import pandas as pd  
 import numpy as np  
 from datetime import datetime  
  
 Exercise 1:

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)  
 2. print(df)

Exercise 2:

1. print(df.head(3))  
2. print(df.info())  
3. print(df.describe())  
  
 Exercise 3:

1. print(df[["Product","Price"]])  
 2. print(df[df["Category"]=="Electronics"])  
  
 Exercise 4:

1. print(df[df["Price"] > 10000])  
 2. print(df[(df["Category"]=="Accessories") & (df["Quantity"] > 50)])  
  
 Exercise 5:

1. df["Total Value"] = df["Price"] \* df["Quantity"]  
 print(df)  
  
 2. df\_dropped\_cat =  df.drop(columns = ['Category'])  
 print(df\_dropped\_cat)

Exercise 6:

1.print(df.sort\_values(by="Price",ascending=False))  
 2. print(df.sort\_values(by=["Quantity","Price"],ascending=[True,False]))

Exercise 7:

1. df\_Cat\_group = df.groupby("Category")["Quantity"].sum()  
 print(df\_Cat\_group)  
  
 2. df\_catAvg\_group = df.groupby("Category")["Price"].mean()  
 print(df\_catAvg\_group)  
  
 Exercise 8:  
 1. df.loc[0:1,"Price"] = None  
 print(df)  
 2. mean\_price = df["Price"].mean()  
 df["Price"] = df["Price"].fillna(mean\_price)  
 print(df)  
 3. df\_quant = df[df["Quantity"] < 50].index  
 df\_drop\_quantity = df.drop(df\_quant)  
 print(df\_drop\_quantity)  
  
 Exercise 9:

1. df["Price"] = df["Price"].apply(lambda x: x\*1.05)  
 print(df)  
 2. df["Discounted Price"] = df["Price"].apply(lambda x: x\*0.90)  
 print(df)  
  
 Exercise 10:

1. df\_2 = pd.DataFrame({  
        "Product" : ["Laptop","Mouse",'Monitor', 'Keyboard', 'Phone'],  
        "Supplier" : ["HP","Logitech","Dell","Samsung","Apple"]  
 })  
 df\_2\_merged = pd.merge(df,df\_2,on="Product",how="left")  
 print(df\_2\_merged)  
  
 Exercise 11:  
 1. df\_pivot = df.pivot\_table(values="Quantity",index="Category",columns="Product",aggfunc="sum")  
 print(df\_pivot)  
  
 Exercise 12:

 1. df\_1 = pd.DataFrame({  
        "Product": ["Laptop","Keyboard","Mouse"],  
        "Price": [50000,2500,1500],  
        "Quantity":[50,30,40]  
 })  
  
 df2 = pd.DataFrame({  
        "Product": ["Phone","Camera","GPU"],  
        "Price": [25000,30000,40000],  
        "Quantity":[10,30,20]  
 })  
  
 2. inventry\_df = pd.concat([df\_1,df2],ignore\_index=True)  
 print(inventry\_df)  
  
 Exercise 13:  
 1. today = datetime.today()  
 five\_dates = pd.date\_range(end=today,periods=5)  
  
 df\_dates = pd.DataFrame({  
        "Date" : five\_dates  
 })  
  
 2. df\_dates["Sales"] = np.random.randint(100,500,size = 5)  
  
 3. print(df["Sales"].sum())  
  
 Exercise 14:  
 1. df\_melt = pd.DataFrame({  
        "Product": ["Laptop","Camera","Phone"],  
        "Region": ["TN","Kerala","Andhra"],  
        "Q1\_Sales": [100,200,150],  
        "Q2\_Sales": [500,600,700]  
 })  
  
 2. df\_melted = pd.melt(df\_melt, id\_vars=["Product","Region"], var\_name="Quarter", value\_name="Sales")  
 print(df\_melted)  
  
 Exercise 15:

1. df\_products = pd.read\_csv("products.csv")  
 print(df\_products)  
  
 2. df\_products["Category"] = ["Electronic","Accessories","Electronic","Electronic"]  
 print(df\_products)  
 df\_products["Price"] = df\_products["Price"] - 1000  
 print(df\_products)  
  
 df\_products.to\_csv("updated\_products.csv")  
  
 Exercise 16:

1. df\_to\_rename = pd.DataFrame({  
        "Prod" : ["Laptop","Mobile"],  
        "Cat" : ["Electronics","Electronics"],  
        "Price": [50000,25000],  
        "Qty": [25,10]  
 })  
  
 df\_rename = df\_to\_rename.rename(columns={"Prod":"Product","Cat":"Category","Qty":"Quantity"})  
  
 2.print(df\_rename)  
  
 Exercise 17:

1. multi\_index = pd.MultiIndex.from\_tuples(  
 [("Store\_A","Laptop"),  
         ("Store\_A","Mouse"),  
         ("Store\_A","Monitor"),  
         ("Store\_B", "Laptop"),  
         ("Store\_B", "Mouse"),  
         ("Store\_B", "Monitor"),  
         ],  
        names=["Store","Product"]  
 )  
  
 data\_multi = {  
        "Price" : [50000,1500,20000,40000,3000,15000],  
        "Quantity": [15,20,25,15,10,30]  
 }  
  
 df\_multi\_index = pd.DataFrame(data\_multi,index=multi\_index)  
  
 2. print(df\_multi\_index)  
  
 Exercise 18:

 1.today\_date = datetime.today()  
 dates = pd.date\_range(end=today\_date,periods=30)  
 sales = np.random.randint(100,1000,size=30)  
  
 df\_sales = pd.DataFrame({  
        "Date" : dates,  
        "Sales": sales  
 })  
 print(df\_sales)  
  
 2. df\_sales.set\_index("Date",inplace=True  
 weekly\_sales = df\_sales.resample("W").sum()  
 print(weekly\_sales)  
  
 Exercise 19:

1. df\_duplicate = pd.DataFrame({  
        "Product" : ["Laptop","Mouse","Phone","Laptop","Mouse"],  
        "Price" : [50000,2500,30000,50000,2500]  
 })  
  
 duplicates = df\_duplicate.duplicated()  
 print(df\_duplicate[duplicates])  
  
 df\_cleaned = df\_duplicate.drop\_duplicates()  
  
 2. print(df\_cleaned)  
  
 Exercise 20:

1. df\_correlation = pd.DataFrame({  
        'Height': [160, 175, 168, 180, 155],  
        'Weight': [60, 70, 65, 85, 50],  
        'Age': [25, 32, 28, 40, 22],  
        'Income': [50000, 60000, 58000, 80000, 45000]  
 })  
  
 2. correlation\_matrix\_df = df\_correlation.corr()  
  
 3. print(correlation\_matrix\_df)  
  
 Exercise 21:

1. print(df\_sales)  
  
 2. df\_sales["Cumulative Sales"] = df\_sales["Sales"].cumsum()

3. df\_sales["Rolling Average"] = df\_sales["Sales"].rolling(window=7).mean()  
 print(df\_sales)  
  
 Exercise 22:

 1.df\_string = pd.DataFrame({  
        "Names" : ["John Doe", "Jane Smith", "Sam Brown"],  
 })  
  
 2.df\_string[["First Name", "Last Name"]] = df\_string["Names"].str.split(' ', expand=True)  
  
 3. df\_string["First Name"] = df\_string["First Name"].str.upper()  
 print(df\_string)  
  
 Exercise 23:

1. df\_np\_where = pd.DataFrame({  
        "Employee" : ["Sai", "Subash"],  
        "Age" : [45,30],  
        "Department" : ["IT", "Finance"]  
 })  
  
 2. df\_np\_where["Status"] = np.where(df\_np\_where["Age"] >= 40, "Senior", "Junior")  
 print(df\_np\_where)

Exercise 24:

1. df\_slice = pd.DataFrame({  
        'Product': ['Laptop', 'Mouse', 'Monitor', 'Keyboard', 'Phone', 'Tablet', 'Printer', 'Webcam', 'Speaker', 'Headphones', 'Charger', 'Case', 'Dock'],  
        'Category': ['Electronics', 'Accessories', 'Electronics', 'Accessories', 'Electronics', 'Electronics', 'Accessories', 'Electronics', 'Accessories', 'Electronics', 'Accessories', 'Accessories', 'Electronics'],  
        'Sales': [80000, 1500, 20000, 3000, 40000, 12000, 2500, 70000, 5000, 18000, 1500, 2000, 3000],  
        'Profit': [20000, 500, 7000, 800, 10000, 3000, 400, 15000, 800, 6000, 400, 600, 700]  
 })  
 print(df\_slice.head(10))  
 print(df\_slice[df\_slice["Category"]=="Electronics"])  
 print(df\_slice[df\_slice["Sales"]>50000][["Sales","Profit"]])

Exercise 25:

1. df\_store\_a = pd.DataFrame({  
        "Employee" : ["Sai","Subash"],  
        "Age" : [30,45],  
        "Salary" : [50000,60000]  
 })  
  
 df\_store\_b = pd.DataFrame({  
        "Employee" : ["Chandra","Akash"],  
        "Age" : [31,46],  
        "Salary" : [55000,66000]  
 })  
 2. df\_verti = pd.concat([df\_store\_a,df\_store\_b])  
 print(df\_verti)  
  
 3. df\_horizontal1 = pd.DataFrame({  
        "Employee" : ["Sai", "Subash"],  
        "Department" : ["HR", "IT"]  
 })  
  
 df\_horizontal2 = pd.DataFrame({  
        "Employee" : ["Sai", "Subash"],  
        "Salary" : [50000,60000]  
 })  
  
 df\_horizontal = pd.merge(df\_horizontal1,df\_horizontal2, on="Employee")  
 print(df\_horizontal)  
  
 Exercise 26:

 1. df\_explode = pd.DataFrame({  
        'Product': ['Laptop', 'Smartphone', 'Tablet'],  
        'Features': [['Touchscreen', '8GB RAM', '256GB SSD'],  
                   ['5G', '64GB Storage', '12MP Camera'],  
                   ['10.5 inch Screen', '4GB RAM']]  
 })  
  
 2. df\_exploded = df\_explode.explode("Features")  
 print(df\_exploded)  
  
 Exercise 27:

1. df\_1["Price"] = df\_1["Price"].map(lambda x : x\*1.10)  
 print(df\_1)  
  
 2. df\_1\_format = df\_1.applymap(lambda x: f"{x:.2f}" if isinstance(x, (int,float)) else x)  
 print(df\_1\_format)  
  
 Exercise 28:

1. df\_city = pd.DataFrame({  
        "City" : ["Chennai","Bangalore", "Hyderabad", "Chennai", "Bangalore", "Mumbai", "Hyderabad"],  
        "Product" : ["Laptop","Mouse","Keyboard","Phone","Tablet","Monitor","CPU"],  
        "Sales" : [80000, 1500, 20000, 3000, 40000, 12000, 10000],  
        "Profit" : [20000, 500, 7000, 800, 10000, 3000, 500]  
 })  
  
 2. def profit\_margin(data):  
        data["Profit Margin"] = data["Profit"] / data["Sales"]  
        return data  
 df\_profit\_margin = df\_city.groupby("City").apply(profit\_margin)  
 print(df\_profit\_margin)

Exercise 29:

1. df\_csv = pd.read\_csv("data\_merge.csv")  
 df\_json = pd.read\_json("data\_merge.json")  
  
 data\_dict = pd.DataFrame({  
        'ID': [1, 2, 3],  
        'Location': ['Chennai', 'Bangalore', 'Mumbai']  
 })  
  
 2. df\_merged\_multiple = pd.merge(df\_csv,df\_json, on="ID")  
 df\_merged\_multiple = pd.merge(df\_merged\_multiple, data\_dict, on="ID")  
  
 print(df\_merged\_multiple)  
  
 Exercise 30:

1.num\_rows = 1000000  
 transaction\_id = np.arange(1, num\_rows+1)  
 customers = np.random.choice(["Sai","Subash","Chandra","Akash","Rex","Alen","Induja"], num\_rows)  
 products = np.random.choice(["Laptop","Mouse","Monitor","Keyboard","Phone"], num\_rows)  
 amounts = np.random.uniform(10,1000,num\_rows)  
 start\_date = pd.Timestamp('2023-01-01')  
 end\_date = pd.Timestamp('2024-01-01')  
 date\_range = pd.date\_range(start=start\_date, end=end\_date)  
 random\_dates = np.random.choice(date\_range, size=num\_rows, replace=True)  
  
 df\_large = pd.DataFrame({  
        "Transaction ID" : transaction\_id,  
        "Customer" : customers,  
        "Product" : products,  
        "Amount" : amounts,  
        "Date" : random\_dates  
 })  
  
 print(df\_large.head())  
  
 2. def analyze\_chunk(chunk):  
        total\_sales = chunk["Amount"].sum()  
        return total\_sales  
  
 chunk\_size = 100000  
 num\_chunks = num\_rows // chunk\_size  
  
 chunks = np.array\_split(df\_large,num\_chunks)  
  
 results = [analyze\_chunk(chunk) for chunk in chunks]  
  
 total\_sales\_large = sum(results)  
  
 print(total\_sales\_large)