Questions related to numpy

1. Creating NumPy Arrays:

- Create a 1D NumPy array with 10 elements, containing the numbers from 1 to 10.
- Create a 2D NumPy array of shape (3, 4) filled with ones.
- Create a 3x3 identity matrix using NumPy.

2. Array Operations:

- Add 5 to each element in a 1D NumPy array: `[1, 2, 3, 4, 5]`.
- Multiply each element in a 1D NumPy array: `[2, 4, 6, 8, 10]` by 2.
- Subtract 3 from each element in a NumPy array: `[6, 8, 10, 12, 14]`.

3. Indexing and Slicing:

- Given a 1D array `arr = np.array([10, 20, 30, 40, 50])`, access the third element.
- Slice a 1D array to get the first 3 elements.
- Given a 2D array, extract the element from the second row and third column.

4. Array Statistics:

- Find the minimum and maximum value in a NumPy array: `arr = np.array([2, 4, 6, 8, 10])`.
- Calculate the sum and mean of the array: `arr = np.array([3, 5, 7, 9])`.

5. Reshaping and Flattening:

- Reshape a 1D array `arr = np.array([1, 2, 3, 4, 5, 6])` into a 2D array of shape (2, 3).
- Flatten a 2D array `arr = np.array([[1, 2, 3], [4, 5, 6]])` into a 1D array.

6. Array Stacking:

- Stack two 1D arrays `[1, 2, 3]` and `[4, 5, 6]` vertically and horizontally.

7. Mathematical Operations:

- Perform element-wise addition and subtraction on two arrays: `arr1 = np.array([2, 4, 6])` and `arr2 = np.array([1, 3, 5])`.
- Perform element-wise multiplication and division on two arrays: `arr1 = np.array([2, 4, 6])` and `arr2 = np.array([1, 2, 3])`.

8. Linear Algebra:

- Calculate the dot product of two 1D arrays: `arr1 = np.array([1, 2])` and `arr2 = np.array([3, 4])`.
- Multiply two matrices using `np.dot()`. Given matrices `A = np.array([[1, 2], [3, 4]])` and `B = np.array([[5, 6], [7, 8]])`, compute their dot product.

9. Random Numbers:

- Generate an array of 10 random numbers between 0 and 1 using `np.random.rand()`.
- Create a random integer array with values between 1 and 100, and of size 5 using `np.random.randint()`.

10. Array Comparisons:

- Compare two arrays element-wise: `arr1 = np.array([1, 2, 3])` and `arr2 = np.array([3, 2, 1])`. Find which elements in `arr1` are greater than `arr2`.

11. Aggregating Functions:

- Use `np.sum()`, `np.mean()`, and `np.std()` to compute the sum, mean, and standard deviation of the array `arr = np.array([10, 20, 30, 40, 50])`.
- Find the index of the maximum value in a NumPy array `arr = np.array([5, 2, 8, 1, 6])` using `np.argmax()`.

12. Broadcasting:

- Add a 1D array `[1, 2, 3]` to a 2D array `[[10, 20, 30], [40, 50, 60]]` using broadcasting.

Questions related to pandas

1. Basic Data Exploration:

- Use `.info()` to find out what type of data each column contains.
- Use `.head()` to show the first 5 rows of the dataset.
- Use `.describe()` to get a summary of numerical columns in the dataset.

2. Handling Missing Data:

- Fill any missing values in the `Price` column with the mean price of all products.
- Fill any missing values in the `Stock_Quantity` column with the median stock quantity.
- Drop any rows that have missing values in the `Product_Name` column.

3. Data Sorting:

- Sort the dataset by the `Price` column in descending order.
- Sort the dataset by the `Stock_Quantity` column in ascending order.

4. Filtering Data:

- Filter and display all products that have a `Price` greater than 1000.
- Filter and display all products with a `Rating` of 4 or higher.

5. Adding New Columns:

- Add a new column called `Total_Value` that represents the product of `Price` and `Stock_Quantity`.
- Add a new column `Discounted_Price`, where products with a `Rating` of 4 or higher get a 10% discount, and others get a 5% discount.

6. Sorting by Multiple Columns:

- Sort the products first by `Category` in ascending order, and then by `Price` in descending order.

7. Working with Categorical Data:

- Count how many products are in each category by using `.value_counts()` on the `Category` column.

8. Creating a New DataFrame:

- Create a new DataFrame containing only the `Product_ID` and `Price` columns from the original dataset.