
        "Product": ["Laptop", "Mouse", "Monitor"],
        "Supplier": ["Supplier A", "Supplier B", "Supplier C"]
    }
)

merged_df = pd.merge(df, df2, on="Product", how='inner')
print(merged_df)



Exercise 11: Pivot Tables
1. Create a pivot table that shows the total quantity of products for each
category and product combination.

pivot_table = df.pivot_table(index=["Category", "Product"], values="Quantity",
aggfunc="sum")
print(pivot_table)



Exercise 12: Concatenating DataFrames
1. Create two separate DataFrames for two different stores with the same columns
(`"Product"`, `"Price"`, `"Quantity"`).
2. Concatenate these DataFrames to create a combined inventory list.

store1_df = pd.DataFrame({"Product": ["Laptop", "Mouse"], "Price": [80000, 1500],
"Quantity": [10, 100]})
store2_df = pd.DataFrame({"Product": ["Monitor", "Keyboard"], "Price": [20000,
3000], "Quantity": [50, 75]})
combined_df = pd.concat([store1_df, store2_df], ignore_index=True)
print(combined_df)


Exercise 13: Working with Dates
```

```
1. Create a DataFrame with a `"Date"` column that contains the last 5 days
starting from today.
2. Add a column `"Sales"` with random values for each day.
3. Find the total sales for all days combined.

import datetime
today = datetime.date.today()
date_range = pd.date_range(start=today, periods=5)

date_df = pd.DataFrame({"Date": date_range})

date_df["Sales"] = [100, 200, 150, 80, 120]
print(date_df)

total_sales = date_df["Sales"].sum()
print(total_sales)


Exercise 14: Reshaping Data with Melt
1. Create a DataFrame with columns `"Product"`, `"Region"`, `"Q1_Sales"`,
`"Q2_Sales"`.
2. Use `pd.melt()` to reshape the DataFrame so that it has columns `"Product"`,
`"Region"`, `"Quarter"`, and `"Sales"`.


Exercise 15: Reading and Writing Data
1. Read the data from a CSV file named `products.csv` into a DataFrame.
2. After performing some operations (e.g., adding a new column or modifying
values), write the DataFrame back to a new CSV file named `updated_products.csv`.

df = pd.read_csv("products.csv")
df["TotalSales"] = df["Price"] * df["Quantity"]
df.to_csv("updated_products.csv", index=False)


Exercise 16: Renaming Columns**
1. Given a DataFrame with columns `"Prod"`, `"Cat"`, `"Price"`, `"Qty"`, rename
the columns to `"Product"`, `"Category"`, `"Price"`, and `"Quantity"`.
2. Print the renamed DataFrame.

df_rename = pd.DataFrame(
    {
    "Prod": ['Laptop', 'Mouse'],
```

```python
    "Cat": ['Electronics', 'Accessories'],
    "Price": [80000, 1500],
    "Qty": [10, 100]
    }
)

df_rename.columns = ["Product", "Category", "Price", "Quantity"]
print(df_rename)
```