HyperionDev

Datasets and DataFrames

Model Answer Approach

Visit our website

Auto-graded task 1

The approach taken begins with a detailed exploration of methods for selecting rows and columns, such as using double square brackets to extract specific columns from a DataFrame, utilising `.iloc[]` for positional indexing of rows and columns, and leveraging `.loc[]` for accessing data based on row labels and column names.

Furthermore, the solution demonstrates effective filtering strategies to isolate rows that meet specific conditions, exemplified by filtering data where the `Card` column equals `4`. Additionally, it showcases the application of the `.sort_values()` method to arrange data from the `Education` column in descending order, enhancing clarity and organisation within the dataset.

Common pitfalls you might encounter while working with `.loc` and `.iloc` include indexing errors due to improper use of row or column labels, and confusion between zero-based indexing in `.iloc` and label-based indexing in `.loc`.

Auto-graded task 2

The approach taken involves a series of operations to analyse the dataset's income and demographic data. Initially, the average income per ethnicity is calculated using the `groupby()` method, which aggregates the income data by ethnicity and computes the mean for each group. Specific ethnic groups such as African American, Asian, and Caucasian are then isolated to determine their average income values individually.

The analysis further includes evaluating the average account balance based on marital status by grouping the data according to the “Married” status and calculating the mean balance for each category. The script also identifies and prints the highest and lowest income values in the dataset by using the `max()` and `min()` methods on the “Income” column, respectively.

Gender distribution is assessed by counting occurrences of each gender with the `value_counts()` method, and the script provides counts of females and males separately by filtering the dataset.

One common issue to watch out for is ensuring that the column names used in the code match exactly with those in the dataset, as any discrepancies can lead to errors. Additionally, be cautious when handling null values and missing data, which can affect calculations.