Module 5 | Concepts and Code Components

1. Checking for Missing Values:
Identifying null or missing values in a DataFrame. This process involves identifying and locating missing or null values within a DataFrame. In this assignment, identifying missing values is crucial to understanding the extent of incomplete data and deciding on appropriate handling strategies.

```
df.isnull()  # Returns DataFrame of booleans indicating missing values
df.isna()    # Equivalent to isnull()
```

Ex.

```
# Check for missing values
missing_values = df.isnull()
print(missing_values)
```

2. Summing Missing Values by Column
Calculate the total number of missing values for each column. Summing missing values by column provides a count of how many missing values each column contains. For this assignment, this practice helps prioritize which columns need attention and what kind of imputation or removal strategies to use.

```
df.isnull().sum()  # Returns Series with count of missing values per column
```

3. Dropping Missing Values
Remove rows or columns with missing values. Dropping missing values involves removing rows or columns that contain null values, depending on the analysis requirements. For this assignment, dropping rows or columns with missing values ensures cleaner data for subsequent analysis but may result in data loss, so it needs to be done judiciously.

```
df.dropna()        # Drops rows with any missing values
df.dropna(axis=1)   # Drops columns with any missing values
```

4. Filling Missing Values
Replace missing values with specified values. Filling missing values, or imputation, involves replacing null values with specific values like the mean, median, or a constant. For this assignment, filling missing values helps retain as much data as possible while making it suitable for analysis.

```
df.fillna(value)              # Fill missing values with 'value'
df.fillna(df.mean(), inplace=True)  # Fill missing values with column mean
```

5. Grouping Data
Group data by one or more columns. Grouping data involves splitting the data into subsets based on the values of one or more columns. For this assignment, grouping data by categories like 'Pclass' or 'Sex' allows for aggregated analysis, providing insights into different segments of the dataset.

```
df.groupby('column')  # Group by a single column
df.groupby(['col1', 'col2'])  # Group by multiple columns
```

6. Aggregating Data
Perform aggregate operations like mean, median, etc., on grouped data. Aggregating data means performing calculations like mean, median, or sum on grouped data to get summary statistics. For this assignment, aggregating data after grouping helps derive meaningful statistics, such as the mean fare by passenger class or the median age by gender and class.

```
df.groupby('column').mean()  # Calculate mean of each group
df.groupby('column').median()  # Calculate median of each group
```

7. Chaining Operations
Combine multiple operations in a single statement using dot notation. Chaining operations involves combining multiple methods in a single line of code using dot notation, improving code readability and efficiency. For this assignment, chaining simplifies complex data manipulation tasks, making it easier to apply multiple transformations or aggregations in a streamlined manner.

```
Ex. df.groupby('column').mean().reset_index()
```

8. Handling Missing Data Using Machine Learning Concepts
Use machine learning models to predict and fill missing values. This involves using machine learning models to predict and fill missing values based on patterns in the data. Though not explicitly required in the assignment, understanding this concept can be useful for advanced handling of missing data where simple imputation might not suffice.

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
df['column'] = imputer.fit_transform(df[['column']])
```

9. Joining Dataframes
Joining dataframes involves merging two or more DataFrames based on a common key, allowing for comprehensive analysis by combining related data. Though not explicitly required for this assignment, oining DataFrames enables the integration of

additional information, like bonuses by passenger class, enhancing the dataset with more context for analysis.

```
# Perform inner join
df_inner = df_cleaned.merge(df_additional, on='Pclass', how='inner')

# Perform left join
df_left = df_cleaned.merge(df_additional, on='Pclass', how='left')
```

10. Data Exploration and Visualization
Data exploration involves summarizing and visualizing data to uncover patterns and insights. Visualization tools like Matplotlib help create plots that make data interpretation easier. For this assignment, exploring and visualizing data helps summarize key statistics and trends, such as average fares by class, aiding in the analysis and presentation of findings.

```
import matplotlib.pyplot as plt
# Calculate mean and median fares
mean_fare = df['Fare'].mean()
median_fare = df['Fare'].median()

# Bar plot of mean fare by class
mean_fare_by_class.plot(kind='bar', title='Mean Fare by Class')
plt.xlabel('Class')
plt.ylabel('Mean Fare')
plt.show()
```