|  |  |  |
| --- | --- | --- |
| Pandas DataFrame  Practice | Date : 10/01/25 | |
| **Completed Pandas DataFrame Assignment. Assignment 1: DataFrame Creation and Basic Operations Assignment 2: Data Cleaning Assignment 3: Aggregation and Grouping Assignment 4: Data Merging and Concatenation Assignment 5: File Handling [Optional] Assignment 6: Sorting and Filtering Assignment 7: Advanced Challenge** | |  |

## Assignment 1: DataFrame Creation and Basic Operations

1. Create a DataFrame for a small retail store with columns `Product`, `Category`, `Price`, and `Quantity`.   
2. Perform the following:  
 - Add a new column `Total` that calculates the total value for each product (`Price \* Quantity`).  
 - Select products in the `Electronics` category with a price greater than 500.  
 - Drop a product by its index.

import pandas as pd

import numpy as np

retail\_store = pd.DataFrame({

'Product': ['wheat powder', 'salt','Headphone','toothpaste', 'Coke'],

'Category': ['Food', 'Food','Electronics', 'self care', 'beverage'],

'Price': [120, 50, 850, 109, 90],

'Quantity' : [5, 2, 1, 1, 3]

})

print(retail\_store)

print('=================================================================')

# Adding new column 'Total' that calculates the total

retail\_store['Total'] = retail\_store['Price'] \* retail\_store['Quantity']

print(retail\_store)

print('=================================================================')

# 'Electronics' category with a price greater than 500

Electronics\_category = retail\_store[(retail\_store['Category'] == 'Electronics') & (retail\_store['Price'] > 500)]

print(Electronics\_category)

print('=================================================================')

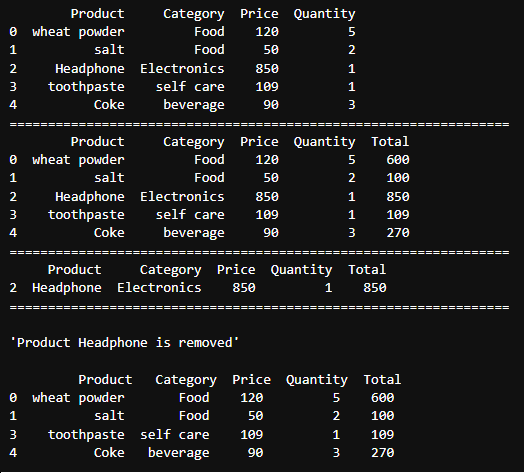
retail\_store = retail\_store.drop(index=2)

# Dropping a product by its index

print("\n'Product Headphone is removed'\n" ,retail\_store)



**Output:**



## Assignment 2: Data Cleaning

1. Create a DataFrame with missing values in columns `Name`, `Age`, `City`, and `Salary`.   
2. Perform the following:  
 - Fill missing values in the `Salary` column with the column's mean.  
 - Replace missing `City` values with "Unknown".  
 - Drop rows where both `Name` and `Age` are missing.

Employee\_details = pd.DataFrame({

'Name': ['Rakshith', np.nan,'Anjan',np.nan],

'Age': [np.nan, np.nan, 23, 48],

'City': ['Shivamogga', np.nan, 'Kalburgi', 'Udupi'],

'Salary' : [58000, 22000, np.nan, 18000]

})

print('\nDataFrame with missing values:\n\n' ,Employee\_details)

print('==========================================================')

# Filling missing values in the Salary column with the column's mean

Employee\_details['Salary'] = Employee\_details['Salary'].fillna(Employee\_details['Salary'].mean())

print('\nDataFrame after filling missing values in Salary column with mean:\n\n', Employee\_details)

print('==========================================================')

# Replacing missing City values with "Unknown"

Employee\_details['City'] = Employee\_details['City'].replace(to\_replace=np.nan, value="Unknown")

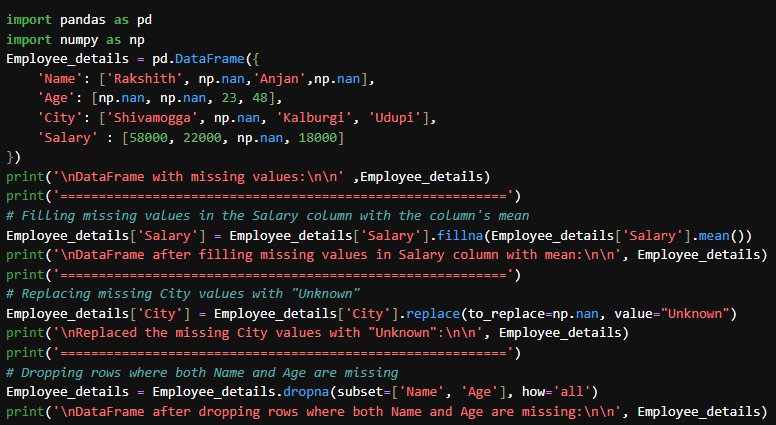
print('\nReplaced the missing City values with "Unknown":\n\n', Employee\_details)

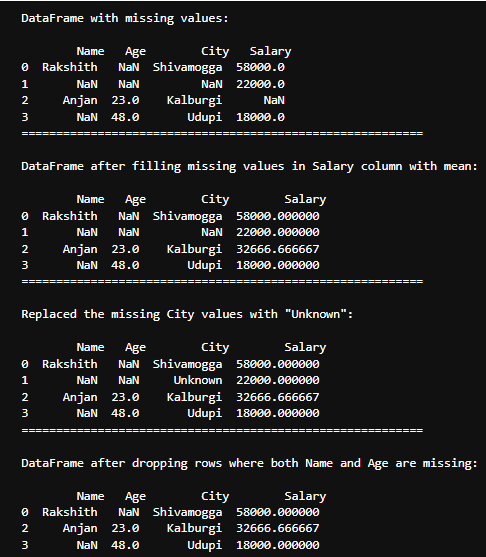
print('==========================================================')

# Dropping rows where both Name and Age are missing

Employee\_details = Employee\_details.dropna(subset=['Name', 'Age'], how='all')

print('\nDataFrame after dropping rows where both Name and Age are missing:\n\n', Employee\_details)





## Assignment 3: Aggregation and Grouping

1. Use a dataset containing `Department`, `Employee`, `Salary`, and `Years\_of\_Experience`.   
2. Perform the following:  
 - Group by `Department` and calculate the total and average salary.  
 - Identify the maximum `Years\_of\_Experience` in each department.  
 - Sort the grouped results by average salary in descending order.

professors\_info = pd.DataFrame({

'Department': ['Electronics & Communication', 'Computer Science', 'Electricals', 'Mechanical', 'Information Science', 'Data Science'],

'Employee': ['Rakshith', 'Chetan', 'Hafeez', 'Siddhu', 'Sinchana', 'Anamika'],

'Salary': [55000, 72000, 68000, 62000, 75000, 71000],

'Years\_of\_Experience': [5, 8, 7, 6, 9, 10]

})

print(professors\_info)

print('===============================================================================')

# Group by Department and calculate total and average salary and maximum Years\_of\_Experience

grouped\_salary = professors\_info.groupby('Department').agg({'Salary': ['sum', 'mean'], 'Years\_of\_Experience': 'max'})

grouped\_salary.columns = ['Total Salary', 'Average Salary', 'Max Years\_of\_Experience']

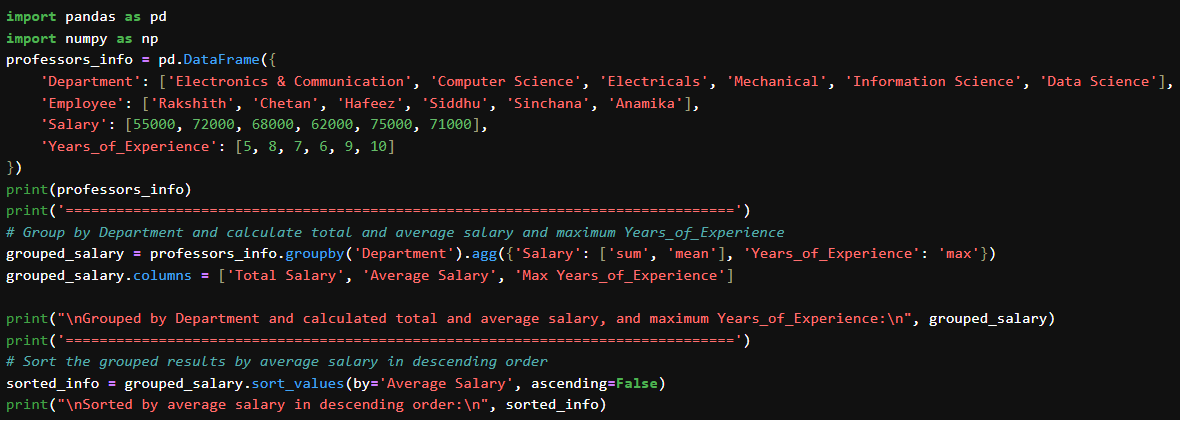
print("\nGrouped by Department and calculated total and average salary, and maximum Years\_of\_Experience:\n", grouped\_salary)

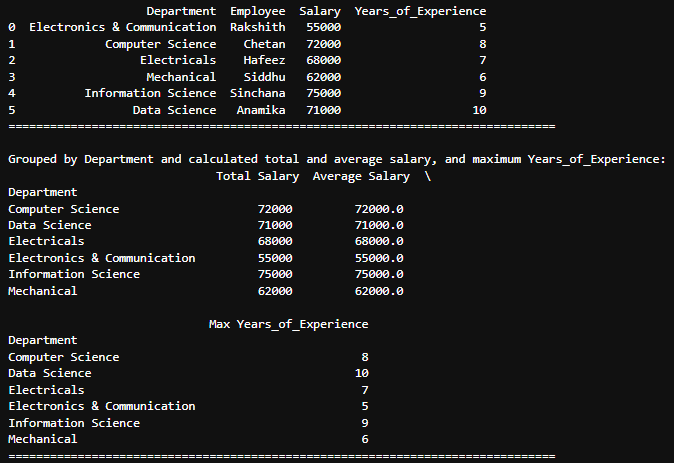
print('===============================================================================')

# Sort the grouped results by average salary in descending order

sorted\_info = grouped\_salary.sort\_values(by='Average Salary', ascending=False)

print("\nSorted by average salary in descending order:\n", sorted\_info)





## Assignment 4: Data Merging and Concatenation

1. Create two DataFrames:  
 - `DF1`: `Customer\_ID`, `Name`, `Email`.   
 - `DF2`: `Customer\_ID`, `Purchase\_Amount`, `Date`.   
2. Merge them using `Customer\_ID`.   
3. Find:  
 - Customers with total purchases above 10,000.  
 - Add a column indicating if the purchase date is in the current year.

**Code**:

DF1 = pd.DataFrame({ 'Customer\_ID': [1, 2, 3],

'Name': ['Rakshith', 'Deekshith', 'Anjan'],

'Email': ['rakshith@gmail.com', 'deekshith@gmail.com', 'anjan@gmail.com']

})

DF2 = pd.DataFrame({ 'Customer\_ID': [1, 2, 2, 3],

'Purchase\_Amount': [10250, 19150, 8100, 300],

'Date': ['2025-01-01', '2024-01-02', '2025-01-03', '2025-01-04']

})

print("\n-Customer information :\n", DF1)

print("\n-Purchase information:\n", DF2)

print('--------------------------------------------------------------------')

# Merging the DataFrames on Customer\_ID

merged\_DF = pd.merge(DF1, DF2, on='Customer\_ID')

print("\n-Customer information and Purchase information:\n\n",merged\_DF )

print('--------------------------------------------------------------------')

# Customers with total purchases above 10,000

print("\n-Customers with total purchases above 10,000:\n\n", merged\_DF[merged\_DF["Purchase\_Amount"]>10000])

print('--------------------------------------------------------------------')

# imported datetime module

from datetime import datetime

# Convert the 'Date' column to datetime format

DF2['Date'] = pd.to\_datetime(DF2['Date'])

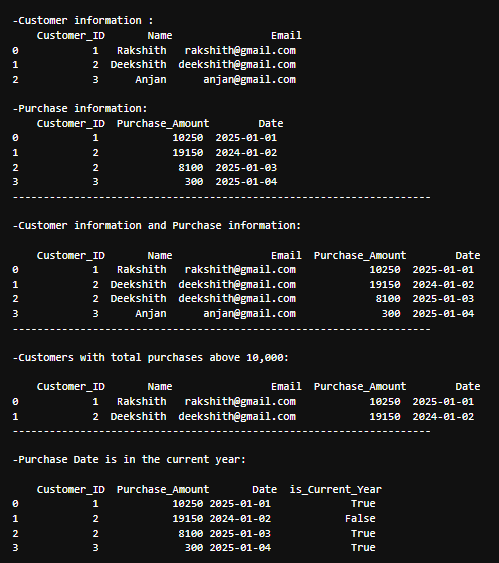
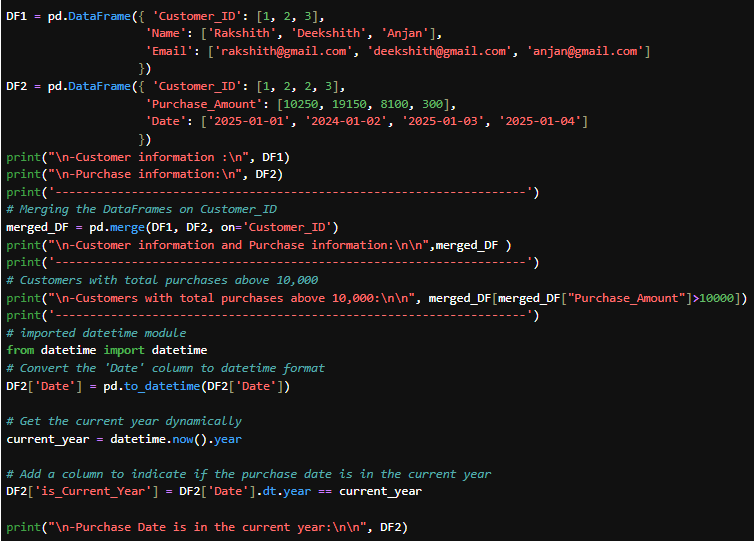
# Get the current year dynamically

current\_year = datetime.now().year

# Add a column to indicate if the purchase date is in the current year

DF2['is\_Current\_Year'] = DF2['Date'].dt.year == current\_year

print("\n-Purchase Date is in the current year:\n\n", DF2)



## Assignment 5: File Handling [Optional]

1. Load a CSV file of your choice into a DataFrame. Perform the following:  
 - Display the first 10 rows and identify any missing values.  
 - Write the cleaned DataFrame to a new CSV file after handling missing data

import pandas as pd

import numpy as np

# Load the CSV file into a DataFrame

df = pd.read\_csv('customers-100.csv')

# Display the first 10 rows

print("First 10 rows of the DataFrame:\n", df.head(10))

print('-------------------------------------------------')

# Identify missing values

missing\_values = df.isnull().sum()

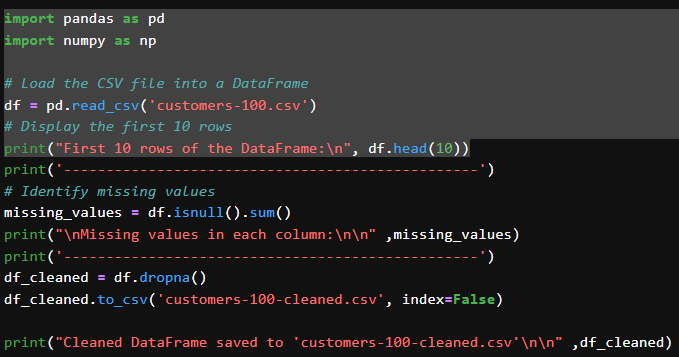
print("\nMissing values in each column:\n\n" ,missing\_values)

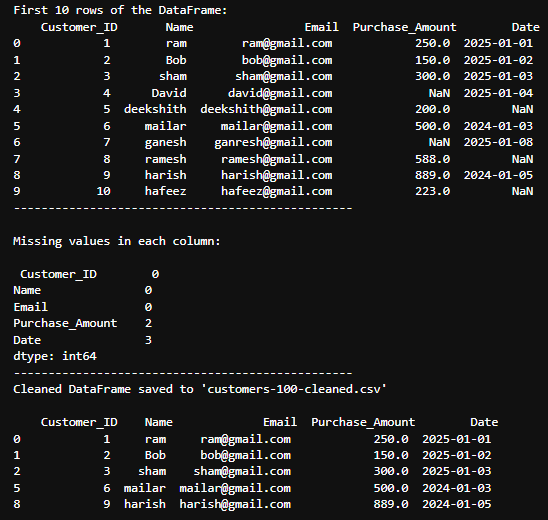
print('-------------------------------------------------')

df\_cleaned = df.dropna()

df\_cleaned.to\_csv('customers-100-cleaned.csv', index=False)

print("Cleaned DataFrame saved to 'customers-100-cleaned.csv'\n\n" ,df\_cleaned)





## Assignment 6: Sorting and Filtering

1. Create a DataFrame representing an e-commerce order list with columns `Order\_ID`, `Customer`, `Product`, `Order\_Date`, and `Amount`.   
2. Perform the following:  
 - Sort orders by `Amount` in descending order.  
 - Filter orders placed in the last 30 days.  
 - Identify customers with total order amounts exceeding 20,000

order\_list = pd.DataFrame({

'Order\_ID': [101, 102, 103, 104, 105],

'Customer': ['Rakshith', 'Deekshith', 'Amith', 'Rahul', 'Gani'],

'Product': ['Book', 'Smartphone', 'T-Shirt', 'Whey-protien', 'Camera'],

'Order\_Date': ['2025-01-15', '2024-01-18', '2024-01-20', '2025-01-22', '2025-01-25'],

'Amount': [800, 58000, 600, 2500, 90000]

})

print("\n-Order List :\n\n", order\_list)

print('-----------------------------------------------------------')

# Sorting the DataFrame by Amount in descending order

sorted\_order\_list = order\_list.sort\_values(by ='Amount', ascending=False)

# Display the sorted DataFrame

print("\n-Sorted Order List by Amount in Descending Order:\n\n", sorted\_order\_list)

print('-----------------------------------------------------------')

#importing module

from datetime import datetime, timedelta

# Convert 'Order\_Date' to datetime format

order\_list['Order\_Date'] = pd.to\_datetime(order\_list['Order\_Date'])

# Get the current date

current\_date = datetime.now()

# Calculate the date 30 days ago

thirty\_days\_ago = current\_date - timedelta(days=30)

# Filter orders placed in the last 30 days

recent\_orders = order\_list[order\_list['Order\_Date'] >= thirty\_days\_ago]

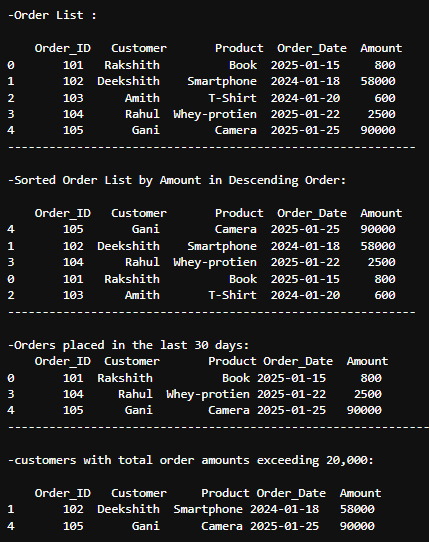
# Display the filtered DataFrame

print("\n-Orders placed in the last 30 days:\n", recent\_orders)

print('-------------------------------------------------------------------')

print("\n-customers with total order amounts exceeding 20,000:\n\n", order\_list[order\_list["Amount"]>20000])





## Assignment 7: Advanced Challenge

Using a dataset of your choice (e.g., public datasets from Kaggle or CSV files), perform the following:  
1. Clean the data by handling missing values and duplicates.   
2. Perform exploratory analysis:  
 - Identify trends (e.g., yearly sales growth or decline).  
 - Group and aggregate data based on relevant features.  
3. Save the analysis results to a new CSV file.

import pandas as pd

import numpy as np

# Load the CSV file into a DataFrame

df = pd.read\_csv('sales\_trend\_data.csv')

print(df)

missing\_values = df.isnull().sum()

print(f"Missing values in each column:\n{missing\_values}")

# Fill missing values

df['Units\_Sold'].fillna(df['Units\_Sold'].mean(), inplace=True)

df['Revenue'].fillna(df['Revenue'].mean(), inplace=True)

df['Category'].fillna(df['Category'].mode()[0], inplace=True)

df['Region'].fillna(df['Region'].mode()[0], inplace=True)

# Check for duplicate rows

duplicates = df.duplicated().sum()

print(f"Number of duplicate rows: {duplicates}")

# Removing duplicate rows

df.drop\_duplicates(inplace=True)

# Verifing that the data is cleaned

print(f"\nData after cleaning:\n{df.head()}")

print(f"\nMissing values after cleaning:\n{df.isnull().sum()}")

print(f"Number of duplicate rows after cleaning: {df.duplicated().sum()}")

# Save the cleaned data to a new CSV file

cleaned\_file\_path = "cleaned\_sales\_trend\_data.csv"

df.to\_csv(cleaned\_file\_path, index=False)

