Certainly! Below is a table that explains the different ways you can filter, access, and manipulate DataFrame columns in Pandas, specifically focusing on techniques like pay[pay['xyz'] == z] and pay['salary'].

Concept	**Description**	**Example**
Filtering with Boolean Indexing (pay[pay['xyz'] == z])	This method is used for selecting rows based on a condition applied to a column. The condition returns a boolean mask, which is then used to filter the rows.	pay[pay['Salary or Hourly'] == 'Hourly'] filters the DataFrame for rows where the 'Salary or Hourly' column is equal to 'Hourly'.
Selecting a Single Column (pay['salary'])	Directly accesses a single column of the DataFrame, returning it as a Series.	pay['Salary'] selects the 'Salary' column of the DataFrame.
Chaining Conditional Filters (pay[pay['xyz'] == z][pay['abc'] > 100])	This method chains multiple conditions, applying one filter after the other to further narrow down the DataFrame.	pay[pay['Salary or Hourly'] == 'Hourly'][pay['Typical Hours'] > 30] filters the DataFrame first for 'Hourly' employees, then for those with more than 30 hours.
Using loc[] for Label-based Indexing (pay.loc[condition])	The loc[] method is used for selecting rows based on a condition, while also allowing you to specify columns to be returned. It's label-based indexing.	pay.loc[pay['Salary or Hourly'] == 'Salary', 'Salary'] selects the 'Salary' column for rows where 'Salary or Hourly' is 'Salary'.
Using iloc[] for Integer-based Indexing (pay.iloc[rows, cols])	The iloc[] method is used for selecting rows and columns based on integer positions rather than labels.	pay.iloc[0:5, 0:2] selects the first 5 rows and the first 2 columns of the DataFrame.
Using query() Method (pay.query('condition'))	The query() method allows you to filter rows based on a string condition, using column names directly without quotes.	pay.query('Salary == "Hourly" and "Typical Hours" > 30') filters rows where 'Salary' is 'Hourly' and 'Typical Hours' is greater than 30.
Using at[] for Scalar Access (pay.at[row, col])	The at[] method is used for getting or setting a single value in a DataFrame, based on row and column labels.	pay.at[3, 'Salary'] returns the value in the 'Salary' column for the row at index 3.

Concept	Description	Example
Using iat[] for Scalar Access by Position (pay.iat[row, col])	The iat[] method is used for getting or setting a single value in a DataFrame, based on row and column positions (integers).	pay.iat[3, 2] returns the value at the 3rd row and 2nd column (integer positions).