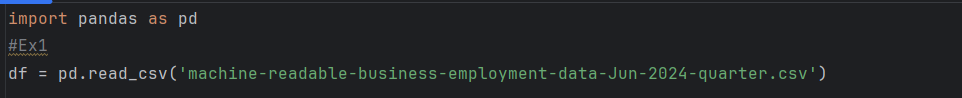
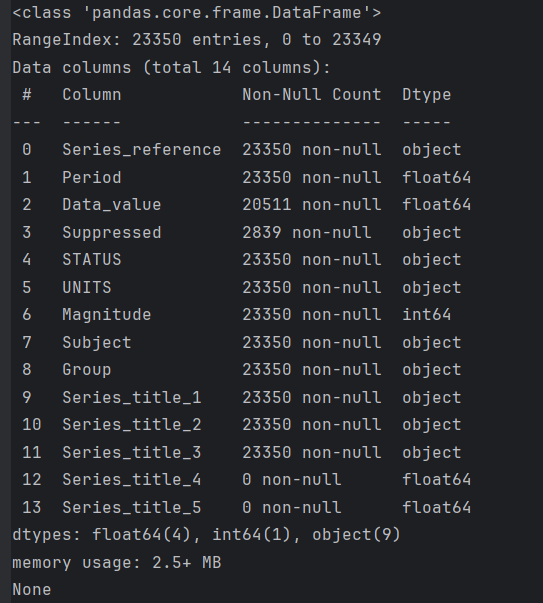
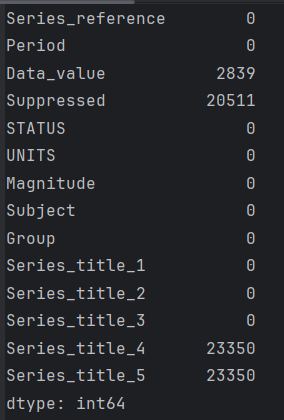
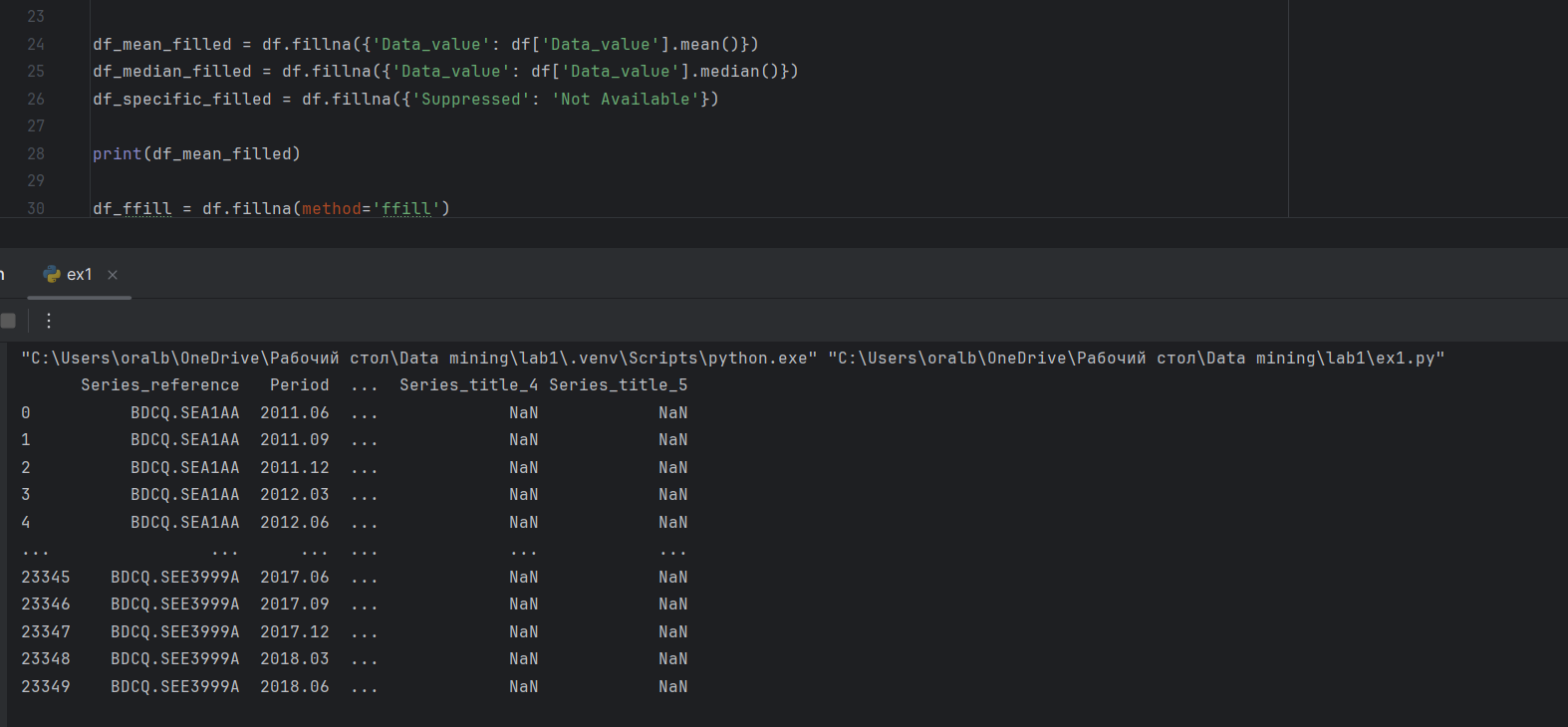
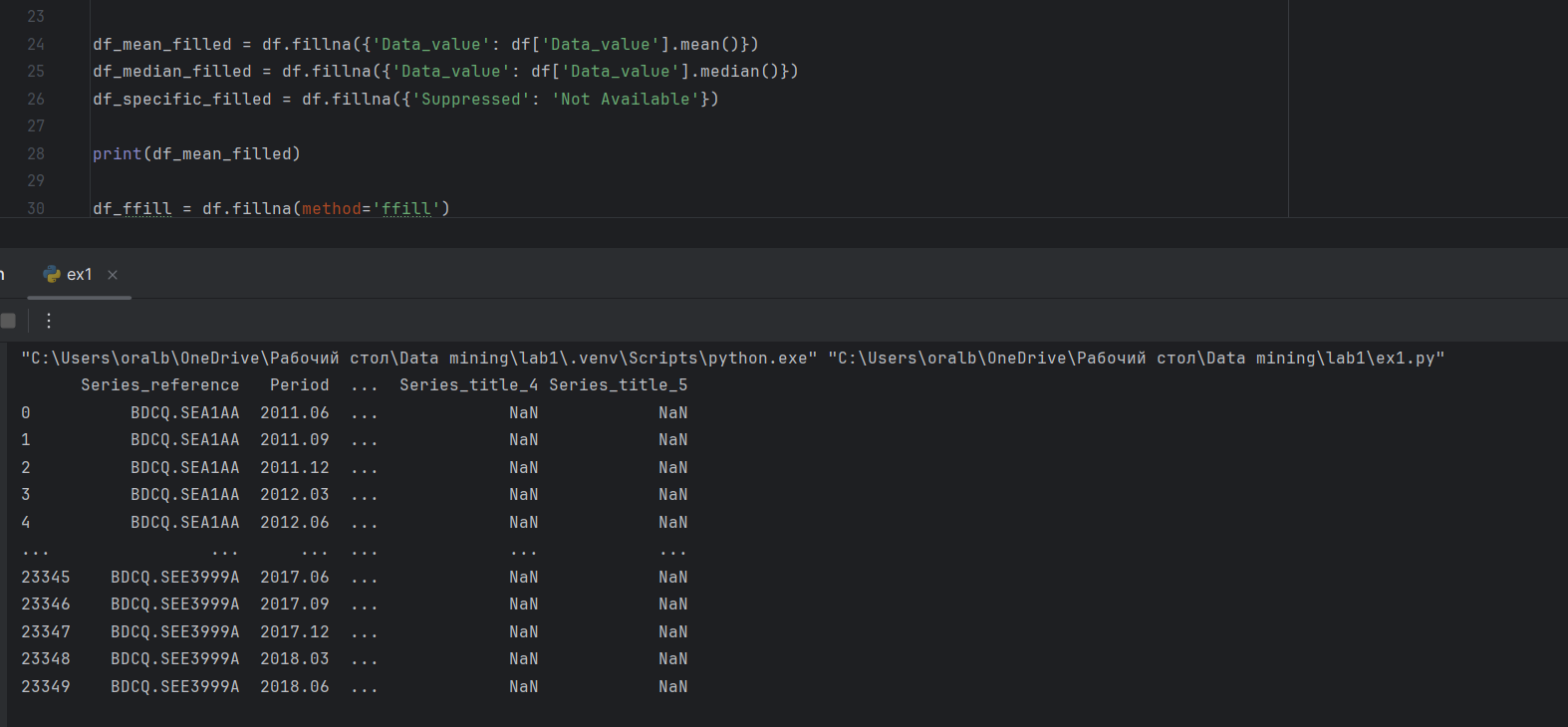
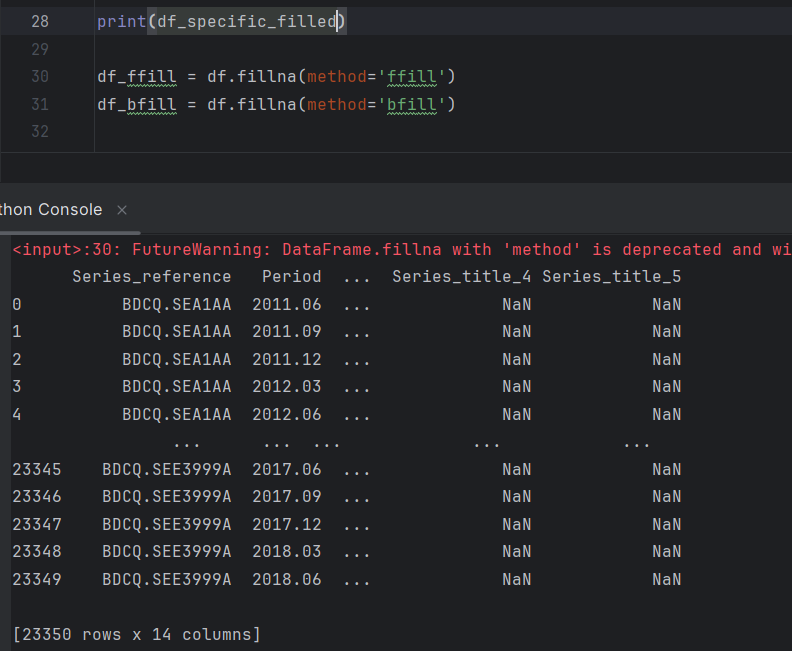
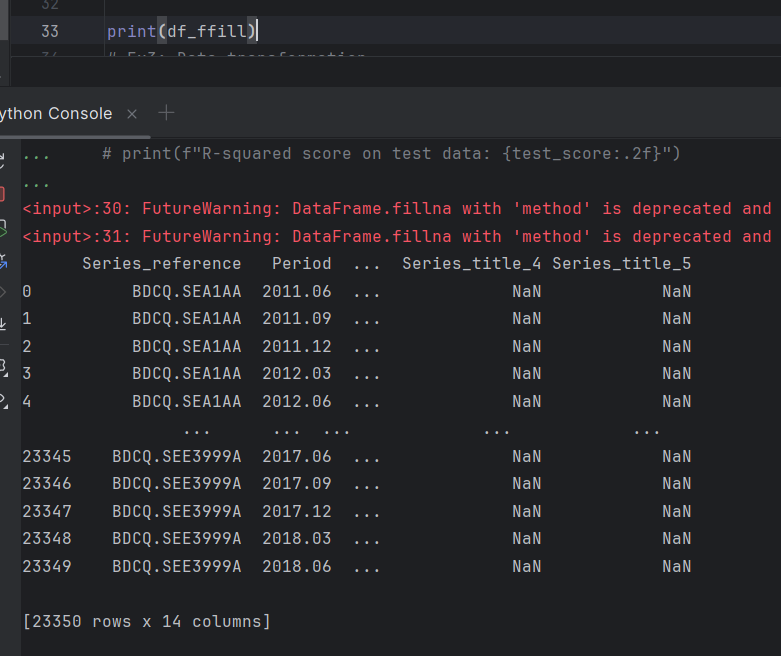
**Assignment 1, Data Mining**

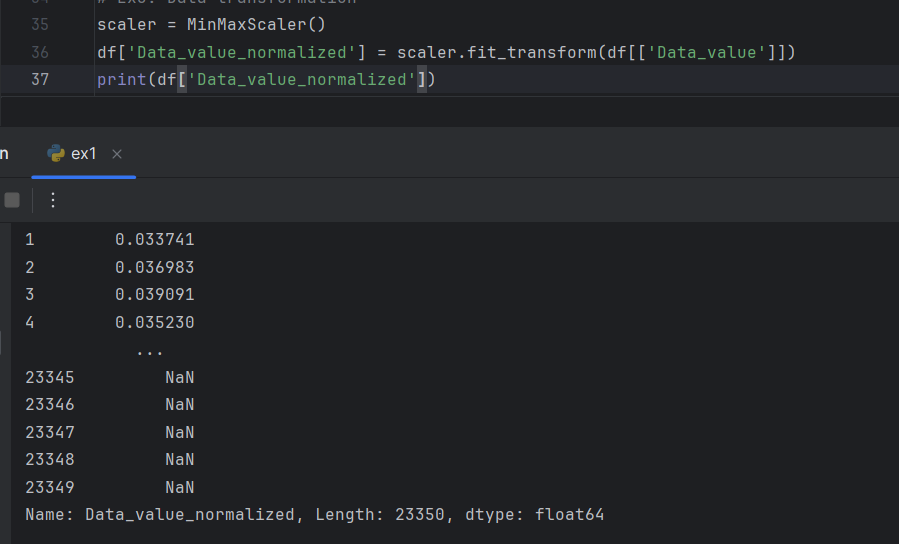
**Exercise 1: Loading Data with Pandas**

1. **Objective**: Learn how to load and inspect datasets using Pandas.
2. **Steps**:
   * Import the Pandas library and load a CSV file into a DataFrame.  
     
   * Use the head(), tail(), and info() functions to inspect the dataset.  
     
   * Check for missing values and data types of each column using isnull() and dtypes.  
     
3. **Questions**:
   * How do you load a CSV file into a Pandas DataFrame?  
     Using command df = pd.read\_csv('file\_name.csv')
   * What information does the info() function provide about the dataset?  
     it provides info about non-null, data type and memory usage of data
   * How can you identify missing values in the dataset?  
     Using df.isnull().sum()

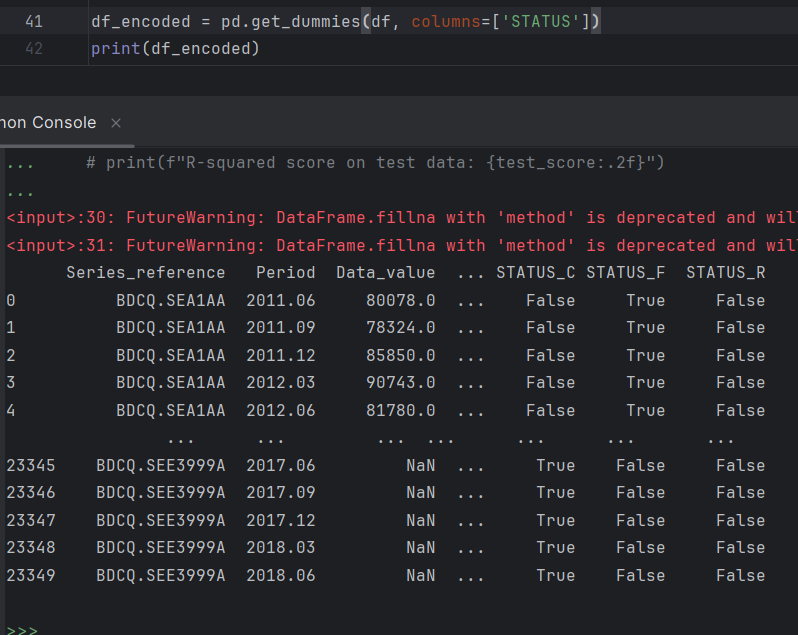
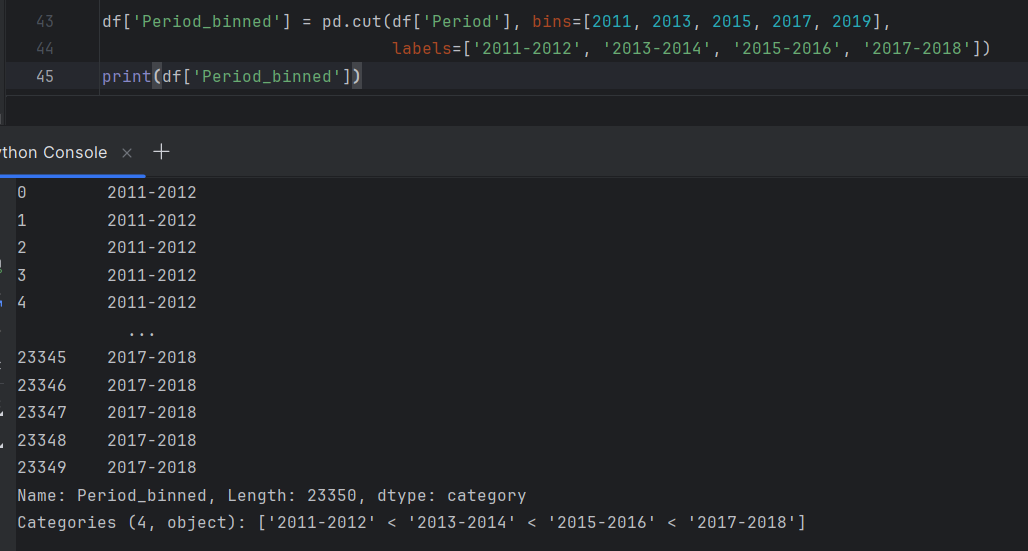
**Exercise 2: Handling Missing Data**

1. **Objective**: Practice techniques for handling missing data in a dataset.
2. **Steps**:
   * Identify missing values in the dataset using isnull().sum().
   * Use different strategies to handle missing data:
     + Remove rows with missing values using dropna().
     + Fill missing values with the mean, median, or a specific value using fillna().  
         
         
       
     + Use forward or backward filling (ffill() or bfill()) to fill missing data.  
       
   * Compare the results of each method.
3. **Questions**:
   * What strategy did you use to handle missing values, and why?  
     I used mean filling for 'Data\_value' because it minimizes data loss and keeps the overall distribution intact.
   * How did filling missing values affect the dataset?  
     It helped retain all rows and provided a complete dataset without NaN values, preserving the dataset's structure.
   * When might it be more appropriate to drop rows with missing values instead of filling them?  
     When a significant portion of the data is missing, or if the missing values are in critical columns.

**Exercise 3: Data Transformation**

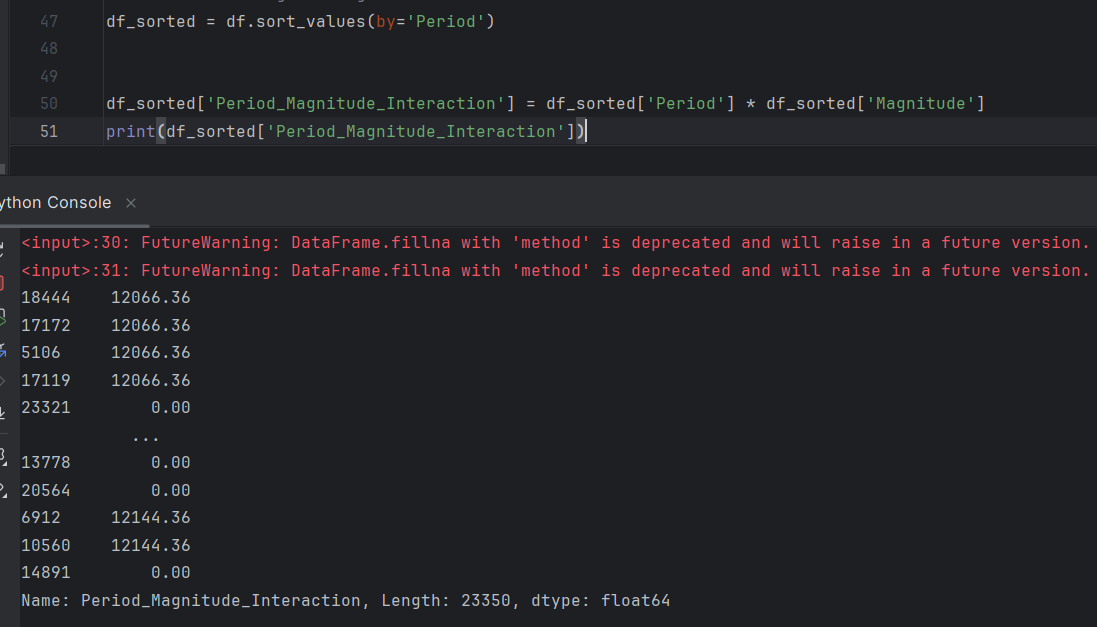
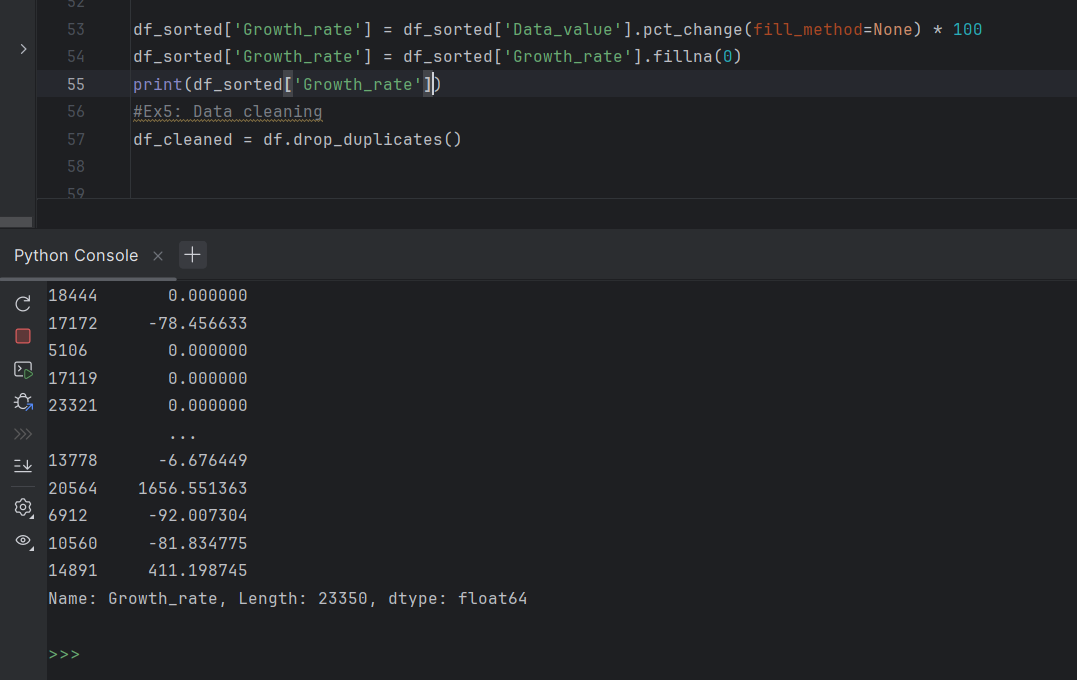
1. **Objective**: Transform data to prepare it for analysis.
2. **Steps**:
   * Normalize numerical features using Min-Max scaling or Z-score standardization with sklearn.preprocessing.  
     



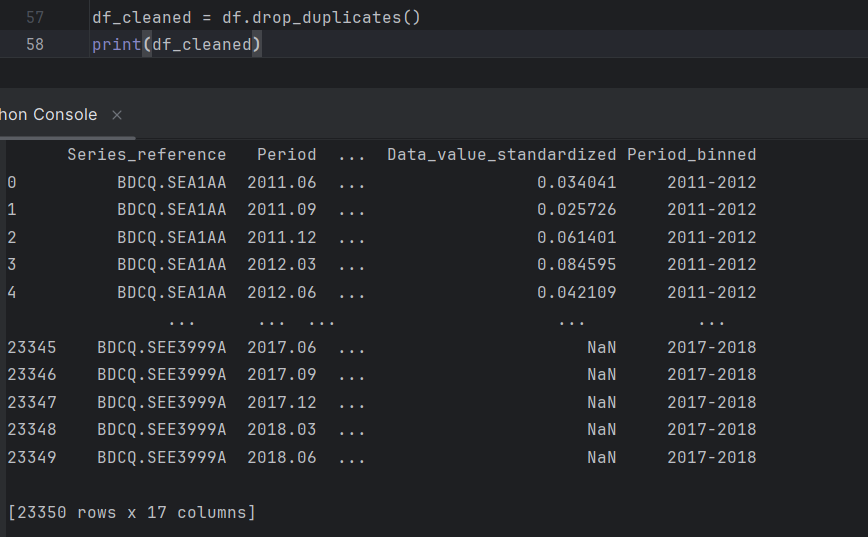
* + Encode categorical variables using one-hot encoding with pd.get\_dummies() or sklearn.preprocessing.OneHotEncoder.  
    
  + Use pd.cut() to bin continuous variables into discrete intervals.  
    

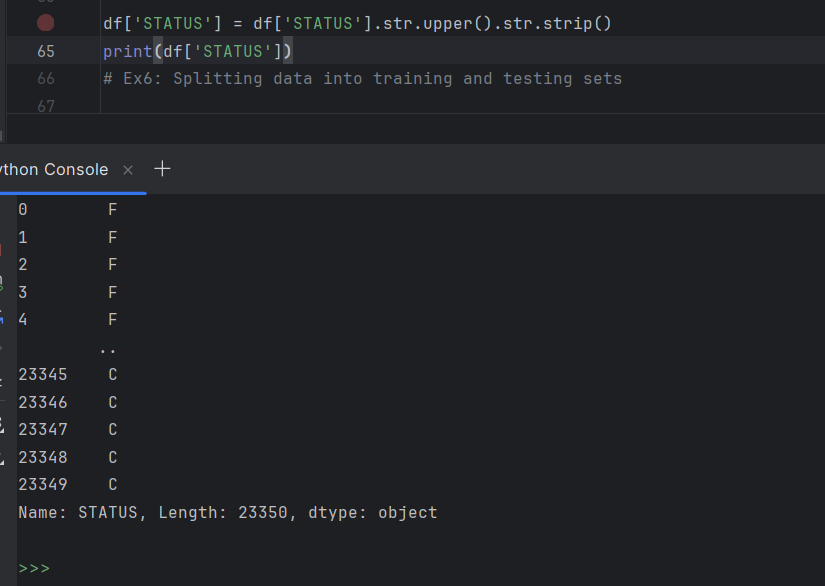
1. **Questions**:
   * What is the difference between normalization and standardization?  
     Normalization scales data between 0 and 1, while standardization scales data to have a mean of 0 and a standard deviation of 1.
   * How does one-hot encoding transform categorical variables?  
     It converts categorical variables into a series of binary columns, where each unique category is represented by a separate column.
   * Why might you want to bin continuous variables into categories?  
     To simplify the data and help models detect patterns in ranges rather than specific values.

**Exercise 4: Feature Engineering**

1. **Objective**: Create new features to improve the predictive power of a dataset.
2. **Steps**:
   * Create new features by combining or transforming existing features (e.g., adding interaction terms or polynomial features).  
     
   * Extract date-based features (e.g., year, month, day) from datetime columns using pd.to\_datetime() and dt accessor.  
     I don’t have such data in my csv file
   * Use domain knowledge to engineer features that might be useful for your specific problem.  
     
3. **Questions**:
   * What new features did you create, and why?  
     The Period\_Magnitude\_Interaction feature was created to capture the combined effect of Period and Magnitude, enhancing the model's ability to detect complex patterns and relationships between these variables and the target variable.
   * How did the new features improve the dataset?  
     They provide additional dimensions for analysis, potentially improving model performance.
   * How can date-based features be useful in a dataset?  
     They can help capture seasonality, trends over time, and time-based patterns.

**Exercise 5: Data Cleaning**

1. **Objective**: Clean data to ensure it's ready for analysis.
2. **Steps**:
   * Remove duplicate rows using drop\_duplicates().  
     
   * Detect and remove outliers using the Z-score method or the IQR method.  
     Изображение выглядит как текст, снимок экрана, программное обеспечение

     Автоматически созданное описание
   * Correct inconsistencies in categorical data (e.g., standardizing text formats or merging similar categories).  
     
3. **Questions**:
   * How did you identify and handle duplicate rows in the dataset?  
     Used df.drop\_duplicates() to remove any repeated rows.
   * What method did you use to detect and remove outliers, and why?  
     Used Z-score method to identify outliers where the absolute value is greater than 3. It's effective for normally distributed data.
   * How did you address inconsistencies in categorical data?  
     Standardized the STATUS column by converting all text to uppercase and removing extra spaces.

**Exercise 6: Splitting Data into Training and Testing Sets**

1. **Objective**: Prepare the data for model training by splitting it into training and testing sets.
2. **Steps**:
   * Use sklearn.model\_selection.train\_test\_split() to split the dataset into training and testing sets.
   * Ensure that the target variable is correctly separated from the features.
   * Explore the impact of different train-test split ratios (e.g., 70-30, 80-20) on model performance.  
     
3. **Questions**:
   * How do you split a dataset into training and testing sets in Python?  
     Using train\_test\_split() from sklearn.model\_selection.
   * What considerations should you keep in mind when choosing a train-test split ratio?  
     Dataset size, ensuring test set is representative, and avoiding data leakage.
   * How does the size of the training set impact the model's ability to generalize?  
     A smaller training set can lead to overfitting, while too small a test set might not adequately evaluate the model.

**Exercise 7: Data Preprocessing Pipeline**

1. **Objective**: Build a preprocessing pipeline to automate the data preparation process.
2. **Steps**:
   * Use sklearn.pipeline.Pipeline to create a pipeline that includes steps such as missing value imputation, feature scaling, and encoding categorical variables.
   * Fit the pipeline to the training data and transform the test data.
   * Integrate the preprocessing pipeline with a machine learning model for end-to-end training and evaluation.  
     Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

     Автоматически созданное описание
3. **Questions**:
   * What are the benefits of using a preprocessing pipeline?  
     Consistency: Ensures consistent data transformation during training and testing.

Modularity: Allows for easy modification and reuse of different steps.

Efficiency: Automates repetitive tasks and reduces the risk of errors.

* + How does the pipeline ensure consistency between training and test data transformations?  
    The pipeline is fitted on the training data and then applied to the test data using the same parameters, ensuring consistent transformations across both datasets.
  + How can you extend the pipeline to include additional preprocessing steps?  
    You can add custom transformers for specific feature engineering, include feature selection steps, or integrate advanced imputation methods like KNN imputation by adding them to the pipeline's steps.