Data Analysis Assignment: Transactional Data

This assignment is designed to test your proficiency with Python's core data science libraries: Pandas, NumPy, Matplotlib, and Seaborn. You will be working with a dataset containing information about various transactions. The goal is to clean, analyze, and visualize the data to uncover meaningful patterns.

The Dataset

You should use the `frozen_confections_data.csv` file provided with this assignment. It contains the following columns:

- `TransactionID`
- `SuccessfulOrder`
- `FlavorCategory`
- `CustomerGender`
- `CustomerAge`
- `ToppingsCount`
- `ScoopsCount`
- `TotalCost`
- `PaymentMethod`
- `CustomerName`

Part 1: Data Loading and Initial Inspection (Pandas)

Objective: Load the dataset and perform basic checks to understand its structure and content.

1. Load the `frozen_confections_data.csv` file into a Pandas DataFrame.
    - **Step:** Use `pd.read_csv()`.
2. Display the first 5 rows of the DataFrame.
    - **Step:** Use the `.head()` method.
3. Get a concise summary of the DataFrame, including the data types of each column and the number of non-null values.
    - **Step:** Use the `.info()` method.
4. Display the columns and their respective data types.
    - **Step:** Use the `.dtypes` attribute.
5. Check the dimensions (number of rows and columns) of the DataFrame.
    - **Step:** Use the `.shape` attribute.

Part 2: Data Cleaning and Manipulation (Pandas)

Objective: Clean the data and prepare it for analysis.

1. Calculate the number of missing values in each column.
    - **Step:** Use `.isnull()` followed by `.sum()`.
2. Check if there are any duplicate rows in the DataFrame.
    - **Step:** Use `.duplicated()` and `.sum()`.
3. Rename the `SuccessfulOrder` column to `OrderSuccess`.
    - **Step:** Use the `.rename()` method with the `columns` parameter.
4. Create a new column named `ItemsCount` which is the sum of `ToppingsCount` and `ScoopsCount`.
    - **Step:** Add the two columns together.
5. Calculate the average `TotalCost` for each `PaymentMethod`.
    - **Step:** Use the `.groupby()` method on `PaymentMethod` and then apply `.mean()` to `TotalCost`.

Part 3: Basic Exploratory Data Analysis (EDA) and NumPy

Objective: Use descriptive statistics and NumPy to gain initial insights.

1. Calculate the mean, median, and standard deviation of `CustomerAge`.
    - **Step:** Use `.mean()`, `.median()`, and `.std()` on the `CustomerAge` column.
2. Find the most common `PaymentMethod` and `FlavorCategory`.
    - **Step:** Use the `.mode()` method.
3. Count the number of successful and unsuccessful orders.
    - **Step:** Use the `.value_counts()` method on the `OrderSuccess` column.
4. Use NumPy to calculate the percentage of successful orders.
    - **Step:** Calculate the sum of successful orders and divide by the total number of records.
5. Find the maximum `TotalCost` and the minimum `TotalCost` from the dataset.
    - **Step:** Use `.max()` and `.min()` on the `TotalCost` column.

Part 4: Data Visualization (Matplotlib & Seaborn)

Objective: Create informative plots to visually represent key relationships in the data.

1. Create a histogram of the `CustomerAge` distribution.
    - **Step:** Use `plt.hist()` from Matplotlib.
2. Visualize the count of orders by `PaymentMethod` using a bar plot.
    - **Step:** Use `sns.countplot()` from Seaborn.
3. Create a countplot to show the number of successful orders (`OrderSuccess`) for each `FlavorCategory`.
    - **Step:** Use `sns.countplot()` and set the `hue` parameter to `OrderSuccess`.
4. Create a box plot to show the distribution of `TotalCost` across different `FlavorCategory`s.
    - **Step:** Use `sns.boxplot()`.

5. Create a violin plot to show the distribution of `CustomerAge` for successful versus unsuccessful orders.
    - **Step:** Use `sns.violinplot()`.
6. Generate a scatter plot showing the relationship between `ToppingsCount` and `TotalCost`.
    - **Step:** Use `plt.scatter()` or `sns.scatterplot()`.
7. Create a bar chart showing the average `TotalCost` per `PaymentMethod`.
    - **Step:** Group the data by `PaymentMethod`, calculate the mean `TotalCost`, and then use `sns.barplot()`.
8. Visualize the correlation matrix of the numerical columns using a heatmap.
    - **Step:** Calculate the correlation matrix using `.corr()` and then use `sns.heatmap()`.
9. Create a pie chart to show the proportion of each `FlavorCategory`.
    - **Step:** Use `plt.pie()` after getting the value counts for `FlavorCategory`.
10. Use a pair plot to visualize the relationships between all numerical features.
    - **Step:** Use `sns.pairplot()` on a subset of the DataFrame containing only numerical columns.